AULA 8 - PROJETO DE ARQUITETURA DE SOFTWARE

Objetivos da Aula

- **Conteúdo:**
  - Compreender o que é arquitetura de software e sua importância fundamental no desenvolvimento de sistemas.
  - Identificar as principais decisões que precisam ser tomadas durante o projeto de arquitetura.
  - Conhecer e aplicar Padrões de Arquitetura comuns para resolver problemas de projeto recorrentes.
  - Explorar exemplos de arquiteturas de aplicações típicas.
- **Nota:** "A arquitetura de software é o 'esqueleto' do sistema; decisões tomadas aqui têm um impacto profundo e duradouro em sua qualidade, manutenibilidade e evolução."

Projeto Software

- **Identificar Nossos Componentes Principais e O relacionamento desses componentes.**
  - Atividade Criativa
  - Ligada da Criação até a Solução até sua Implementação

Projeto Software

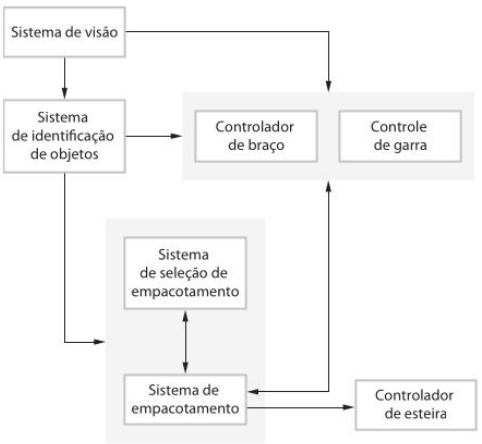
- **Projeto Alto Nível (Conceito) → Projeto Detalhado**
  1. **Contexto e interações externas com o sistema**
  2. **Arquitetura do Sistema**
  3. **Principais Objetos**
  4. **Modelos de Projeto**
  5. **Interfaces**

O que é Projeto de Arquitetura de Software?

- **Definição Principal (Sommerville, Cap. 6):**
  - "O projeto de arquitetura está preocupado com a **compreensão de como um sistema deve ser organizado** e com a **estrutura geral desse sistema**."
  - É o primeiro estágio no processo de projeto de software e serve como um elo crítico entre o projeto e a engenharia de requisitos.
  - Identifica os principais componentes estruturais de um sistema e os relacionamentos entre eles.

A arquitetura de um sistema de controle robotizado

Figura 6.1 A arquitetura de um sistema de controle robotizado de empacotamento



O que é Projeto de Arquitetura de Software?

- **Resultado do Processo:**
  - Um **modelo de arquitetura** que descreve como o sistema está organizado em um conjunto de componentes de comunicação.
- **Contexto no Processo de Software:**
  - Em processos dirigidos a planos: Uma fase distinta após a engenharia de requisitos.
  - Em processos ágeis: Geralmente, um estágio inicial para estabelecer uma arquitetura global, com desenvolvimento incremental subsequente. A refatoração arquitetural é mais cara e complexa que a de código.

Por que a Arquitetura de Software é Importante?

Análise de Sistema:

impacto profundo na capacidade do sistema de atender a requisitos críticos não funcionais (desempenho, confiabilidade, segurança, manutenibilidade).

1

Comunicação com Stakeholders:

alto nível do sistema, compreensível

2

3

Reúso em Larga Escala:

Afeta diretamente o desempenho, a robustez, a capacidade de distribuição e a manutenibilidade de um sistema.

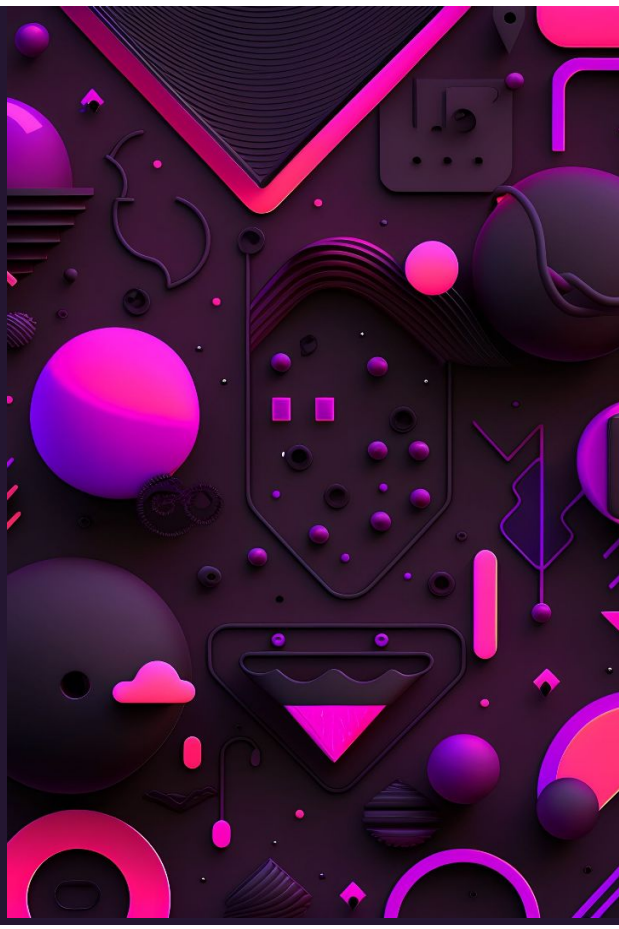
Decisões de Projeto de Arquitetura

Questões Fundamentais a Serem Consideradas (Sommerville, Cap. 6):

- Existe uma **arquitetura genérica de aplicação** que pode atuar como modelo?
- Como o sistema será **distribuído** (núcleos, processadores, redes)?
- Quais **padrões ou estilos de arquitetura** podem ser usados?

Decisões de Projeto de Arquitetura

- Qual será a **abordagem fundamental para estruturar** o sistema (ex: camadas, módulos)?
- Como os componentes estruturais serão **decompostos** em subcomponentes?
- Que estratégia será usada para **controlar o funcionamento** dos componentes?
- Qual a melhor organização de arquitetura para satisfazer os **requisitos não funcionais**?



Relação com Requisitos Não Funcionais:

Desempenho

Localizar operações críticas, minimizar comunicação entre componentes distantes.

Segurança (Safety):

Localizar operações de segurança em poucos componentes para facilitar a validação.

Proteção (Security):

Arquitetura em camadas com ativos críticos nas camadas internas.

Manutenibilidade e Disponibilidade

Componentes autocontidos e de baixa granularidade.

Padrões de Arquitetura (Estilos Arquiteturais)

- **Definição (Adaptado de Shaw & Garlan, 1996, "Software Architecture: Perspectives on an Emerging Discipline"):**
  - "Um padrão de arquitetura é uma **descrição abstrata e estilizada de boas práticas de organização de sistemas**, experimentadas e testadas em diferentes sistemas e ambientes."
  - Encapsulam conhecimento sobre como organizar sistemas para resolver problemas recorrentes.
- **Argumento:** Padrões promovem o reúso de conhecimento arquitetural, facilitam a comunicação e ajudam a construir sistemas mais robustos e manuteníveis.

Padrão de Arquitetura: Modelo-Visão-Controlador (MVC)

- **Nome:** MVC (Model-View-Controller)
- **Descrição (Sommerville, Cap. 6, Tabela 6.1):**
  - Separa a **apresentação** (Visão), a **interação do usuário** (Controlador) e os **dados do sistema e lógica de negócio** (Modelo)
  - **Modelo (Model):** Gerencia os dados e a lógica de negócio.
  - **Visão (View):** Renderiza o Modelo em uma forma adequada para interação do usuário.
  - **Controlador (Controller):** Processa a entrada do usuário (vinda da Visão), interage com o Modelo para atualizar dados ou executar lógica, e seleciona a Visão apropriada para exibir a resposta.

Figura 6.3 Arquitetura de aplicações Web usando o padrão MVC

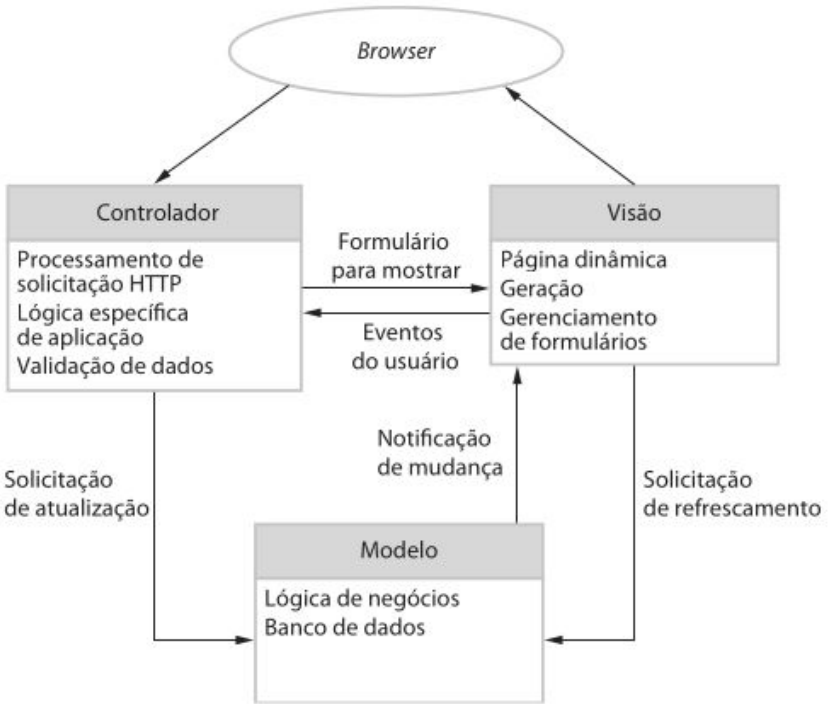
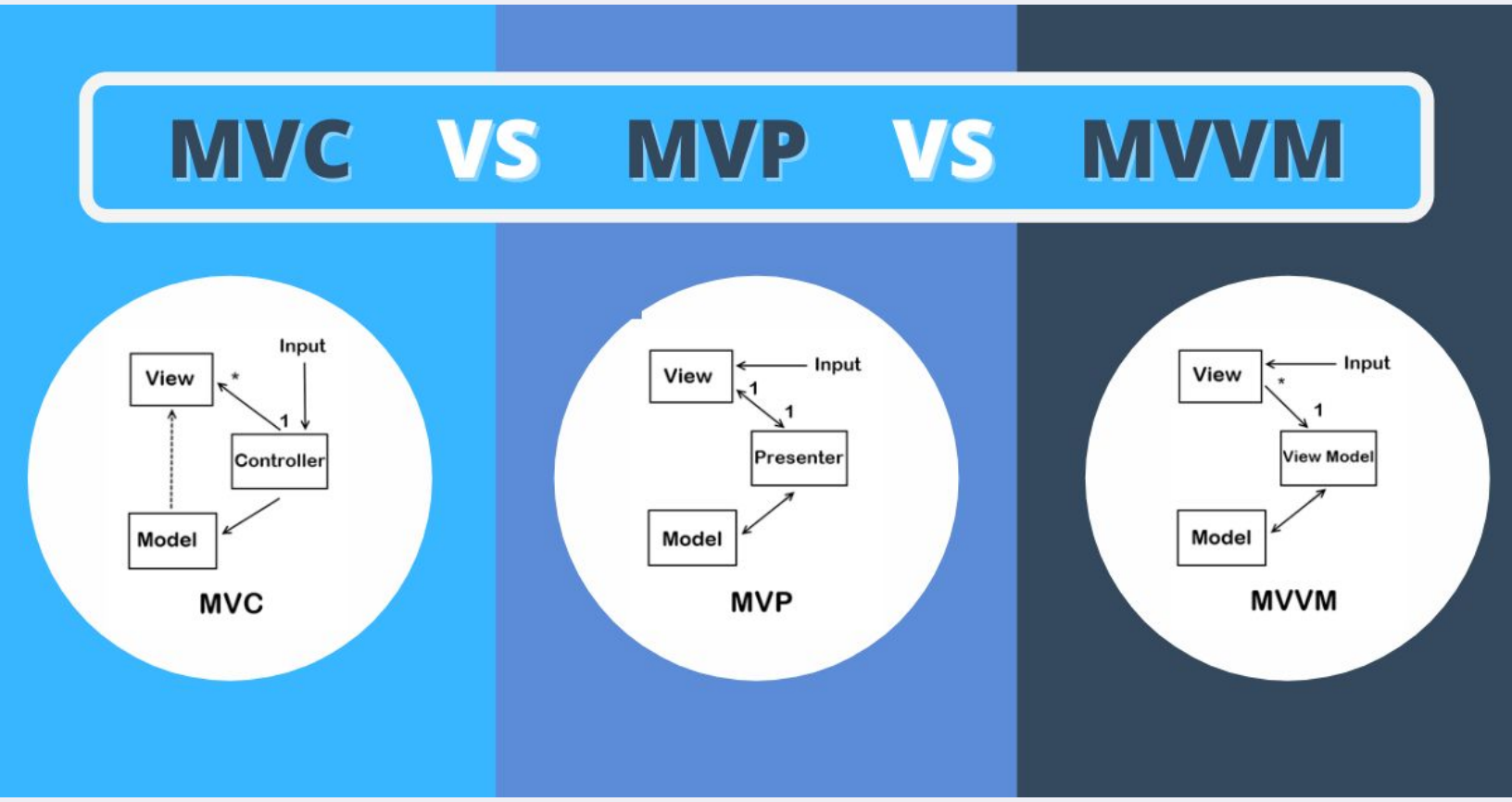


Tabela 6.1 O padrão modelo-visão-controlador (MVC)

Nome	MVC (Modelo-Visão-Controlador)
Descrição	Separa a apresentação e a interação dos dados do sistema. O sistema é estruturado em três componentes lógicos que interagem entre si. O componente Modelo gerencia o sistema de dados e as operações associadas a esses dados. O componente Visão define e gerencia como os dados são apresentados ao usuário. O componente Controlador gerencia a interação do usuário (por exemplo, teclas, cliques do mouse etc.) e passa essas interações para a Visão e o Modelo. Veja a Figura 6.2.
Exemplo	A Figura 6.3 mostra a arquitetura de um sistema aplicativo baseado na Internet, organizado pelo uso do padrão MVC.
Quando é usado	É usado quando existem várias maneiras de se visualizar e interagir com dados. Também quando são desconhecidos os futuros requisitos de interação e apresentação de dados.
Vantagens	Permite que os dados sejam alterados de forma independente de sua representação, e vice-versa. Apoia a apresentação dos mesmos dados de maneiras diferentes, com as alterações feitas em uma representação aparecendo em todas elas.
Desvantagens	Quando o modelo de dados e as interações são simples, pode envolver código adicional e complexidade de código.

Comparação: MVC, MVP e MVVM

Arquitetura	Responsabilidades	Vantagens	Desvantagens
MVC	Model, View, Controller	Simplicidade, boa separação de conceitos	Controller pode ficar sobrecarregado
MVP	Model, View, Presenter	Melhor testabilidade, separação mais clara	Código mais complexo
MVVM	Model, View, ViewModel	Forte desacoplamento, reatividade, testabilidade	Maior curva de aprendizado



Padrão de Arquitetura: Arquitetura em Camadas

- **Nome:** Arquitetura em Camadas (Layered Architecture)
- **Descrição (Sommerville, Cap. 6, Tabela 6.2):**
  - Organiza o sistema em um conjunto de camadas hierárquicas.
  - Cada camada fornece serviços à camada imediatamente superior e usa serviços da camada imediatamente inferior.
  - Camadas superiores dependem das inferiores, mas as inferiores são independentes das superiores.

Arquitetura em Camadas

Figura 6.4 Uma arquitetura genérica em camadas

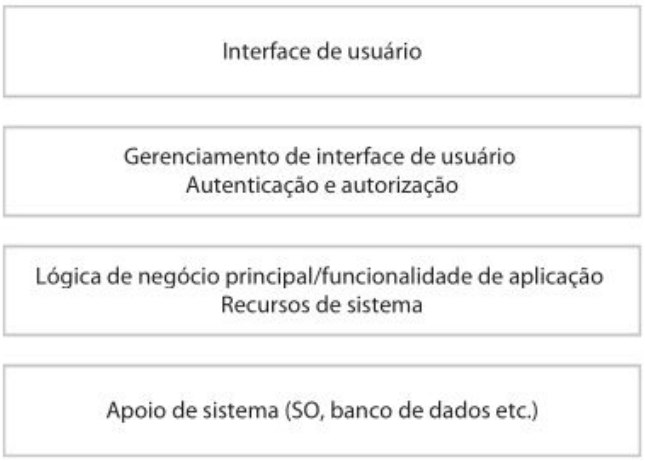


Tabela 6.2 O padrão de arquitetura em camadas

Nome	Arquitetura em camadas
Descrição	Organiza o sistema em camadas com a funcionalidade relacionada associada a cada camada. Uma camada fornece serviços à camada acima dela; assim, os níveis mais baixos de camadas representam os principais serviços suscetíveis de serem usados em todo o sistema. Veja a Figura 6.4.
Exemplo	Um modelo em camadas de um sistema para compartilhar documentos com direitos autorais, em bibliotecas diferentes, como mostrado na Figura 6.5.
Quando é usado	É usado na construção de novos recursos em cima de sistemas existentes; quando o desenvolvimento está espalhado por várias equipes, com a responsabilidade de cada equipe em uma camada de funcionalidade; quando há um requisito de proteção multinível.
Vantagens	Desde que a interface seja mantida, permite a substituição de camadas inteiras. Recursos redundantes (por exemplo, autenticação) podem ser fornecidos em cada camada para aumentar a confiança do sistema.
Desvantagens	Na prática, costuma ser difícil proporcionar uma clara separação entre as camadas, e uma camada de alto nível pode ter de interagir diretamente com camadas de baixo nível, em vez de através da camada imediatamente abaixo dela. O desempenho pode ser um problema por causa dos múltiplos níveis de interpretação de uma solicitação de serviço, uma vez que são processados em cada camada.

Padrão de Arquitetura: Outros Padrões

- **Arquitetura de repositório**
  - Todos os dados em um sistema são gerenciados em um repositório central, acessível a todos os componentes do sistema. Os componentes não interagem diretamente, apenas por meio do repositório.
- **Arquitetura cliente-servidor**
  - Em uma arquitetura cliente-servidor, a funcionalidade do sistema está organizada em serviços — cada serviço é prestado por um servidor.
- **Arquitetura de duto e filtro**
  - O processamento dos dados em um sistema está organizado de modo que cada componente de processamento(filtro) seja discreto e realize um tipo de transformação de dados.

Vamos Codar?