Projeto de Sistemas de Informação

Arquitetura de Software

Lesandro Ponciano

2024

“Nada que é visto, é visto de uma vez e por completo.”

Euclides (330 - 277 a.C.)

Objetivos da Aula

- Contextualizar arquitetura de software em Projeto de Sistemas de Informação
- Analisar as diferentes visões da arquitetura
- Discutir os principais estilos arquiteturais
  - Arquiteturas em Camadas
  - Arquiteturas de Repositórios
  - Dutos e Filtros
  - Cliente-servidor
  - *Model-View-Controller*

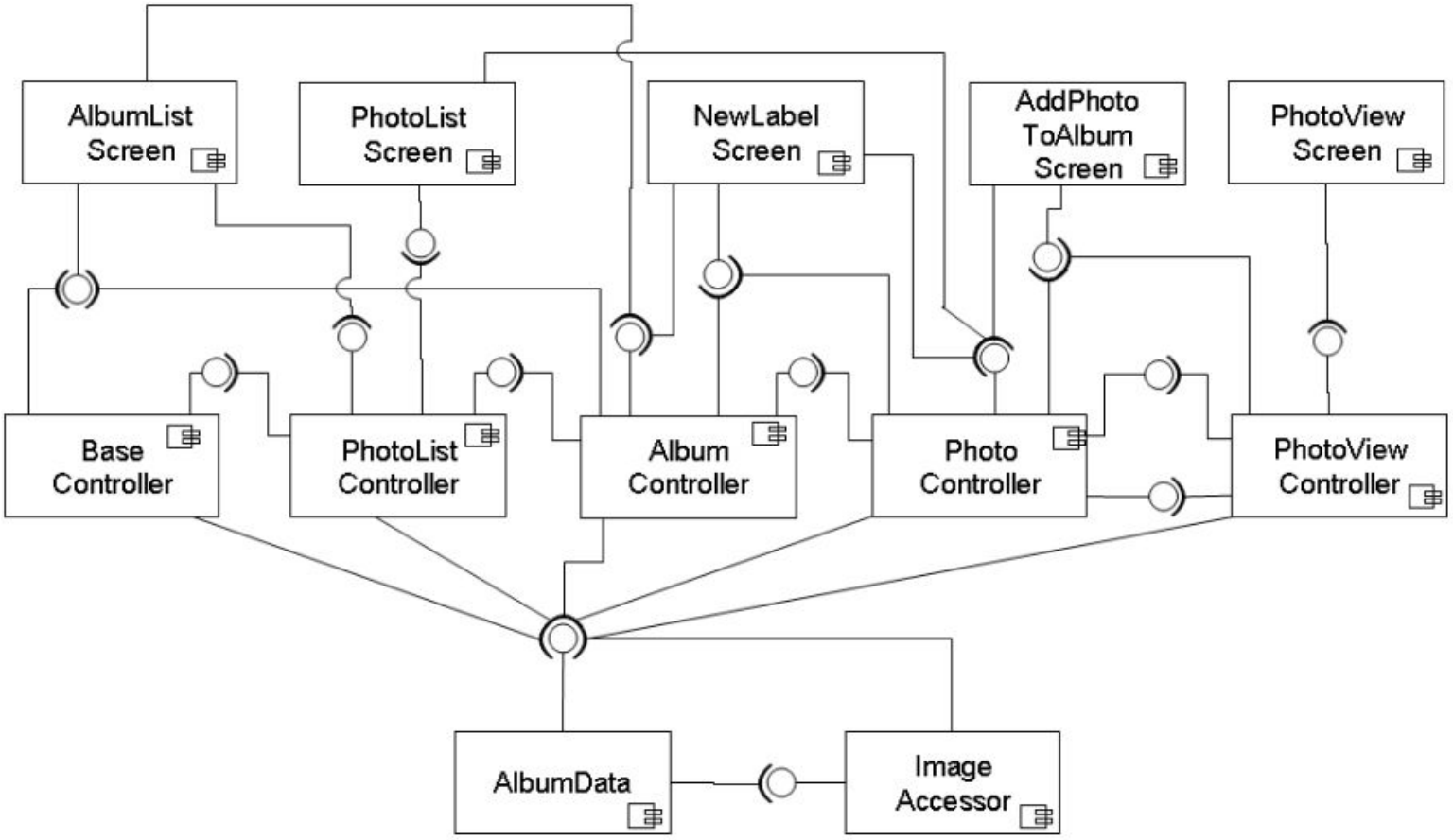
Arquitetura de Software

"*Software architecture is the structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time*"

(Garlan and Perry, IEEE TSE, April 1995)

- A arquitetura de um software é uma estrutura de componentes interconectados por meio de interfaces
  - Componentes são compostos de componentes menores e interfaces
  - A interação entre componentes ocorre através de suas interfaces

Exemplo de Arquitetura



Decisões de Projeto da Arquitetura

- Arquitetos precisam considerar as seguintes questões
  1) Existe alguma arquitetura genérica de aplicação que pode atuar como um modelo para o sistema que está sendo projetado?
  2) Como o sistema será distribuído por meio de um número de núcleos ou processadores?
  3) Que padrões ou estilos arquiteturais podem ser usados?
  4) Qual será a abordagem fundamental para se estruturar o sistema?
  5) Como os componentes estruturais do sistema serão decompostos em subcomponentes?
  6) Que estratégia será usada para controlar o funcionamento dos componentes do sistema?
  7) Qual a melhor organização da arquitetura para satisfazer os requisitos não funcionais do sistema?
  8) Como o projeto de arquitetura será avaliado?
  9) Como a arquitetura do sistema deve ser documentada?

Requisitos Não-Funcionais

- Há estreita relação entre requisitos não funcionais e a arquitetura do software
  - A arquitetura pode ser definida em observância a tais requisitos

- Pontos principais
  - Desempenho: ex.: menos distribuição e menos uso de rede
  - Proteção: ex.: uso de camadas com proteção das camadas inferiores
  - Segurança: ex.: centralizar em poucos componentes as operações de segurança
  - Disponibilidade: ex: maior redundância
  - Manutenção: ex.: componentes menores que podem ser rapidamente alterados/substituídos

Documentação da Arquitetura

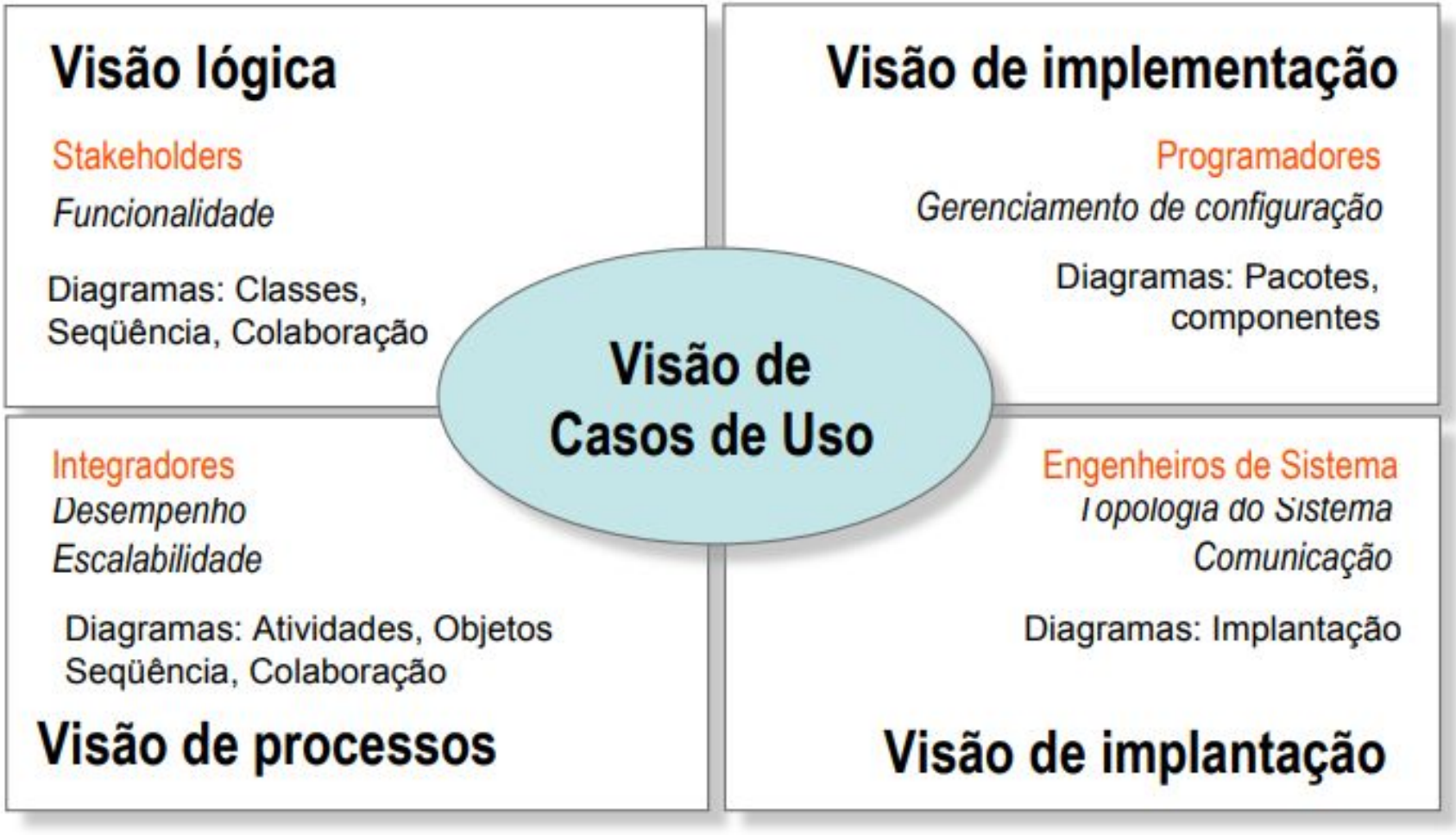
- Há três razões principais para documentar a arquitetura

1) Comunicação com stakeholders
   - É uma representação em alto nível do sistema e pode ser usada para conversar com os interessados
2) Análise de sistema
   - As decisões de arquitetura têm efeito profundo sobre a possibilidade do sistema atender ou não requisitos críticos
3) Reuso em larga escala
   - A arquitetura geralmente é a mesma para sistemas semelhantes

Visões da Arquitetura

- É importante separar diferentes aspectos da arquitetura em visões separadas
  - Isso permite gerenciar a complexidade
- Benefícios
  - Cada visão descreve diferentes conceitos da engenharia
  - Reduz a quantidade de informação que o arquiteto trata em um dado momento
- Muitos arquitetos usam as diferentes visões sem reconhecê-las como visões arquiteturais separadas

Modelo de Visão 4+1



Estilos Arquiteturais

- Arquiteturas em Camadas
- Arquiteturas de Repositórios (*Blackboard*)
- Dutos e Filtros (*Pipes* and *filters*)

Da desordem à estrutura

- Cliente-servidor
- Brokers

Sistemas Distribuídos

- Model-view-Controller
- Presentation-abstraction-control

Sistemas Interativos

- Microkernel
- Microservices
- Reflection

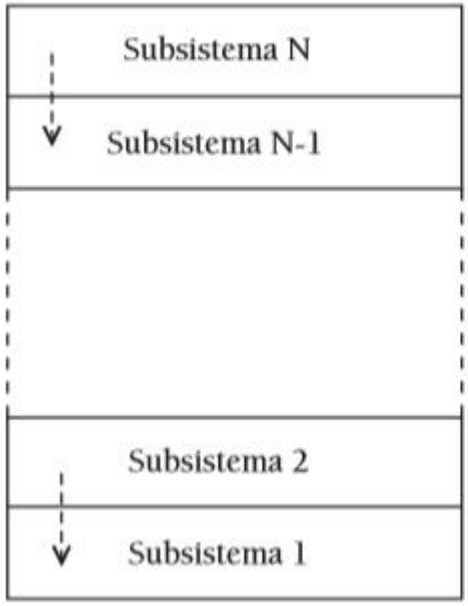
Sistemas Adaptáveis

Arquitetura em Camadas

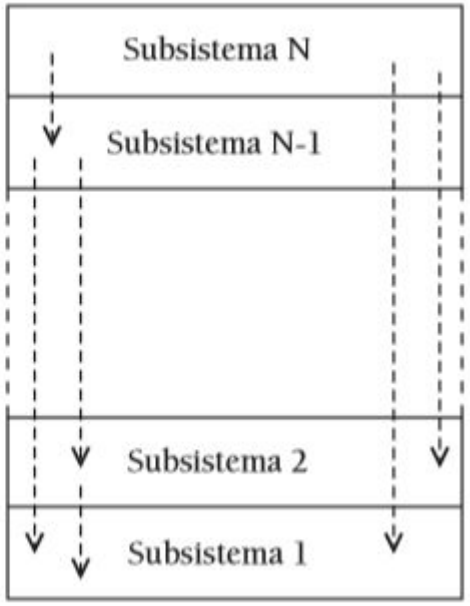
- Características
  - O sistema é organizado em um conjunto de camadas
  - Cada camada oferece um conjunto de serviços
  - Uma camada *C* somente solicita serviços da camada inferior (*C-1*) e fornece serviços para a camada superior (*C+1*)

- Vantagens
  - Camadas podem ser facilmente substituídas por outras equivalentes, desde que tenham interfaces estáveis
  - Mudanças na camada *C* teoricamente só impacta a camada *C+1*
  - Camadas superiores podem ser independentes de plataforma/hardware

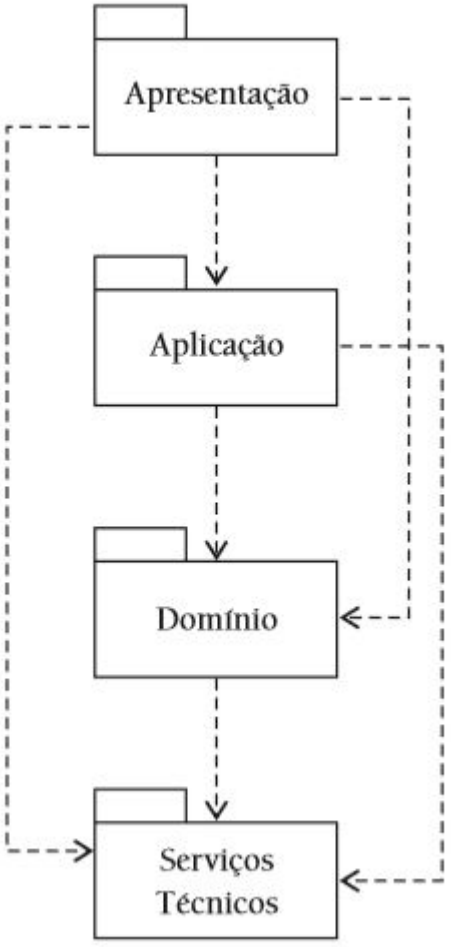
Camadas Lógicas



Arquitetura Fechada



Arquitetura Aberta



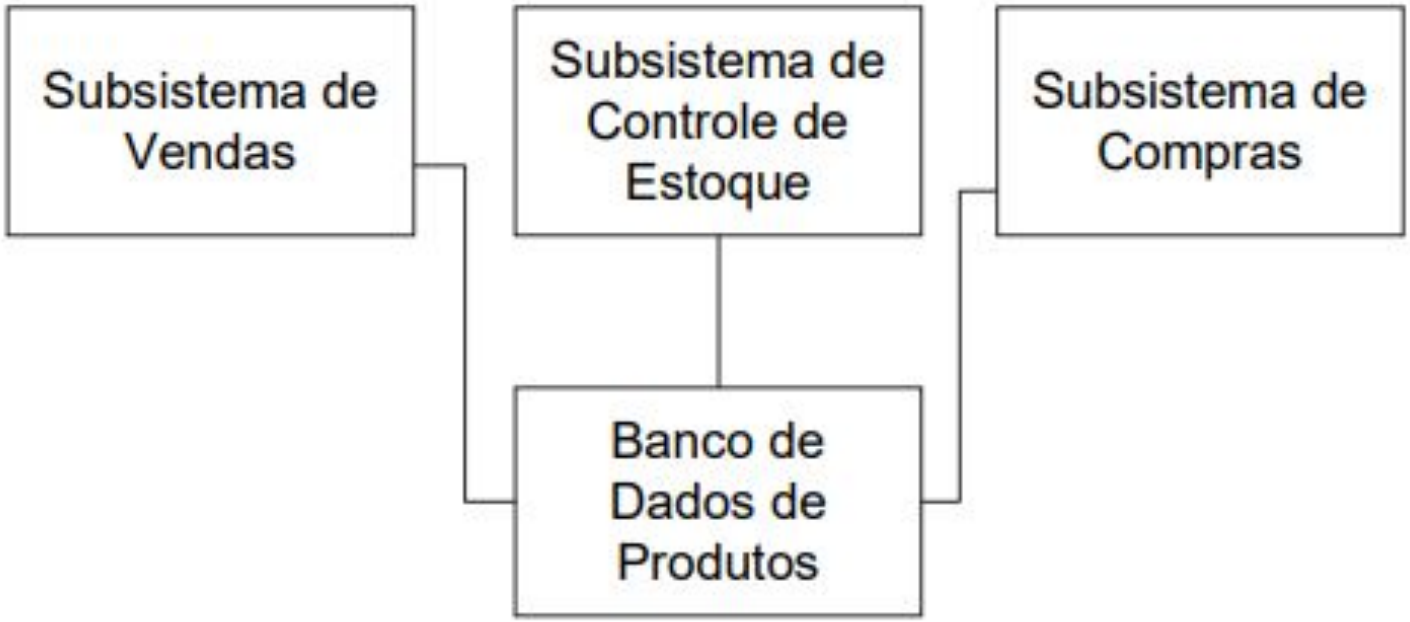
Exemplo UML de arquitetura aberta

Arquitetura de Repositório

- O uso desse estilo é comum principalmente quando grandes quantidades de dados são compartilhados entre subsistemas

- Os subsistemas manipulam uma base de dados compartilhada entre eles
  - Um (ou mais) subsistema escreve os dados
  - Vários subsistemas leem os dados

Exemplo (não é UML)



Características

- Vantagens
  - Maneira eficiente de compartilhar dados
  - *Backup* é centralizado (mais fácil)
  - Formas de proteção dos dados podem ser implementadas
  - Os subsistemas que gravam dados não necessitam saber quem os usa
  - Fácil integrar novos subsistemas

- Desvantagens
  - Os subsistemas devem entender o formato dos dados
  - Manter e evoluir grandes volumes de dados pode ser difícil
  - Subsistemas diferentes podem ter requisitos diferentes associados a dados

Dutos e Filtros

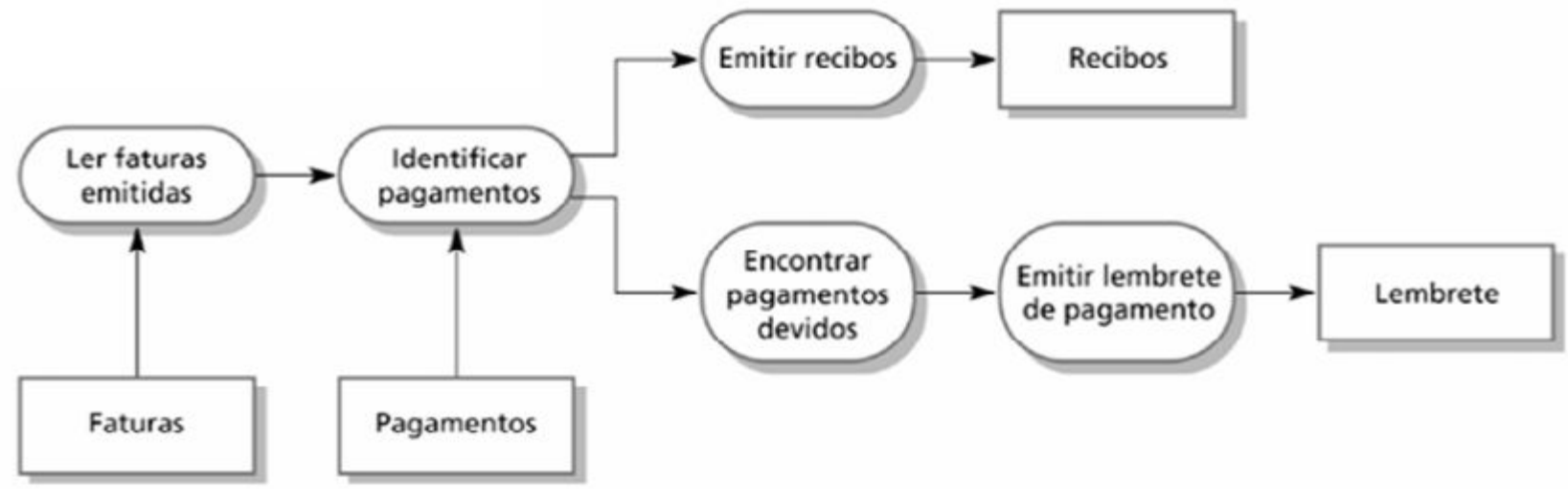
- Padrão de organização da dinâmica de um sistema e que é usado principalmente em aplicações de processamento de dados

- Dois papéis principais
  - Dutos: componentes que conduzem ou distribuem os dados
  - Filtros: componentes que transformam os dados

- Os dados de entrada
  - se movem pelos dutos
  - são transformados pelos filtros até serem convertidos em dados de saída

Exemplo (não é UML)

- Entradas: Faturas e Pagamentos
- Saídas: Recibos e Lembretes



Características

- Vantagem
  - O módulo de transformação (filtro) é modular
  - É simples evoluir o sistema pela adição de filtros
  - Se aplica tanto a sistemas sequenciais quanto a sistemas concorrentes

- Desvantagem
  - O formato dos dados trafegados deve ser acordado entre os módulos, para que todos saibam operar sobre eles
  - Pode haver um *overhead* causado pela padronização dos dados
  - Incompatibilidade no formato dos dados pode dificultar o reuso de filtros

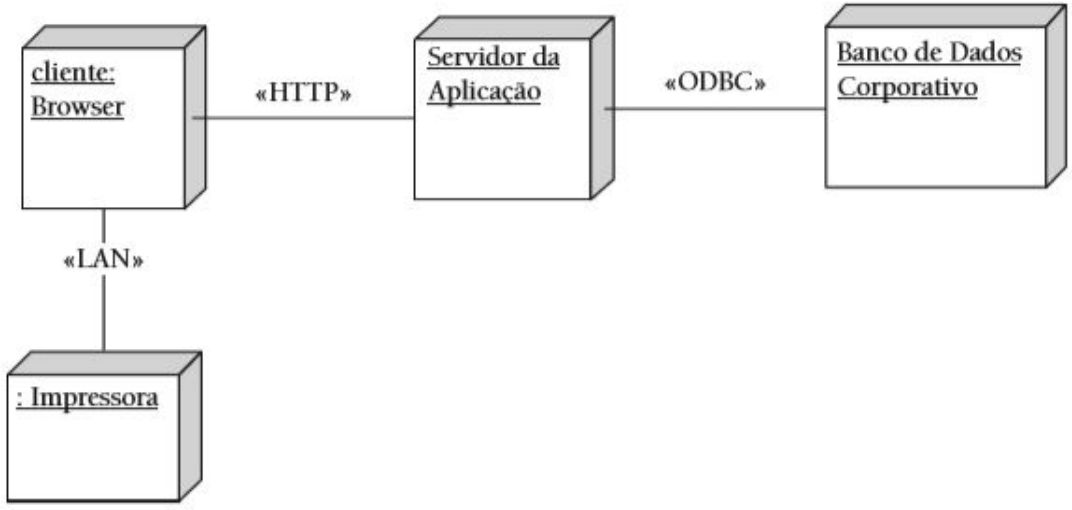
Arquitetura Cliente-Servidor

- Organizada como um conjunto de serviços
  - Servidores dos serviços
  - Clientes dos serviços

- Exemplos de servidores
  - Servidor de dados
  - Servidor de impressão
  - Servidor de arquivos

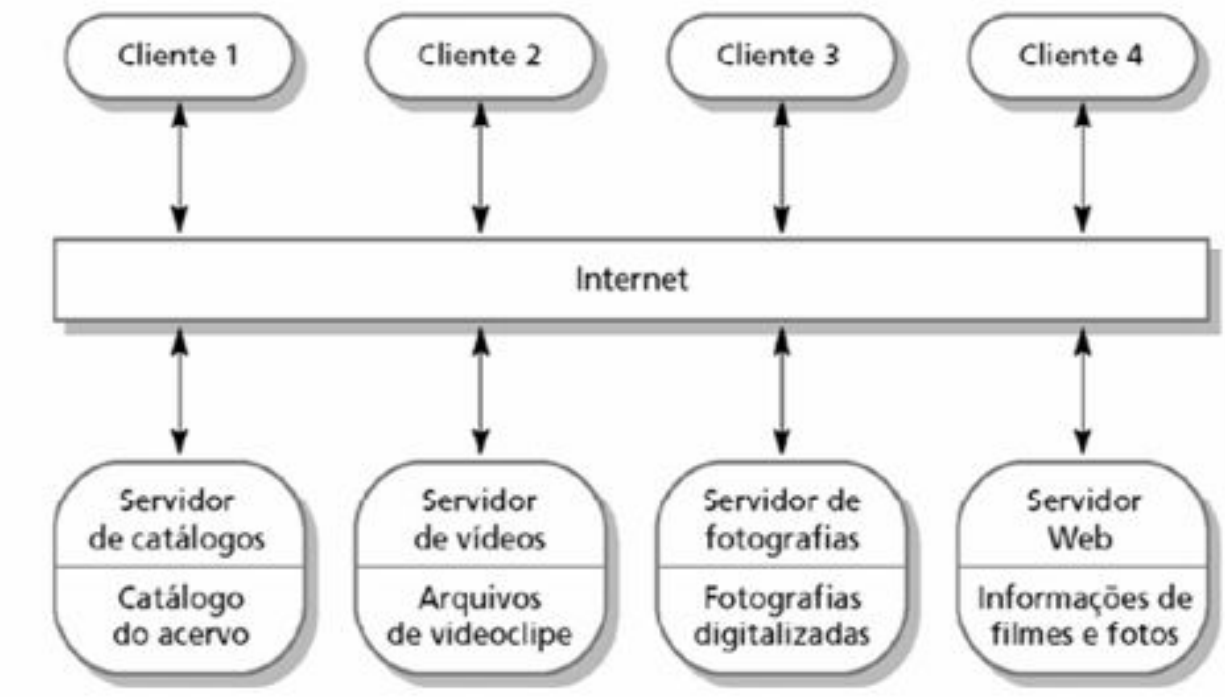
Implantação Física

Cliente-servidor de 3 camadas físicas



Exemplo Detalhado

Exemplo (não é UML)



Características

- Requer uma estrutura de rede para clientes acessarem os serviços
- Clientes sabem quais os serviços e servidores estão disponíveis
- Servidores não sabem quem são os clientes
- É usado o protocolo “requisição - resposta” (*request-reply*)

Características

- Vantagens
  - A distribuição de dados é fácil e direta
  - Faz uso efetivo dos recursos em rede
  - É fácil adicionar novos servidores ou atualizar servidores existentes

- Desvantagens
  - Não prevê modelo de dados compartilhado
  - Pode haver redundância de serviços em diferentes servidores
  - Não prevê registro central de serviços, é difícil descobrir quais serviços estão disponíveis

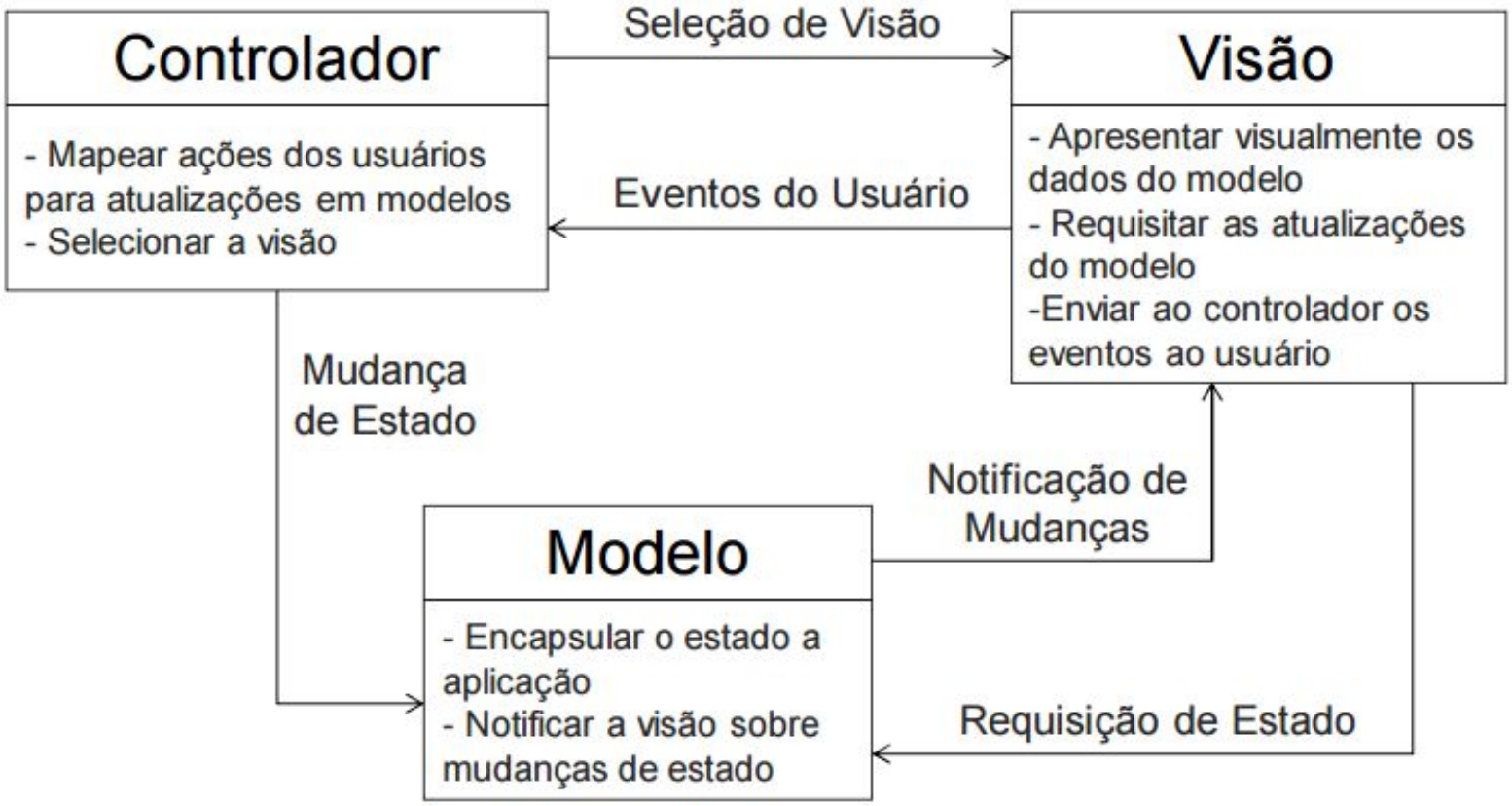
Modelo-Visão-Controlador

- Separa a apresentação e a interação dos dados do sistema

- Organiza o sistema em três componentes que têm responsabilidades distintas mas interagem entre si
  - Modelo (*model*): contém as funcionalidades e dados principais
  - Visão (*view*): responsável por apresentar os dados ao usuário
  - Controlador (*controller*): trata os eventos de entrada

- Quando é recomendado?
  - Há várias maneiras de visualizar e interagir com os dados
  - Os requisitos de interação com os dados são desconhecidos (ou são voláteis)

Visão Geral (não é UML)



Características

- Vantagens
  - Permite que os dados sejam alterados independente de sua representação (e vice versa)
  - Apoia a apresentação dos mesmos dados de maneiras diferente
  - Facilita a distribuição do componente de visão, os dados são mantidos centralizados e protegidos

- Desvantagens
  - Complexidade excessiva quando o modelo de dados e de interações é muito simples
  - A estrutura do padrão pode impor código adicional desnecessário

Atividade de Fixação

1. Explique o que é arquitetura de software.
2. Exemplifique o que o arquiteto deve se questionar ao tomar decisões de projeto associadas à arquitetura do software?
3. Explique o modelo de visão 4+1.
4. Discuta uma característica marcante de cada um seguintes estilos arquiteturais: I) Arquiteturas em Camadas; II) Arquiteturas de Repositórios; III) Dutos e Filtros; IV) Cliente-servidor; V) *Model-view-Controller.*
5. Proponha uma versão em UML para os seguintes diagramas:
   a. Diagrama do slide 15, usar diagrama de componentes
   b. Diagrama do slide 18, usar diagrama de componentes
   c. Diagrama do slide 22, usar diagrama de implantação
   d. Representação do MVC (slide 26), usar diagrama de pacotes

Referências

Summerville, Ian. Engenharia de Software. 9.ed (**Capítulo 6**)

BEZERRA, E. Princípio de Análise e Projeto de Sistemas com UML, Rio de Janeiro, Elsevier, 2007. **(Capítulo 11)**

GUEDES, Gilleanes T. A. UML 2: uma abordagem prática. 2. ed. São Paulo: Novatec, c2011. 484 p. ISBN 9788575222812 **(Capítulo 6, 12 e 13)**

Projeto de Software

Prof. Dr. Lesandro Ponciano

https://orcid.org/0000-0002-5724-0094