UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE RUSSAS

Arquitetura de Software

Estilos e Padrões Arquiteturais

Profa. Dra. Anna Beatriz Marques

Contextualização

- O projeto da arquitetura é uma atividade complexa e desafiadora
- Arquitetos buscam formas de capturar e reutilizar o **conhecimento arquitetural** sobre **boas soluções** arquiteturais
- Padrões emergem da prática, não podem ser inventados, e sim descobertos

“Arquitetos experientes geralmente consideram o design arquitetural como um processo de selecionar, personalizar e combinar padrões”

1

Estilos Arquiteturais X Padrões Arquiteturais

Conceitos

- Um estilo arquitetural é uma transformação imposta ao projeto de um sistema inteiro
  - Estabelece uma estrutura para todos os componentes do sistema
- Um padrão arquitetural também impõe uma transformação no projeto de arquitetura, porém:
  - O escopo de um padrão é menos abrangente, considera em um aspecto da arquitetura
  - Um padrão impõe uma regra sobre a arquitetura, em relação a alguma funcionalidade ou comportamento
  - Os padrões podem ser usados com um estilo arquitetural

Estilos Arquiteturais

- Um estilo arquitetural expressa:
  - Uma organização **estrutural**
  - Um conjunto pré-definido de **subsistemas e suas responsabilidades**
  - Inclui **regras e diretrizes para organizar** o relacionamento entre os subsistemas

- São **“templates”** para arquiteturas concretas

Padrões Arquiteturais

- Um padrão arquitetural expressa:
  - Um **conjunto de decisões** de design repetidamente adotado na prática
  - Possui **propriedades** que permitem o **reuso**
  - Descreve uma **classe de arquiteturas** (podem ser instanciadas)

- Estrutura de um padrão arquitetural:
  - **Contexto:** uma situação comum do mundo real que ocasiona um problema
  - **Problema:** um problema genérico que surge de um contexto
  - **Solução:** uma solução arquitetural de sucesso que resolve o problema

2

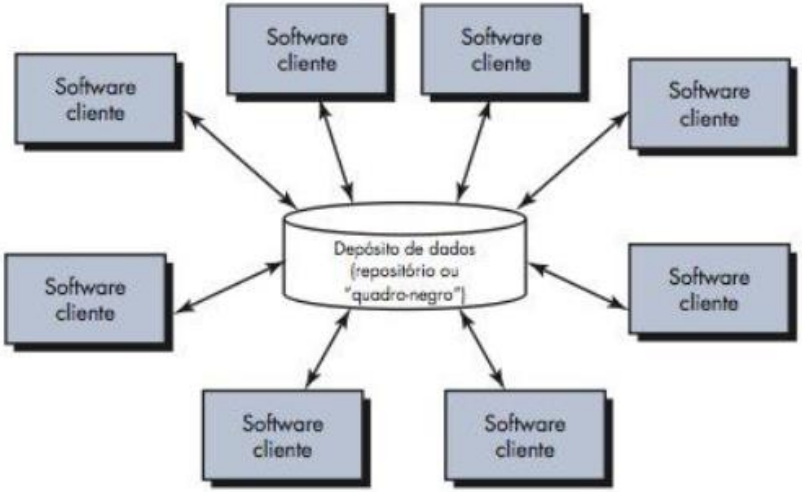
Principais Estilos Arquiteturais

Principais estilos arquiteturais

- Arquiteturas centralizadas em dados
- Arquiteturas de fluxo de dados
- Arquiteturas de chamadas e retornos
- Arquiteturas orientadas a objetos
- Arquiteturas em camadas

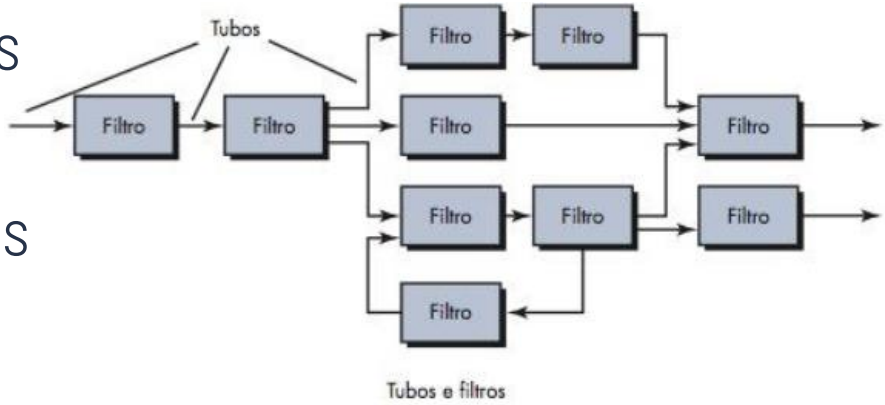
Principais estilos arquiteturais

- Arquiteturas centralizadas em dados
- Arquiteturas de fluxo de dados
- Arquiteturas de chamadas e retornos
- Arquiteturas orientadas a objetos
- Arquiteturas em camadas



Principais estilos arquiteturais

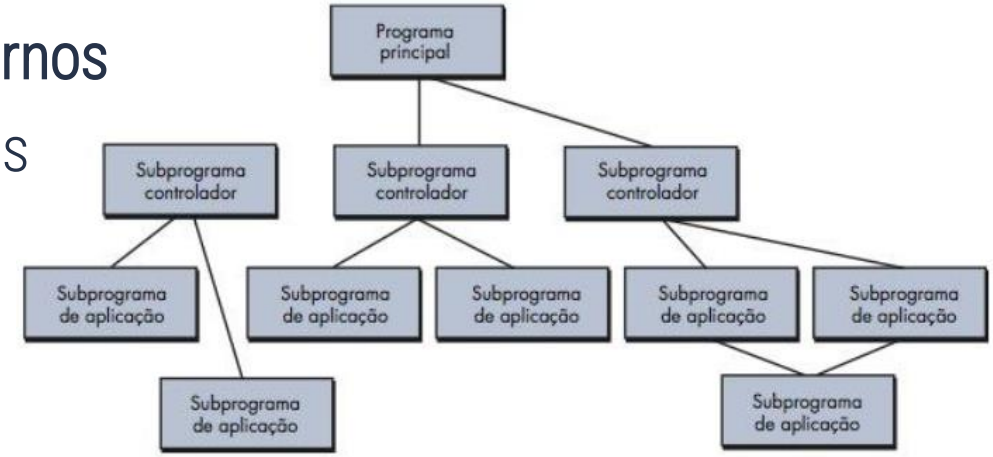
- Arquiteturas centralizadas em dados
- **Arquiteturas de fluxo de dados**
- Arquiteturas de chamadas e retornos
- Arquiteturas orientadas a objetos
- Arquiteturas em camadas



Tubos e filtros

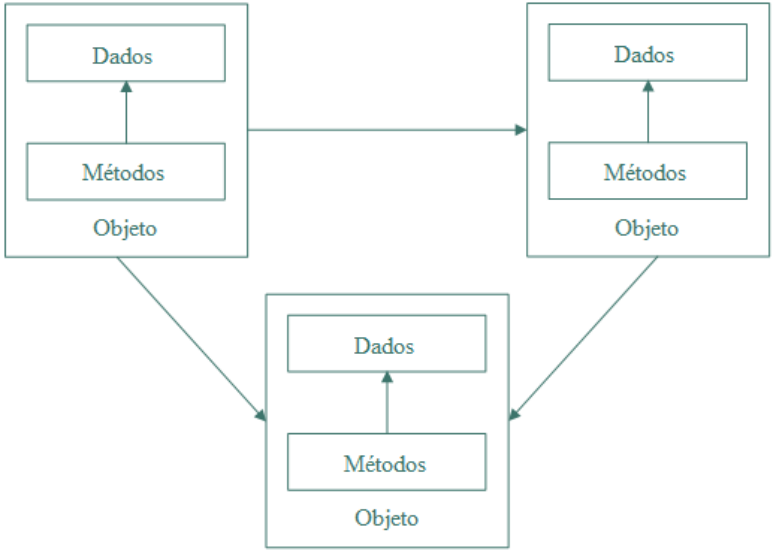
Principais estilos arquiteturais

- Arquiteturas centralizadas em dados
- Arquiteturas de fluxo de dados
- **Arquiteturas de chamadas e retornos**
- Arquiteturas orientadas a objetos
- Arquiteturas em camadas



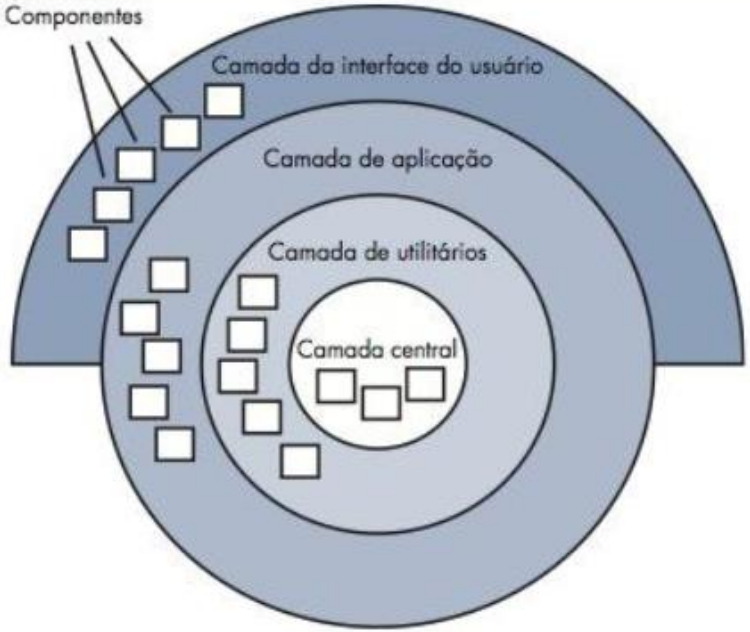
Principais estilos arquiteturais

- Arquiteturas centralizadas em dados
- Arquiteturas de fluxo de dados
- Arquiteturas de chamadas e retornos
- **Arquiteturas orientadas a objetos**
- Arquiteturas em camadas



Principais estilos arquiteturais

- Arquiteturas centralizadas em dados
- Arquiteturas de fluxo de dados
- Arquiteturas de chamadas e retornos
- Arquiteturas orientadas a objetos
- **Arquiteturas em camadas**



3

Principais Padrões Arquiteturais

Divisão em Camadas

Padrão Arquitetural: *Divisão em camadas*

- Contexto:
  - Sistemas complexos requerem que suas partes sejam **desenvolvidas e evoluídas de forma separada**
  - É necessária uma **separação de questões** tratadas pelo sistema de forma bem documentada e clara
- Problema:
  - O software precisa ser dividido para que seus módulos possam ser desenvolvidos e evoluídos de forma separada
  - O software deve fornecer **portabilidade, modificabilidade e reuso**.

Padrão Arquitetural: *Divisão em camadas*

- Solução:
  - Sistema com várias camadas de abstração
  - Camadas de níveis superiores dependem das camadas de níveis inferiores
  - Partes do sistema devem poder ser trocadas
  - Podem existir várias camadas em um mesmo nível de abstração dependendo de camadas inferiores
    - Interface gráfica cliente X Interface WEB

Padrão Arquitetural: *Divisão em camadas*

- Pontos positivos
  - Reuso das camadas
  - Dependências tendem a permanecer “locais”
- Pontos negativos
  - Cascateamento de alterações para as camadas superiores quando o comportamento de uma camada inferior muda

Cliente - Servidor

Padrão Arquitetural: *Cliente-Servidor*

- Contexto:
  - Um **grande número de clientes** deseja acessar **recursos e serviços compartilhados**
  - O sistema deve prover o **controle de acesso e qualidade do serviço**
- Problema:
  - Os serviços comuns **são divididos e organizados** em locais comuns
    - O software deve promover modificabilidade e reuso
  - Conjuntos de recursos e serviços compartilhados **são gerenciados**
    - O gerenciamento é centralizado >> escalabilidade e disponibilidade

Padrão Arquitetural: *Cliente-Servidor*

- Baseado em programas servidores e programas clientes
- Cliente
  - Estabelece a conexão, envia mensagens para o servidor e aguarda mensagens de resposta.
- Servidor
  - Aguarda mensagens, executa serviços e retorna resultados.

Padrão Arquitetural: *Cliente-Servidor*

- Vantagens
  - Utilização dos recursos do servidor
  - Escalabilidade
  - Aumentando a capacidade computacional do servidor

- Desvantagens
  - Introduz complexidade
  - Custos de comunicação

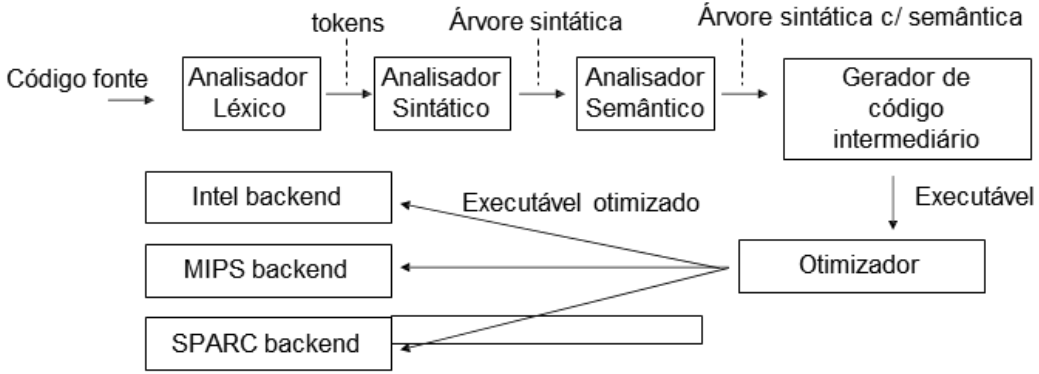
Pipes & Filters

Padrão Arquitetural: *Pipes and filters*

- Contexto:
  - Muitos sistemas precisam transformar um fluxo de dados de uma entrada para uma saída
  - Diversos tipos de transformação ocorrem repetidamente na prática, sendo desejável que possam ser desenvolvidos de forma independente, como partes reusáveis
- Problema:
  - O sistema precisa ser dividido em componentes pouco acoplados, reusáveis e com mecanismos de interação simples e genéricos

Padrão Arquitetural: *Pipes and filters*

- Solução: Dividir a tarefa entre várias etapas sequenciais
  - Saída de uma etapa é a entrada da etapa seguinte
  - Cada etapa de processamento é implementada por um filtro (filter)
    - Consome e entrega os dados incrementalmente
  - Cada "pipe" implementa o fluxo dos dados entre os filtros



Padrão Arquitetural: *Pipes and filters*

- Pontos positivos
  - Não é preciso criar arquivos intermediários (mas é possível)
  - Flexibilidade na troca de filtros
  - Flexibilidade na recombinação
  - Eficiência no processamento em paralelo
    - Vários filtros consumindo e produzindo dados em paralelo.
- Ponto negativo
  - Gerenciamento de erros
    - Ausência de um estado global compartilhado

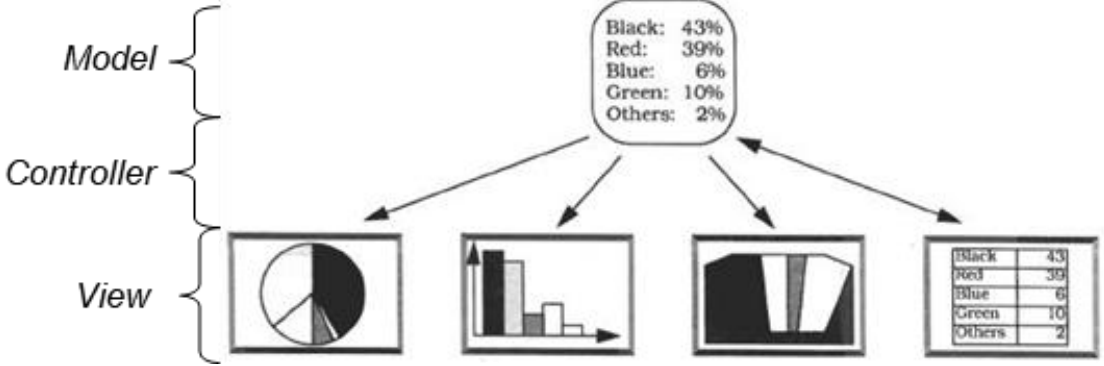
Model-View-Controller (MVC)

Padrão Arquitetural: *Model-View-Controller*

- Contexto:
  - A interface de usuário é a parte do software mais frequentemente modificada em um sistema interativo
  - É importante que as alterações possam ser realizadas separadamente do restante do sistema
  - Usuários desejam visualizar os dados de diferentes perspectivas
- Problema:
  - Como separar as funcionalidades da interface e as funcionalidades do sistema, mantendo a interface responsiva à entrada de dados e mudanças nos dados do sistema? Como fornecer múltiplas visões sobre os dados?

Padrão Arquitetural: *Model-View-Controller*

- A aplicação é dividida em 3 componentes
  - Model – contém a funcionalidade principal e os dados
  - View – exibe a informação aos usuários
  - Controller – intermedia a comunicação entre *model* e *view* e gerencia as notificações de mudanças de estado do sistema



Padrão Arquitetural: *Model-View-Controller*

- Pontos positivos
  - Múltiplas “views” de um mesmo modelo
  - “views” sincronizadas
  - Organização clara de abstrações

- Pontos negativos
  - Aumento da complexidade
  - “Controllers” e “Views” tendem a ser bastante acoplados

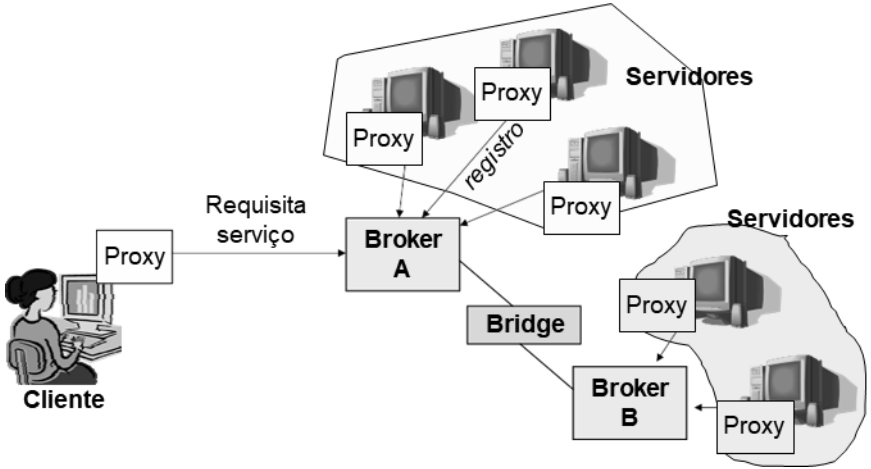
Broker

Padrão Arquitetural: *Broker*

- Contexto:
  - Muitos sistemas são desenvolvidos a partir de uma coleção de serviços distribuídos em múltiplos servidores
  - É necessário se preocupar com a interoperabilidade dos sistemas e disponibilidade dos serviços
- Problema:
  - Como estruturar o software distribuído para que a interação entre clientes e servidores seja facilmente gerenciada sem a necessidade de conhecer a localização e natureza dos servidores de serviços?

Padrão Arquitetural: *Broker*

- Estruturar sistemas distribuídos que precisam interagir através de invocação remota de serviços.

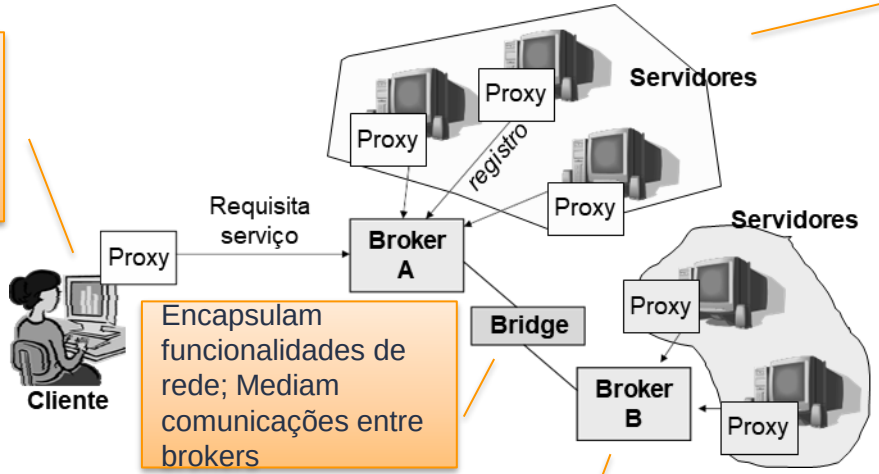


Padrão Arquitetural: *Broker*

- Estruturar sistemas distribuídos que precisam interagir através de invocação remota de serviços.

Expõem a funcionalidade através de interfaces

Implementam a funcionalidade para o usuário
Envia requisições para os servidores



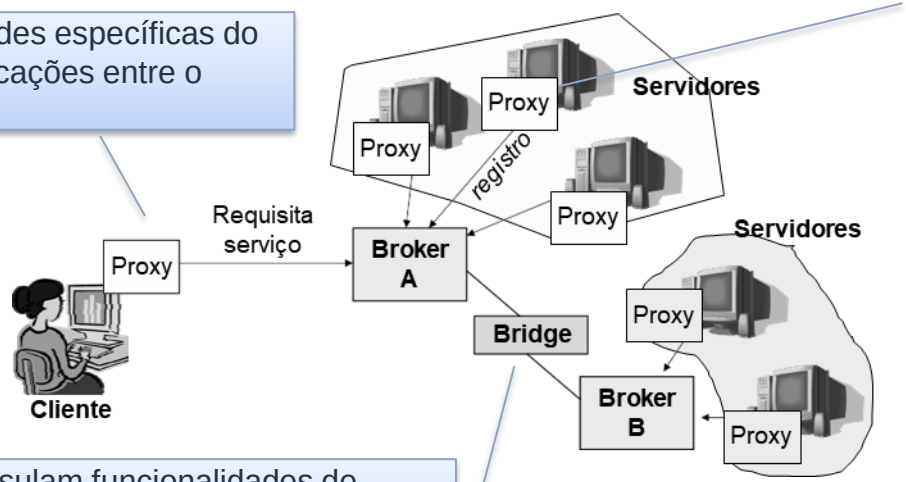
Encapsulam funcionalidades de rede; Mediam comunicações entre brokers

Registro de servidores; Transferência de mensagens; Recuperação de erros; Comunicação com outros brokers; Localizar servidores

Padrão Arquitetural: *Broker*

- Estruturar sistemas distribuídos que precisam interagir através de invocação remota de serviços.

Encapsulam funcionalidades específicas do sistema; Mediam comunicações entre o cliente e o broker



Invocam os serviços do servidor; Encapsulam funcionalidades específicas do sistema; Mediam comunicações entre o servidor e o broker

Encapsulam funcionalidades de rede; Mediam comunicações entre brokers

Padrão Arquitetural: *Broker*

Exemplos

- B2B
  - Interação com fornecedores para a solicitação de serviços
- Utilização de serviços de busca
  - Google, Amazon
- Objetos em uma mesma aplicação estão distribuídos
  - Escalabilidade, tolerância a falhas, etc.

Padrão Arquitetural: *Broker*

- Pontos positivos
  - Transparência de localização dos serviços
  - Flexibilidade: Se os servidores forem trocados mas as interfaces permanecerem as mesmas, não há impacto para o resto do sistema.
  - Portabilidade
- Pontos negativos
  - Sobrecarga de processamento
  - Debug: Uma falha na execução de um serviço pode ter sido causada tanto pelo cliente quanto pelo servidor. Mais variáveis para observar.

Para pensar....

O que um arquiteto deve considerar ao selecionar um estilo/padrão arquitetural? Como os atributos de qualidade estão relacionados com esta decisão?

Respondam nos comentários do vídeo

Referências

- Bass, L., Clements, P., & Kazman, R. (2012). *Software architecture in practice*. 3a edição. Addison-Wesley Professional.
- Pressman, R. & Maxim, B. (2016) Engenharia de Software: Uma abordagem professional. 8a edção.

Obrigada