Desenvolvimento Rápido de Aplicações

# Arquitetura de Software

Profa. Joyce Miranda

- Uma historinha...
  - ▶ João é um programador...
  - ▶ João está ansioso...
  - ▶ Seu chefe pediu pra ele desenvolver um "sisteminha"...



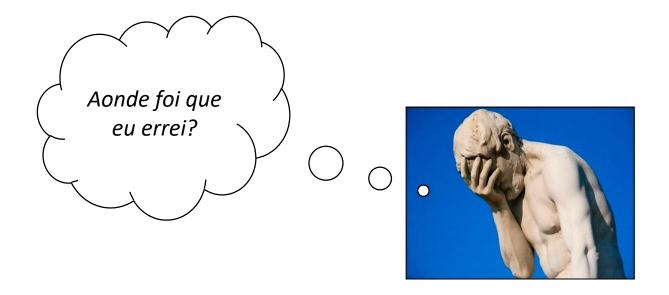
- Uma historinha...
  - No auge de sua empolgação, João consegue desenvolver o código-fonte de uma única vez, como um relâmpago.



- Uma historinha...
  - ▶ É chegada a hora da verdade: o primeiro teste!
  - ▶ Tudo que se é possível imaginar acontece, exceto funcionar.
  - João desconfia de tudo: do hardware, do compilador, do Windows...



- Uma historinha...
  - Em desespero, João sai mexendo no código aleatoriamente a fim de encontrar e resolver o problema.
  - Conclusão
    - ▶ João tentou ser <u>eficiente</u> em tempo, mas não foi <u>eficaz</u>!

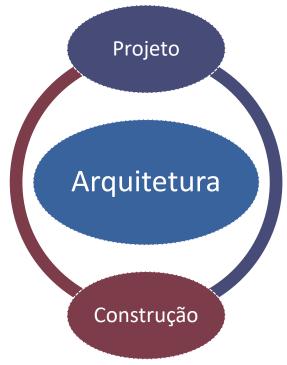


- Uma historinha...
  - Analisando a solução de João:
    - Ele não se preocupou com comentários, modularização e identação de código;
    - Para agilizar o desenvolvimento, ele tomou a sábia decisão de usar exemplos da internet;
    - ▶ E para mostrar que é muito eficiente, antes mesmo dos testes, ele modificou as rotinas e as deixou mais "rebuscadas".

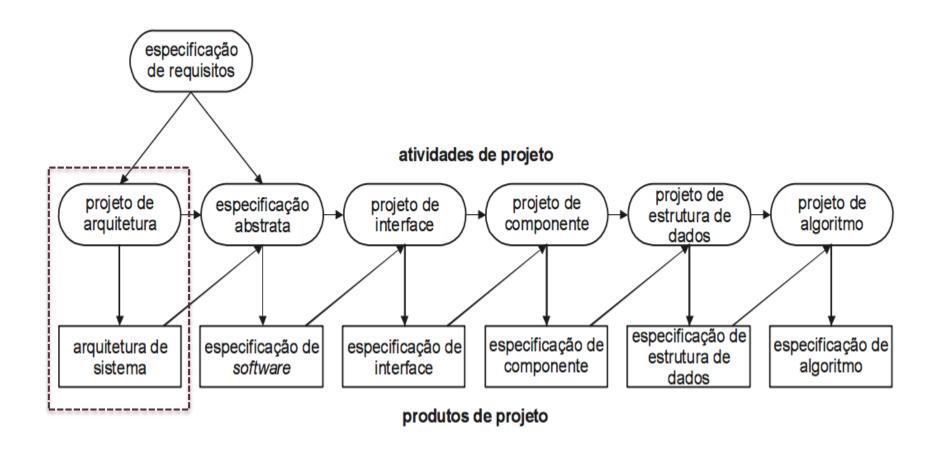


#### Escopo

Conjunto de decisões estratégicas relacionadas à estrutura e ao comportamento do software a fim de atender seus requisitos funcionais e não funcionais.



Processo do projeto de software



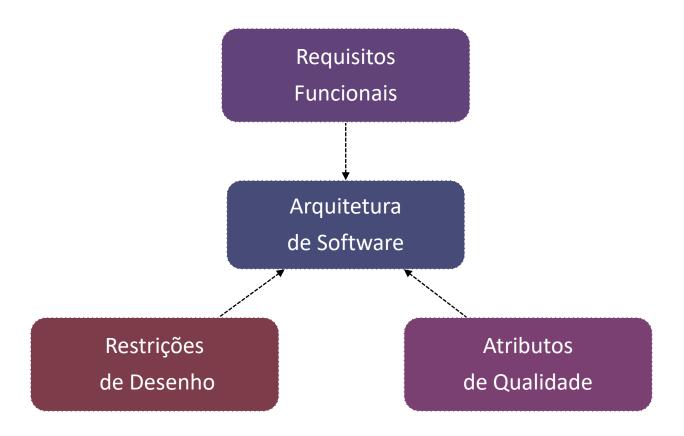
Definição formal

Organização fundamental de um sistema, expressa nos seus componentes, nos <u>relacionamentos</u> entre eles e com o ambiente, e nos <u>princípios</u> que governam seu projeto e sua evolução.

- Objetivos
  - Composição do sistema
    - Definir os elementos estruturais e suas interfaces
  - Interação
    - Estabelecer o comportamento obtido pela colaboração dos componentes sistêmicos
  - Agregação
    - Compor elementos estruturais e comportamentais em subsistemas

- Exemplo de elementos arquiteturais:
  - Servidores
  - Banco de Dados
  - Pacotes
  - Módulos
  - Classes
  - Relacionamentos
  - Bibliotecas
  - Código Fonte
  - Executáveis

Quais requisitos influenciam a arquitetura de software?



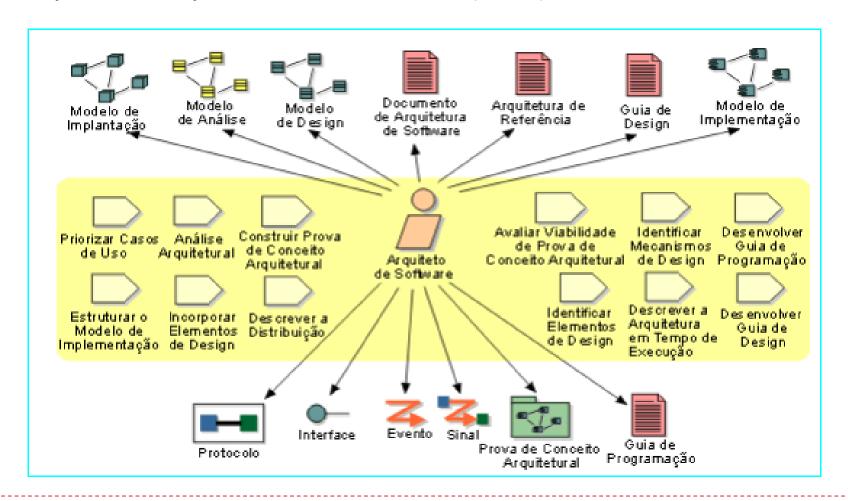
- Atributos de Qualidade (Requisitos não funcionais)
  - Desempenho
  - Confiabilidade
  - Escalonamento
  - Manutenibilidade

Restrições Arquiteturais

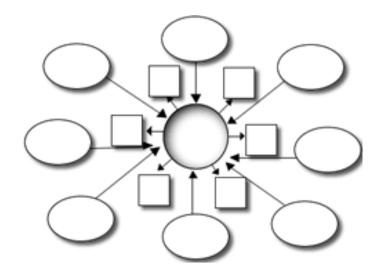
Infraestrutura Técnica Normas e padrões Tecnologias disponíveis Mercadológica Recomendações de fornecedores Financeira Orçamento disponível Políticas de licencianmento de software Legal

- Arquiteto de Software
  - Deve ter amplo conhecimento:
    - Dos requisitos da aplicação;
    - Das tecnologias disponíveis para apoiar a construção da arquitetura e do próprio software
    - Dos processos de software adequados ao desenvolvimento das aplicações.

Papel do Arquiteto de Software (RUP)



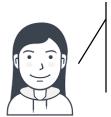
- Representação
  - Architecture Description Language (ADL)
    - Expressa características estruturais e comportamentais.
    - Pode ser um linguagem formal ou uma combinação de outros meios que permite representar diferentes pontos de vista da arquitetura.
    - Exemplos: Darwin, ACME, UML



#### Vantagens

- Apoia todo o processo de desenvolvimento
  - Gerencia a complexidade do software
  - Suporta a análise de impacto de mudanças.
  - Reduz os custos de manutenção.
  - Ajuda a verificar se o sistema suporta requisitos não funcionais importantes.
- Promove a reutilização em larga escala
  - Aproveitamento de componentes em soluções similares
- Facilita a comunicação entre *stakeholders* 
  - Ponto central de discussão entre diferentes usuários.

- Diferentes visões de arquitetura
  - ▶ Equaliza a necessidade dos *stakeholders*



Estrutura, Interações, Extensibilidade, Manutenibilidade

Funcionalidade, Desempenho, Disponibilidade, Segurança



Usuário

#### **Desenvolvedores**



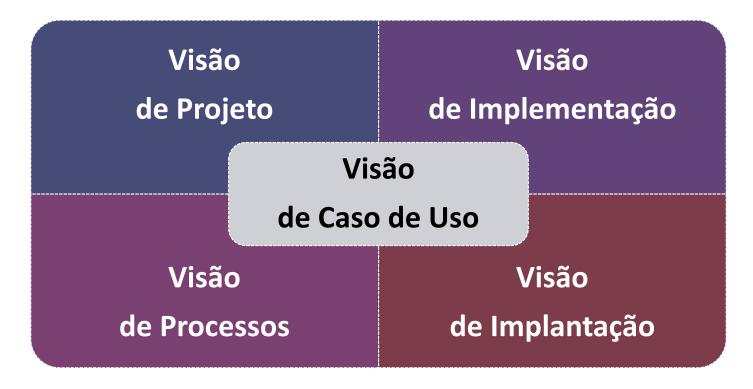
Orçamento, Prazo

Risco, Qualidade, Distribuição de Atividades

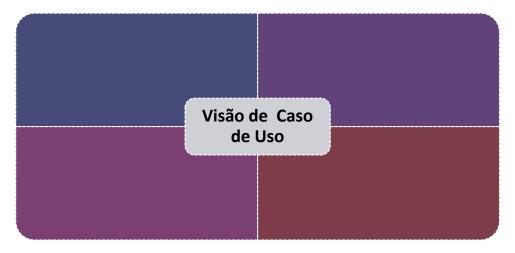


Gerente

- Visões da Arquitetura de Software
  - ▶ Modelo 4+1

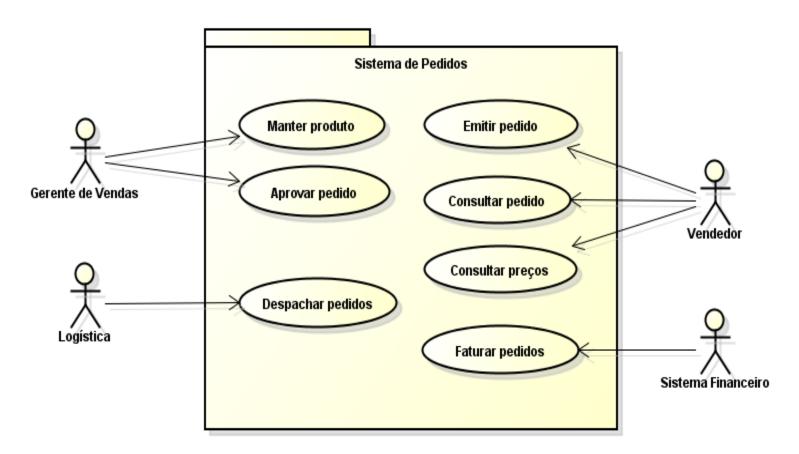


Visões da Arquitetura de Software (Modelo 4+1)



- Perspectiva
  - Usuário Final
  - Comportamento externo do sistema
    - Funcionalidades

Visão de Caso de Uso - Exemplo

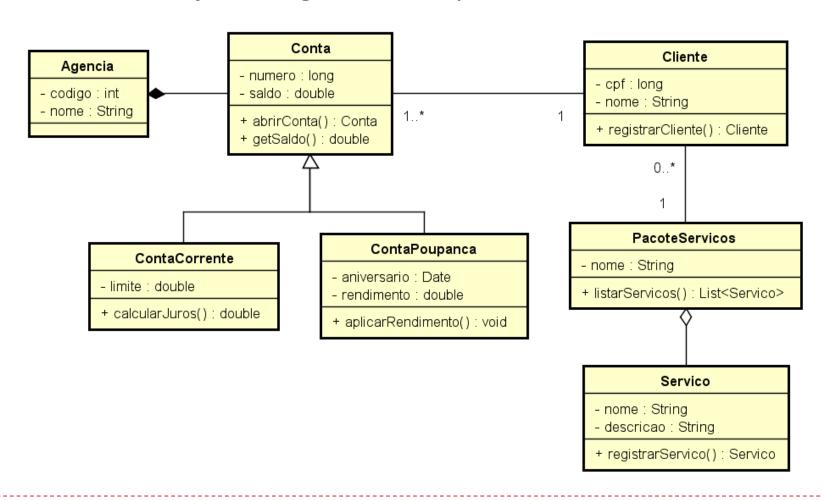


Visões da Arquitetura de Software (Modelo 4+1)

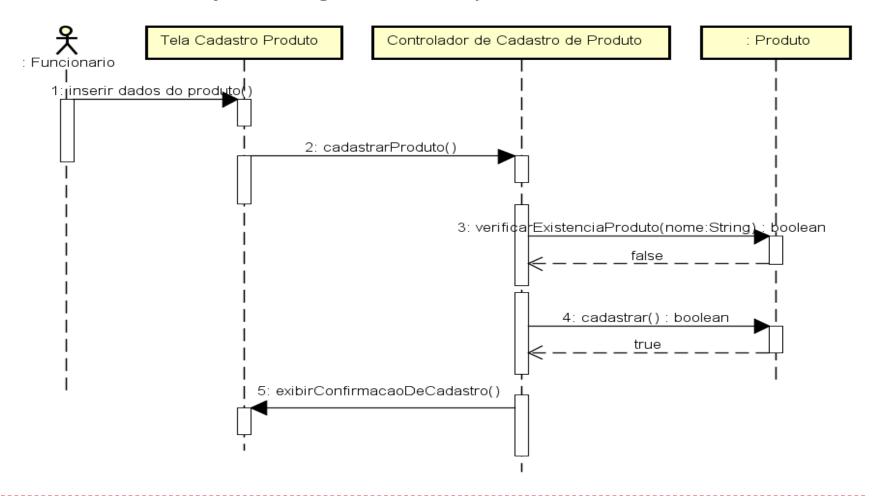


- Perspectiva (Lógica)
  - Analista e designers
  - Descreve e especifica a estrutura estática e colaborações dinâmicas do sistema
    - □ Classes, pacotes, relacionamentos.

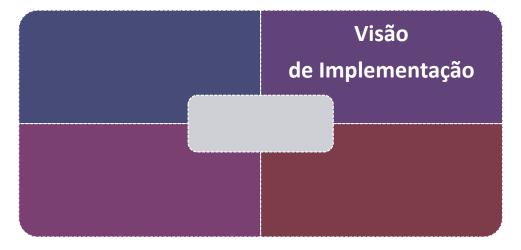
Visão de Projeto/Lógica - Exemplo



Visão de Projeto/Lógica - Exemplo

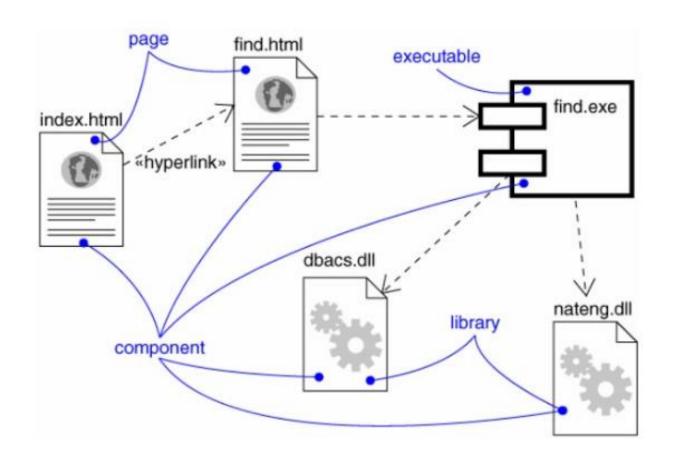


Visões da Arquitetura de Software (Modelo 4+1)

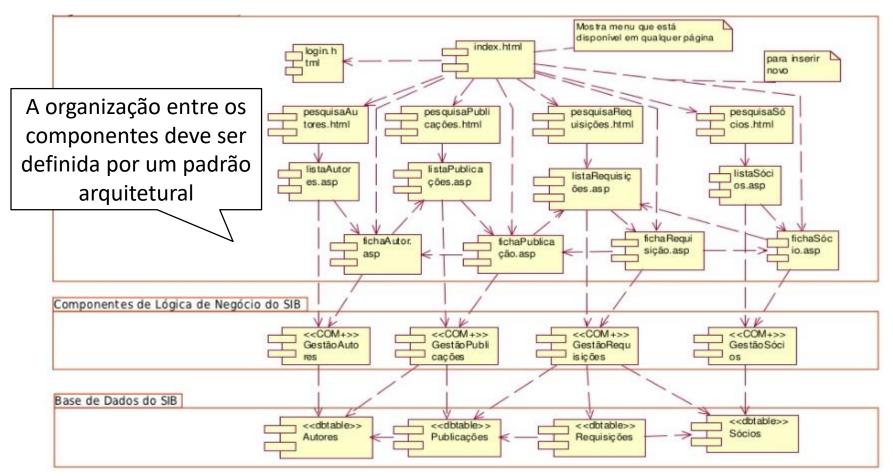


- Perspectiva (Componentes)
  - Programadores
  - Descreve e especifica artefatos relacionados ao código da aplicação
    - □ Componentes, módulos, camadas e suas dependências
    - □ Componentes: executáveis, bibliotecas, banco de dados.

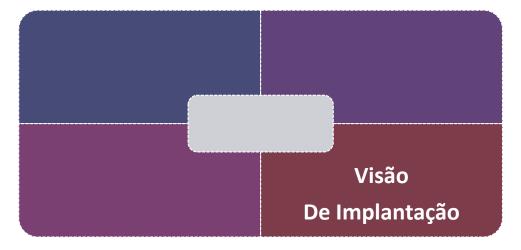
Visão de Implementação/Componentes - Exemplo



Visão de Implementação/Componentes - Exemplo

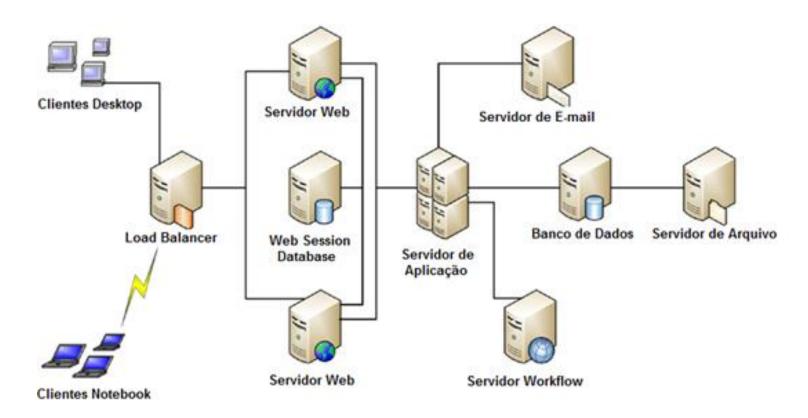


Visões da Arquitetura de Software (Modelo 4+1)

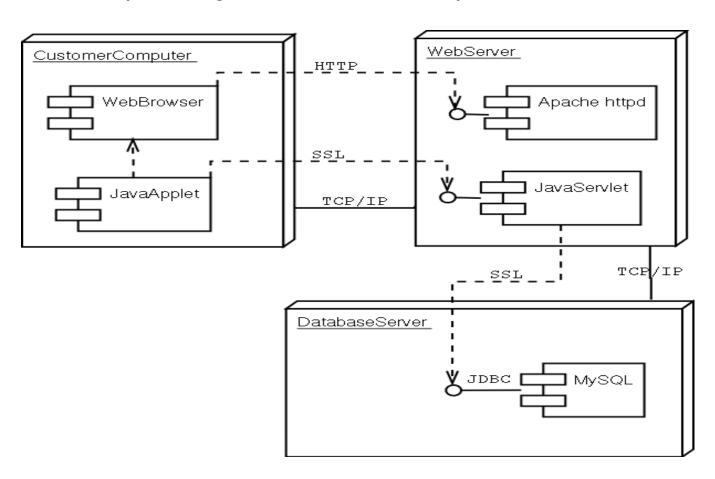


- Perspectiva (Física)
  - Engenharia de Sistema/Analistas de Suporte
  - Define a estrutura física do sistema
    - □ Topologia de hardware (computadores e periféricos)
    - □ Interação hardware/software
    - □ Critérios para liberação, instalação e comunicação.

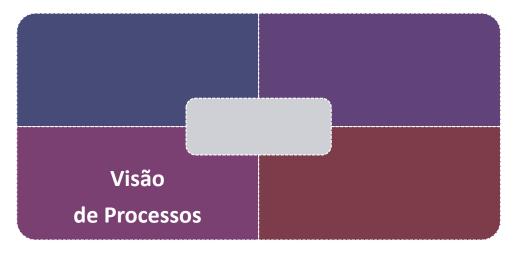
Visão de Implantação/Física - Exemplo



Visão de Implantação/Física - Exemplo

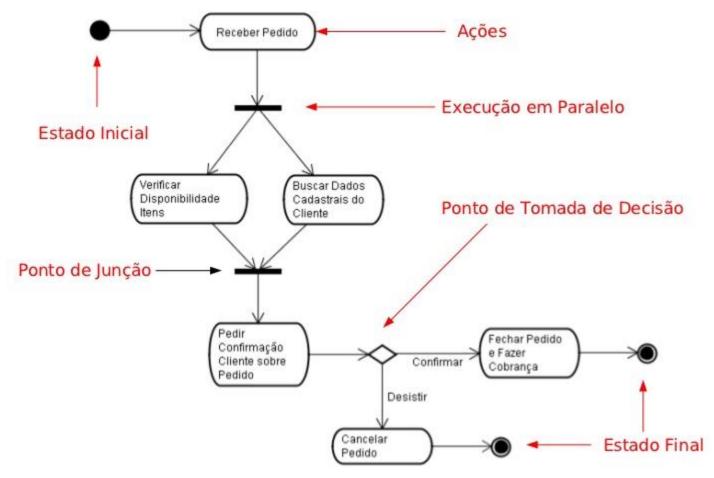


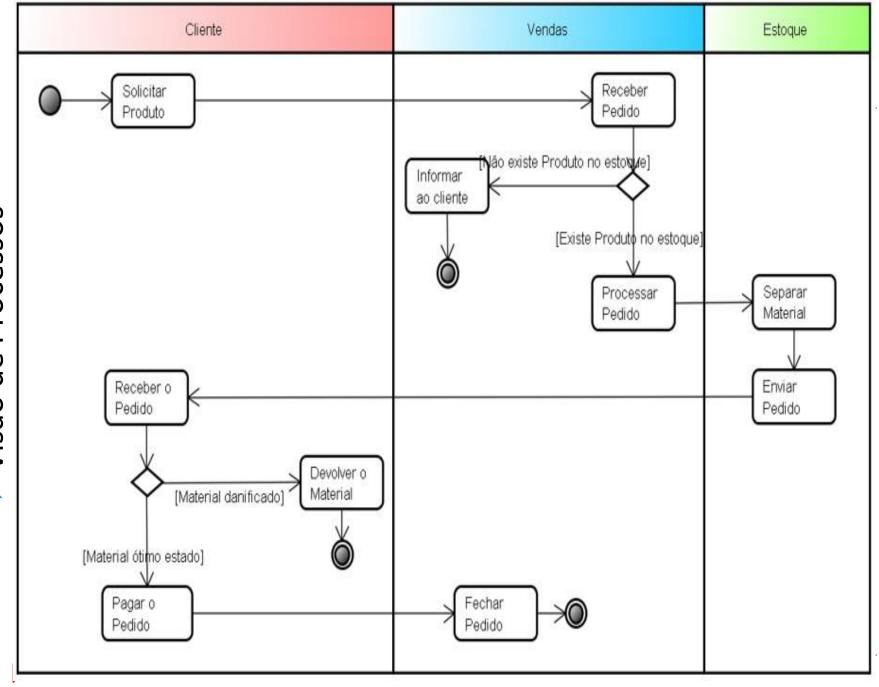
Visões da Arquitetura de Software (Modelo 4+1)

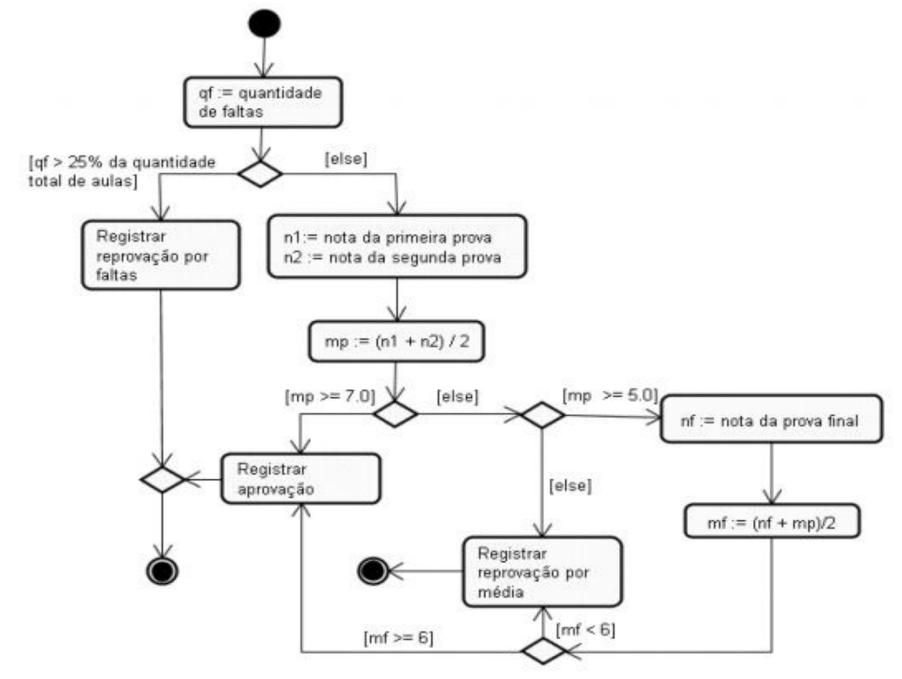


- Perspectiva
  - Integradores de Sistema/Testadores
  - Considera questões de desempenho, escalabilidade e processamento.
    - □ Descrever o fluxo de atividades em um processo
    - □ Descreve aspectos simultâneos do sistema.
      - □ Processos concorrentes (threads) devem ser definidos nessa visão.

Visão de Processos - Exemplo







Exemplo

## Documento de Arquitetura de Software

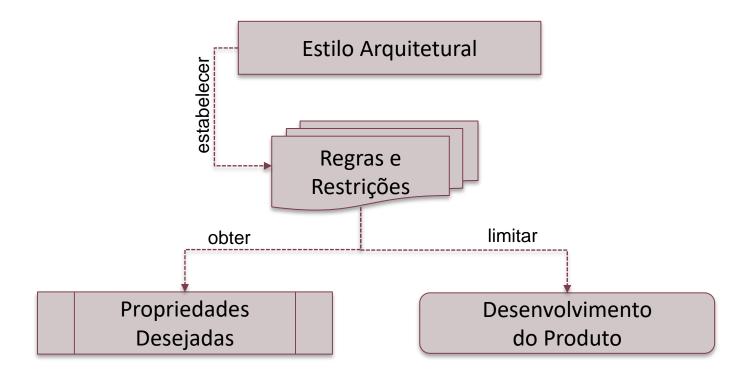
#### **▶ EXERCÍCIO**

- Com base no modelo 4+1, represente as diferentes visões de arquitetura de software para resolver o seguinte problema:
  - Contexto: Uma pizzaria deseja agilizar o atendimento realizado em sua loja física.
  - ▶ **Especificações:** Deseja-se que seus atendentes utilizem dispositivos móveis para registrar e enviar os pedidos diretamente à cozinha. Além disso, deseja-se que os atendentes possam fechar a conta e imprimí-la a partir de uma impressora que deverá ser compartilhada.

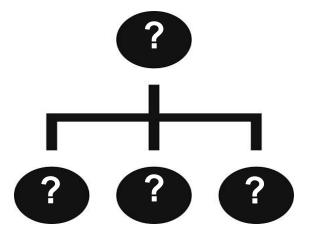


#### Padrão Arquitetural

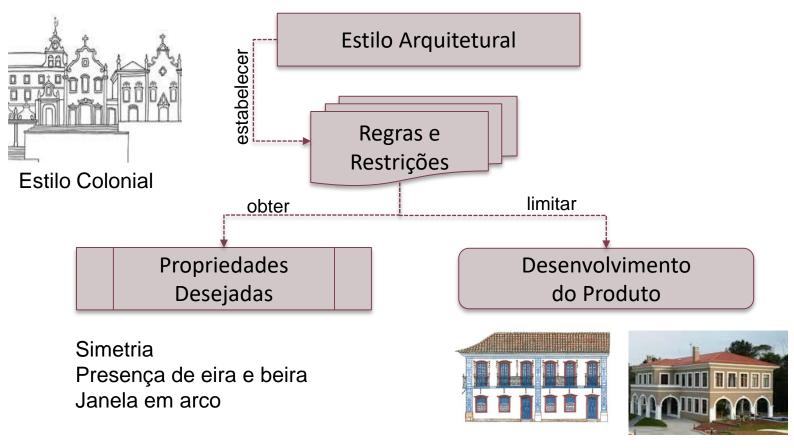
Solução estruturada de design, pronta para ser reutilizada para solucionar problemas recorrentes de arquitetura.



- Padrão Arquitetural
  - Deve ser abstrato
  - É um template que precisa ser refinado
    - Identifica a estrutura geral da organização do software
    - Define elementos, relações e regras a serem seguidas que já tiveram sua utilidade avaliada em soluções de problemas passados.

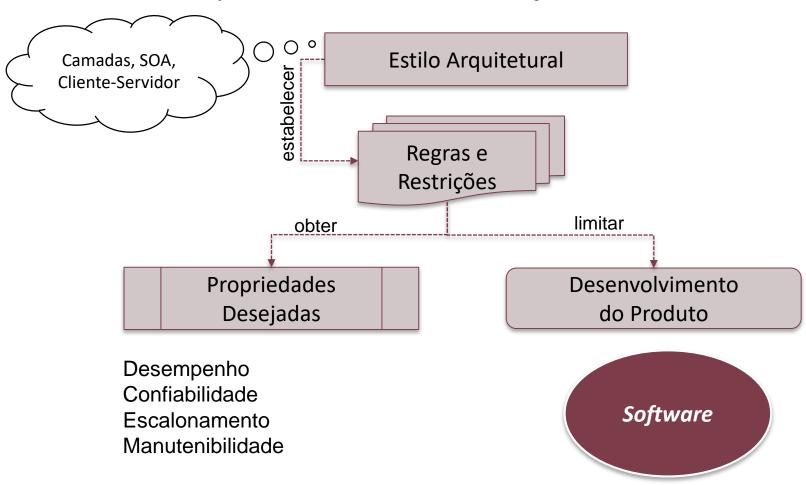


Padrão Arquitetural – Na construção civil



Arquiteturas distintas, mesmo estilo

▶ Padrão Arquitetural – Na construção de software



#### Padrão Arquitetural

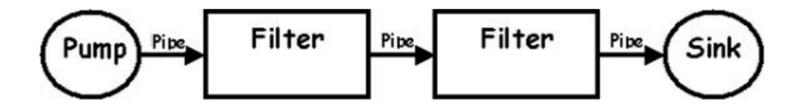
 Cada padrão propõe uma maneira de organizar o sistema e define características que indicam quando devemos utilizá-lo.



Os estilos <u>não</u> são mutuamente exclusivos.

#### Pipes & Filters

- Contexto
  - Sequência de transformações sobre uma fonte de dados.
  - Divisão de uma tarefa de processamento em uma sequência de passos (Filters) que são conectados por canais (Pipes).
- Características
  - Arquitetura linear
  - ▶ *Filter*: executa transformações sobre os dados de entrada.
  - ▶ *Pipe*: conector que passa dados de um filtro para outro.



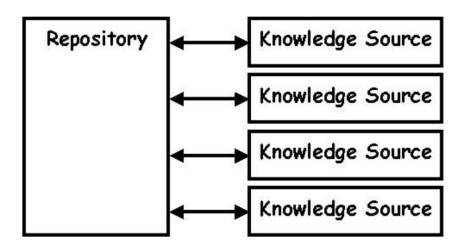
- Pipes & Filters
  - Shell do Linux
    - A saída de um programa pode ser "linkada" como a entrada de outro programa

```
$ ls | grep b | sort -r | tee arquivo.out | wc -l
```

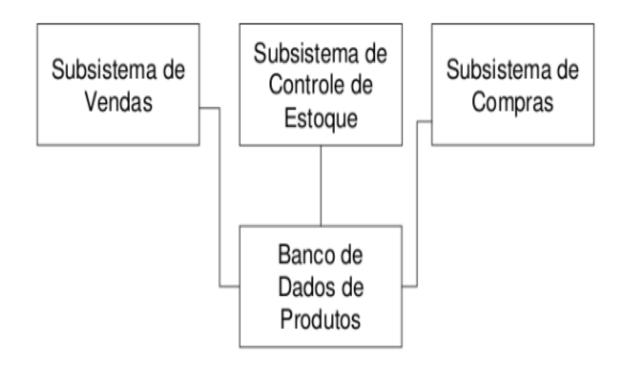
 Exemplo de componentes com independência computacional que executam uma transformação nos dados de entrada e condutores que transmitem os dados de saída de um componente a outro.

#### Repository

- Contexto
  - Útil quando subsistemas compartilham um mesmo repositório de dados.
- Características
  - Todos os subsistemas podem ler e escrever no repositório
  - A formas de acesso e a sincronização das interações são definidas pelo repositório



- Repository
  - Ex: Sistemas Gerenciadores de Banco de Dados (SGBD)



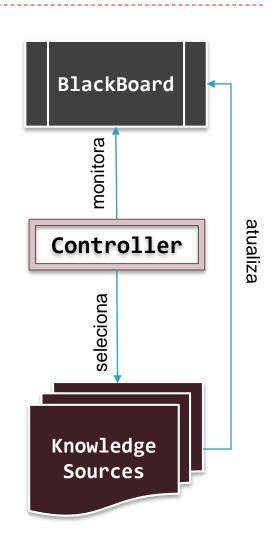
#### BackBoard

#### Contexto

- Solução para problemas não determinísticos.
- Problemas que abrangem diferentes domínios de conhecimento.
- Subsistemas reúnem seus conhecimentos para alcançar uma solução aproximada.

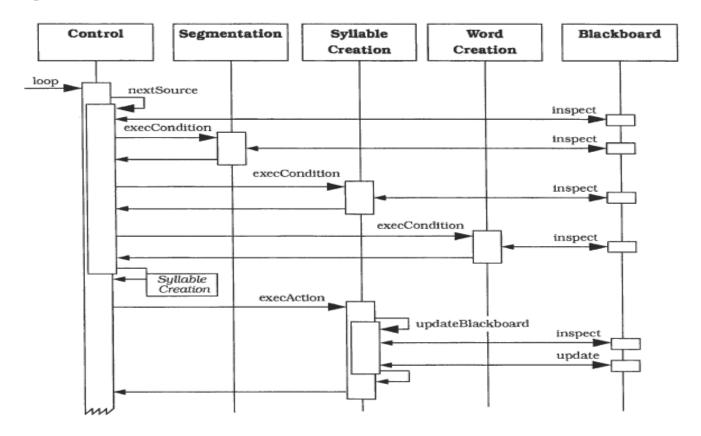
#### Características

- ▶ BlackBoard: elemento central de armazenamento
- Knowledge Sources: subsistemas que resolvem aspectos específicos do problema
- Controller: monitora mudanças e decide qual ação executar em seguida

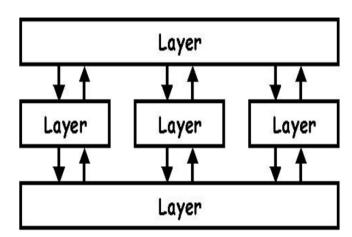


#### BlackBoard

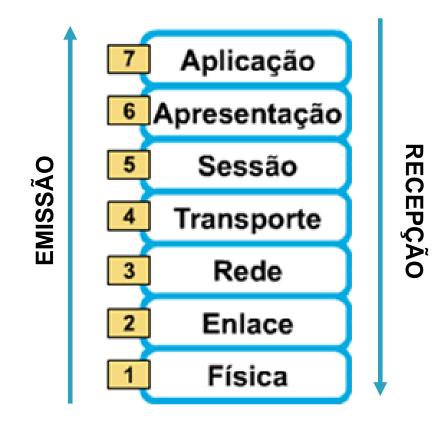
Programa de reconhecimento de voz.



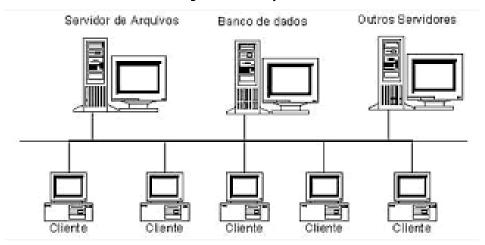
- Layers (Camadas)
  - Contexto
    - Decomposição do sistema em camadas, com alto grau de abstração e baixa dependência entre as camadas.
  - Características
    - Cada camada provê um conjunto de funcionalidades bem específicas.
      - ☐ Fornece serviços para a camada superior
      - □ Solicita serviços da camada inferior.
    - Comunicação apenas entre camadas vizinhas



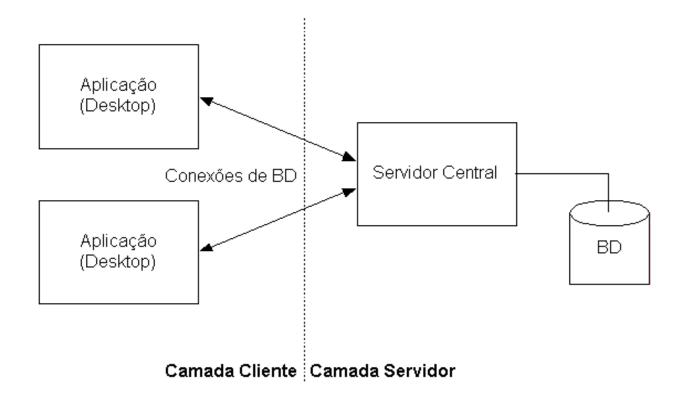
- Layers (Camadas)
  - Ex: Modelo DE Referência OSI



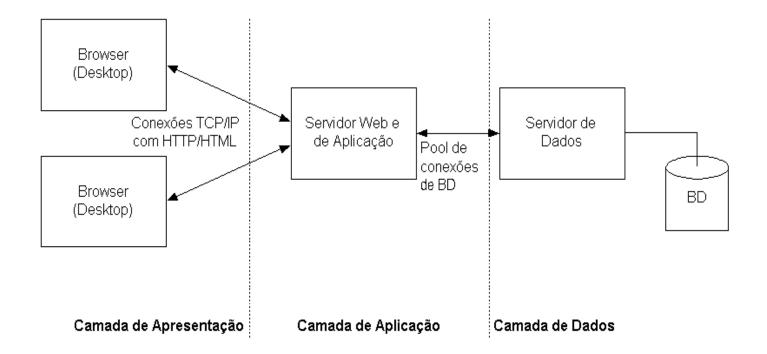
- Cliente-Servidor (2 Camadas)
  - Contexto
    - Sistemas distribuídos com componentes pouco acoplados que seguem o modelo de comunicação requisição/resposta.
  - Características
    - ▶ Um cliente faz um pedido ao servidor e espera pela resposta.
    - ▶ O servidor executa o serviço e responde ao cliente.



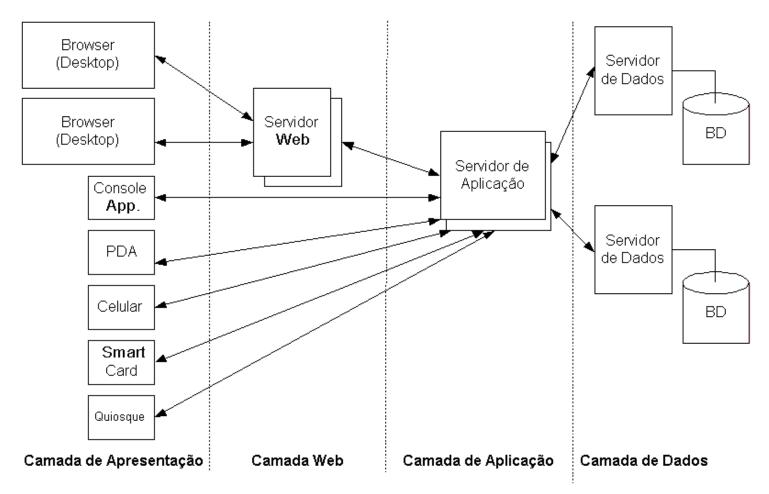
- Cliente-Servidor (2 Camadas)
  - Camada cliente trata da lógica de negócio e da interface
  - Camada servidor trata dos dados



- Cliente-Servidor (3 Camadas)
  - Camada de apresentação (interface)
  - Camada de aplicação (lógica de negócio)
  - Camada de dados



Cliente-Servidor (N Camadas)



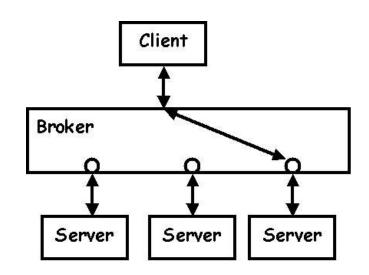
#### Broker/SOA

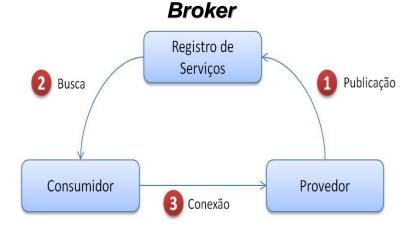
#### Contexto

- Clientes e servidores interagem por meio de um intermediador (broker)
- Arquitetura Orientada a Serviços
  - Comunicação estabelecida por meio de chamadas remotas a serviços

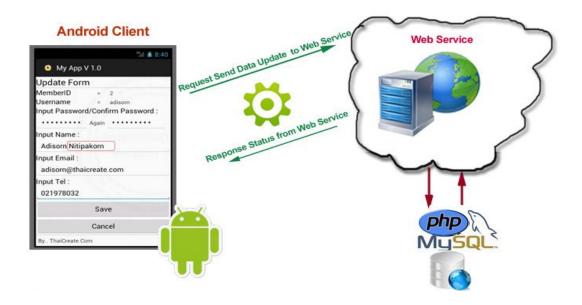
#### Características

- □ Os servidores se registram junto ao broker e tornam seus serviços disponíveis aos clientes.
- Clientes acessam a funcionalidade dos servidores enviando requisições através do broker.

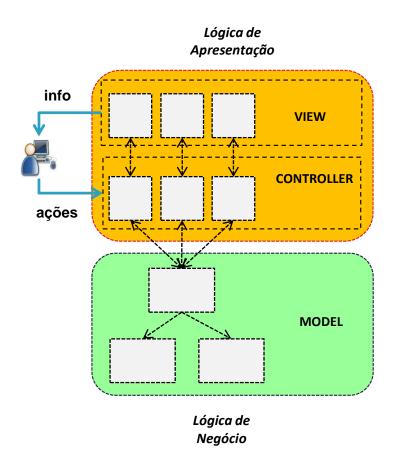




- Broker/SOA
  - Exemplo
    - WebService
      - □ Proporcionar interoperabilidade entre sistemas distribuídos, independente da plataforma e da linguagem de programação.



- MVC Model View Controller
  - Contexto
    - Sistemas interativos.
    - Separação entre <u>lógica de negócio</u> e <u>lógica de apresentação</u>.
  - Características
    - Camadas
      - □ Model
        - □ Regras de negócio e acesso a dados
      - □ View
        - Interface com o usuário
      - □ Controller
        - ☐ Intermedia a comunicação entre Model e View.



#### ► MVC – Model View Controller

