Desenvolvimento Rápido de Aplicações

Arquitetura de Software

Profa. Joyce Miranda

Arquitetura de Software

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

Arquitetura de Software

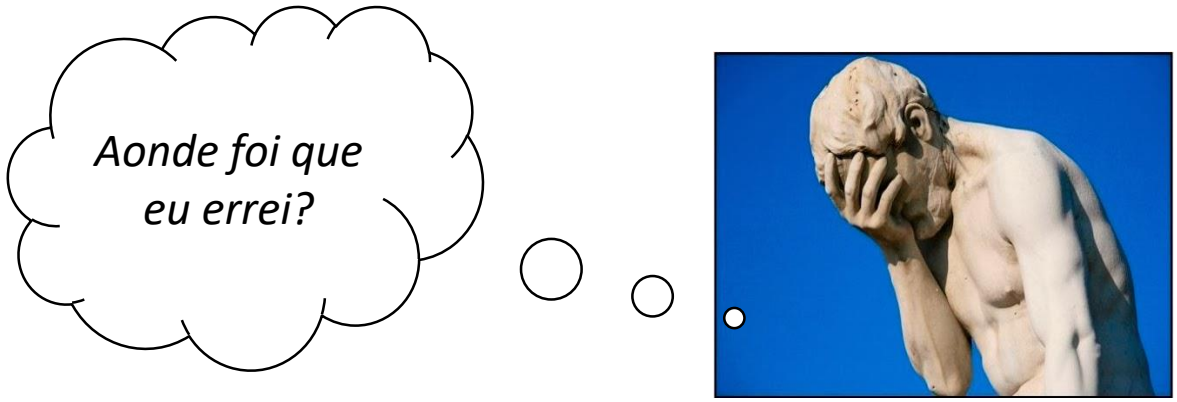
- Uma historinha...
  - No auge de sua empolgação, João consegue desenvolver o código-fonte de uma única vez, como um relâmpago.

Arquitetura de Software

- Uma historinha...
  - É chegada a hora da verdade: o primeiro teste!
  - Nessa hora tudo acontece, só o sistema que não funciona.
  - João desconfia de tudo: do hardware, do compilador, do Windows...

Arquitetura de Software

- Uma historinha...
  - Desesperado, João sai mexendo no código aleatoriamente a fim de encontrar e resolver o problema.
  - João vai gastar pelo menos duas vezes o tempo previsto para realizar a tarefa.
  - Conclusão
    - João tentou ser eficiente em tempo, mas não foi eficaz!

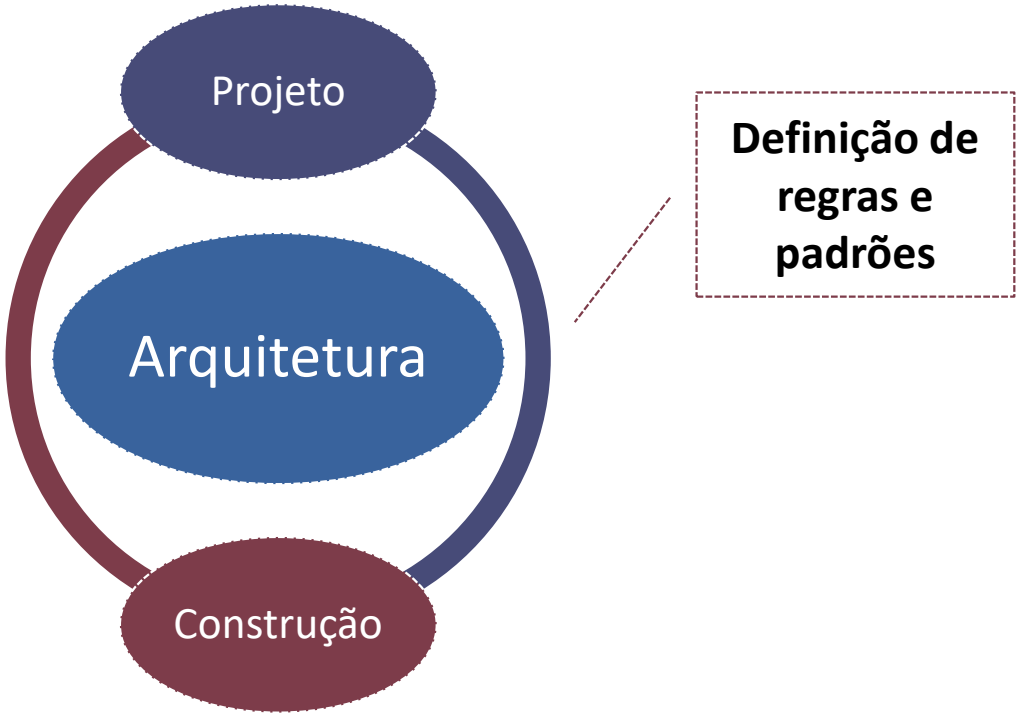


Arquitetura de Software

- 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”.

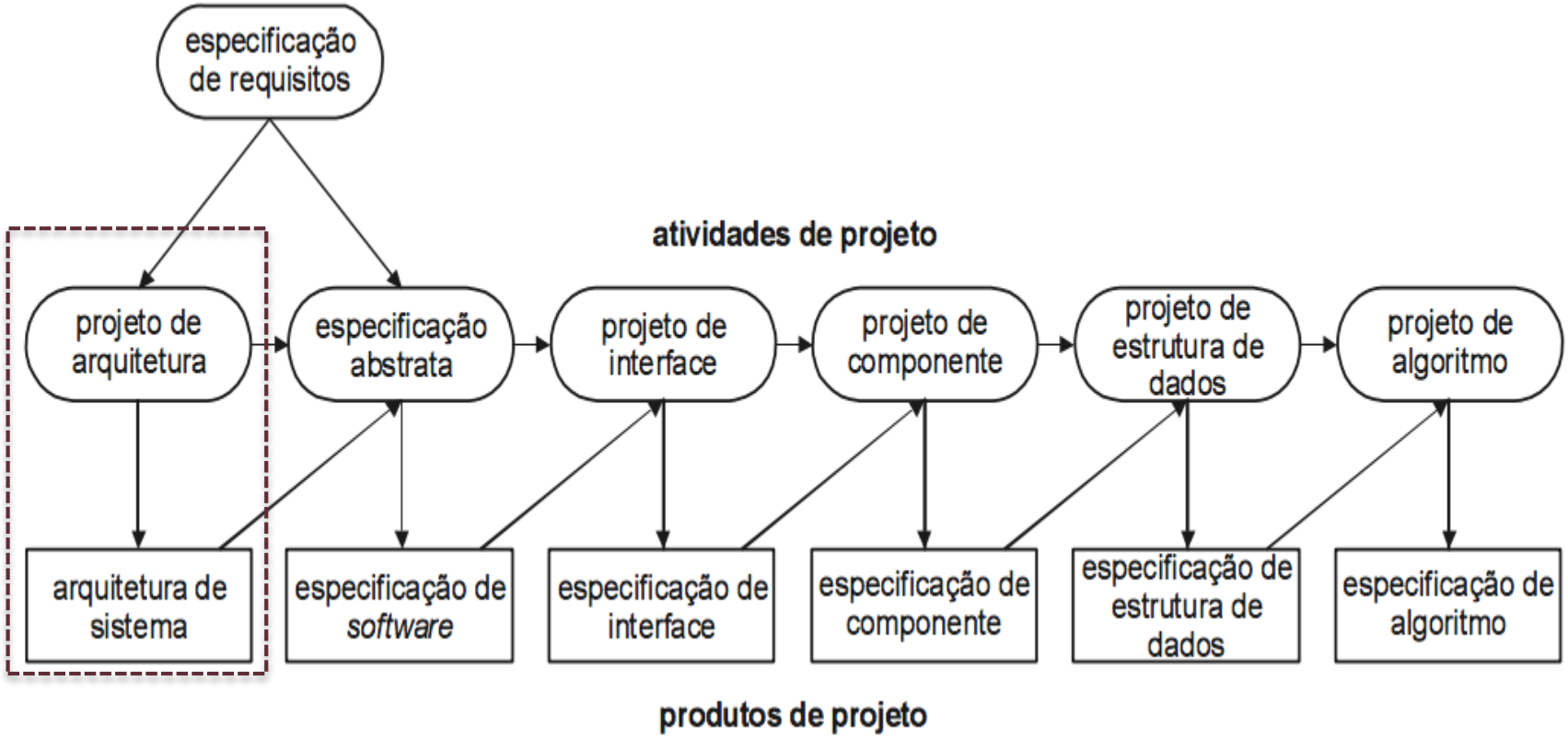
Arquitetura de Software

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



Arquitetura de Software

- A partir da arquitetura de software é possível obter um projeto detalhado.



Arquitetura de Software

- Definição formal

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

Arquitetura de Software

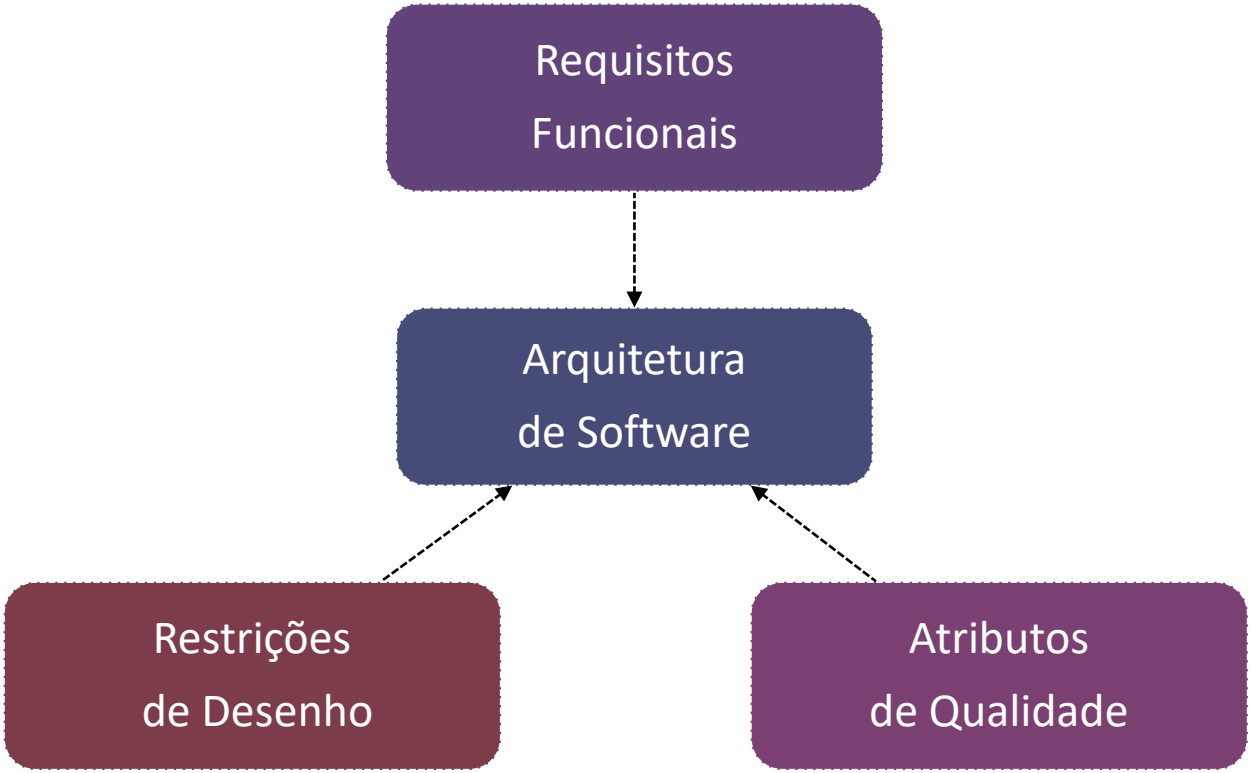
- Objetivos
  - Composição do sistema
    - Definir os elementos estruturais e suas interfaces
  - Interação
    - Estabelecer o comportamento obtido pela colaboração dos componentes sistêmicos
  - Agregação
    - Compor elementos estruturais e comportamentais em subsistemas

Arquitetura de Software

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

Arquitetura de Software

- Quais requisitos influenciam a arquitetura de software?



Arquitetura de Software

- Atributos de Qualidade (Requisitos não funcionais)
  - Desempenho
  - Confiabilidade
  - Escalonamento
  - Manutenibilidade
- A Arquitetura tem capacidade de afetar o desempenho, a robustez, a capacidade de distribuição e manutenção do sistema.

Arquitetura de Software

- Restrições Arquiteturais

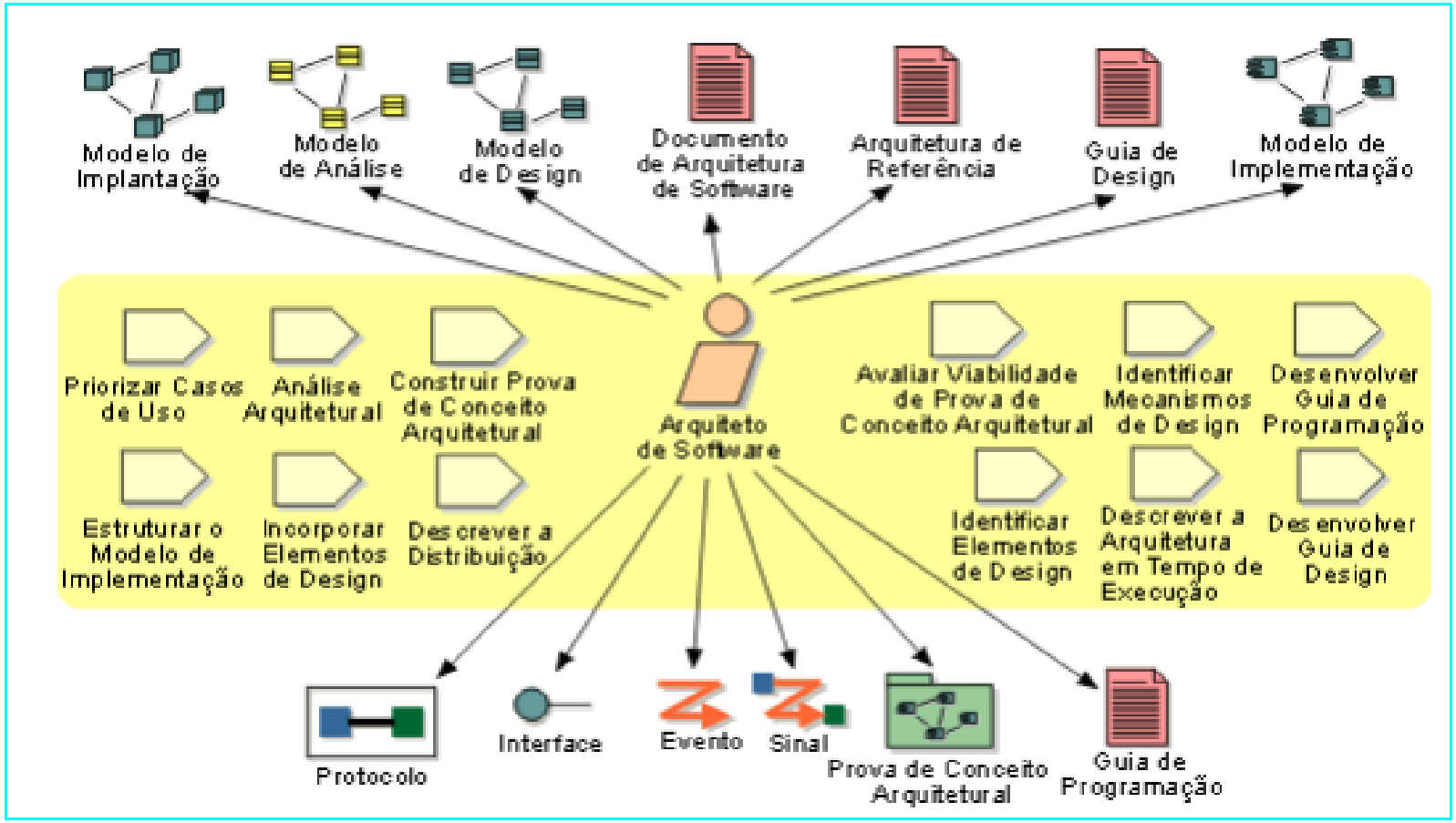
Técnica	• Infraestrutura • Normas e padrões
Mercadológica	• Tecnologias disponíveis • Recomendações de fornecedores
Financeira	• Orçamento disponível
Legal	• Políticas de licenciamento de software

Arquitetura de Software

- Arquiteto de Software
  - Elaborar o Documento de Arquitetura do 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.

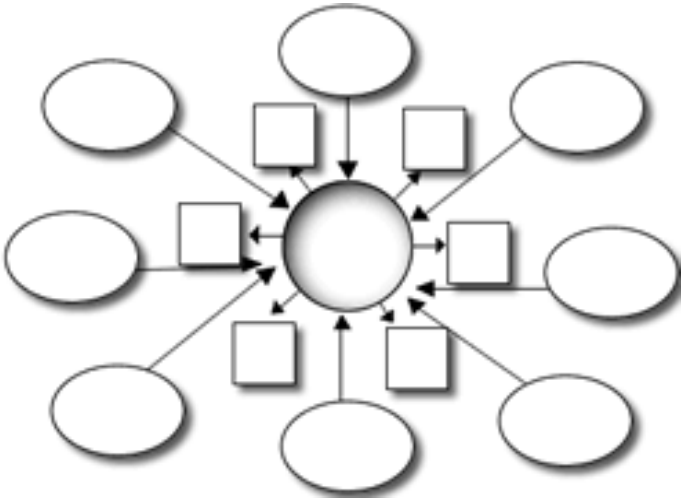
Arquitetura de Software

- Papel do Arquiteto de Software (RUP)



Arquitetura de Software

- 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

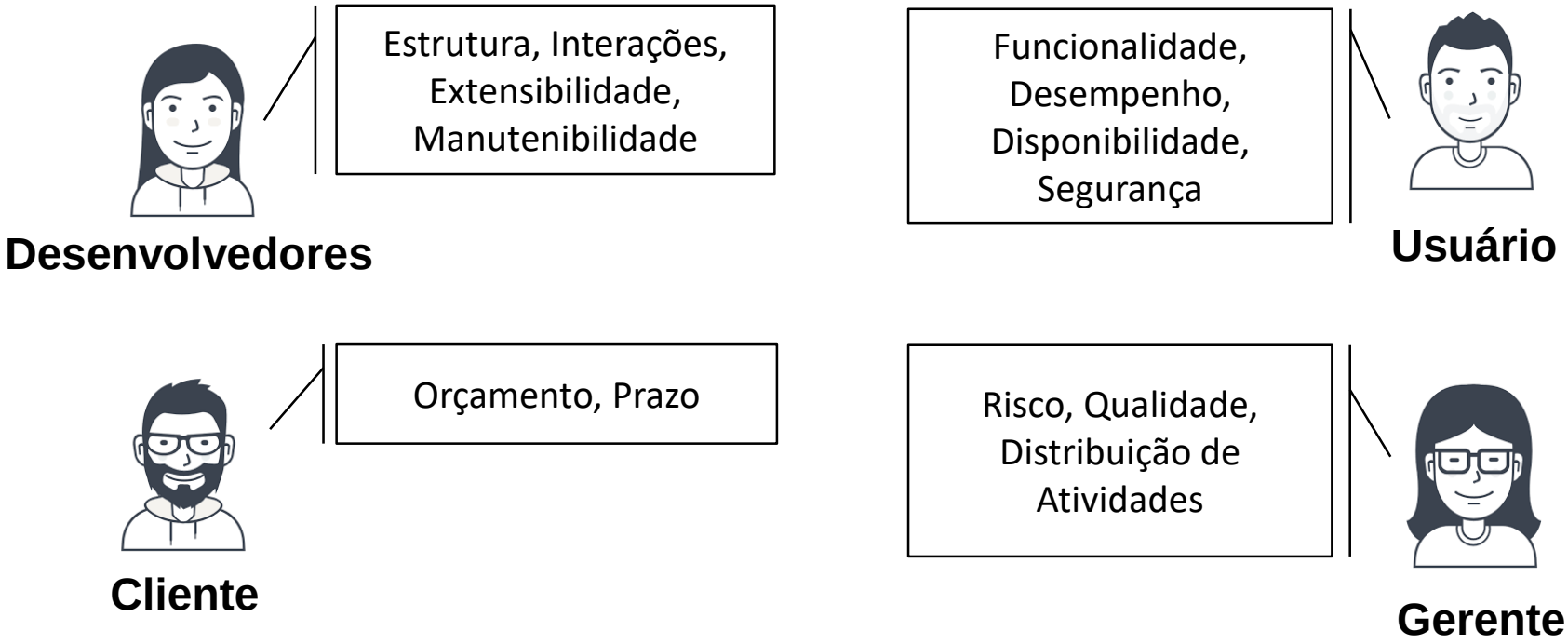


Arquitetura de Software

- Vantagens
  - Apoia todo o processo de desenvolvimento
    - 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.

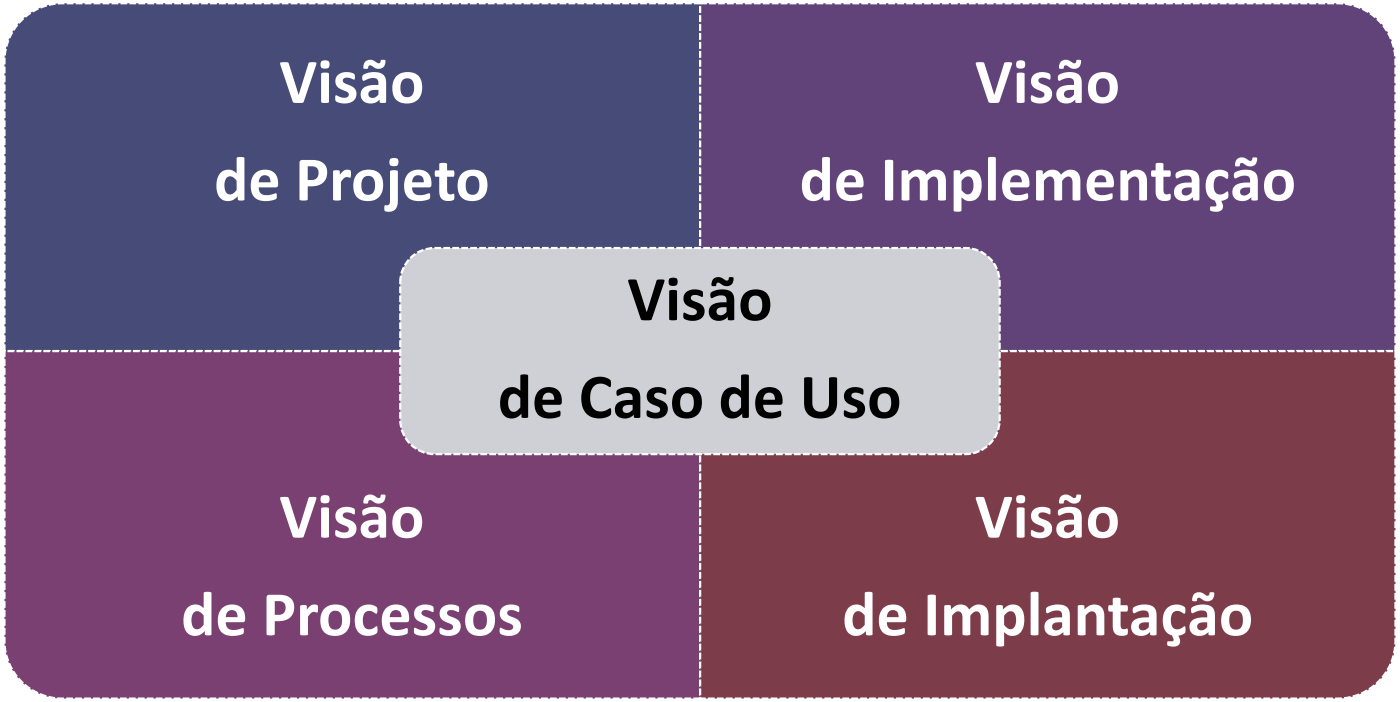
Arquitetura de Software

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



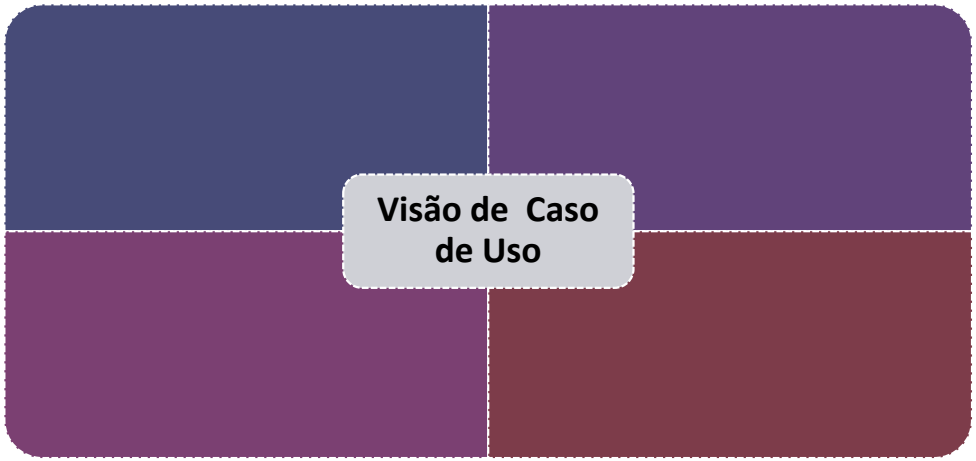
Arquitetura de Software

- Visões da Arquitetura de Software
  - Modelo 4+1



Arquitetura de Software

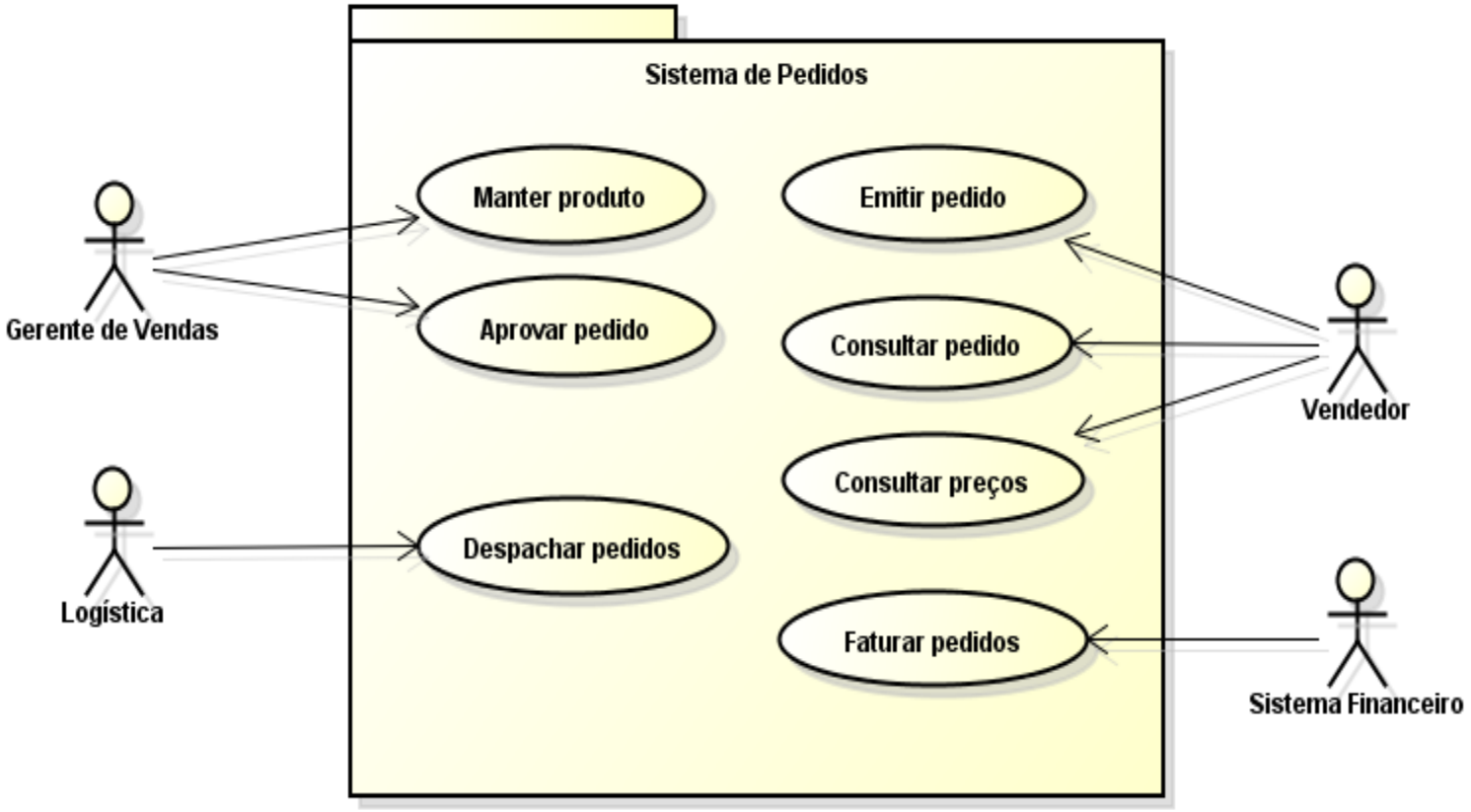
- Visões da Arquitetura de Software (Modelo 4+1)



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

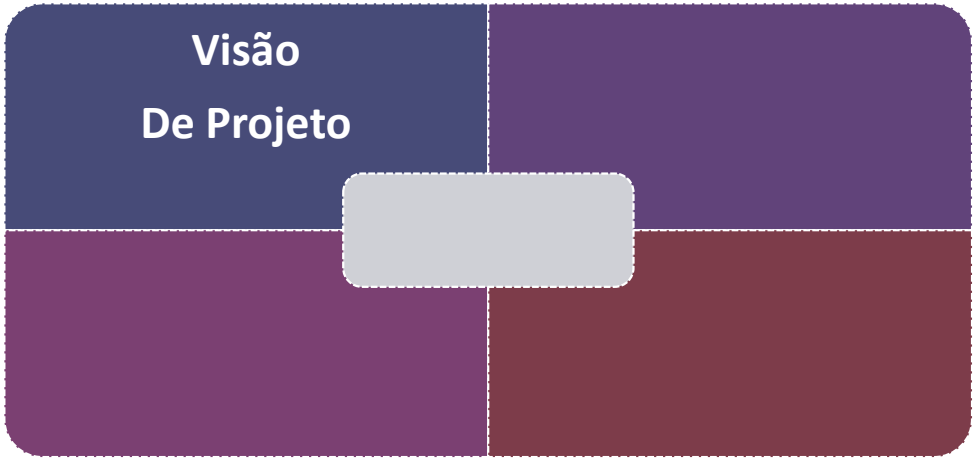
Arquitetura de Software

- Visão de Caso de Uso - Exemplo



Arquitetura de Software

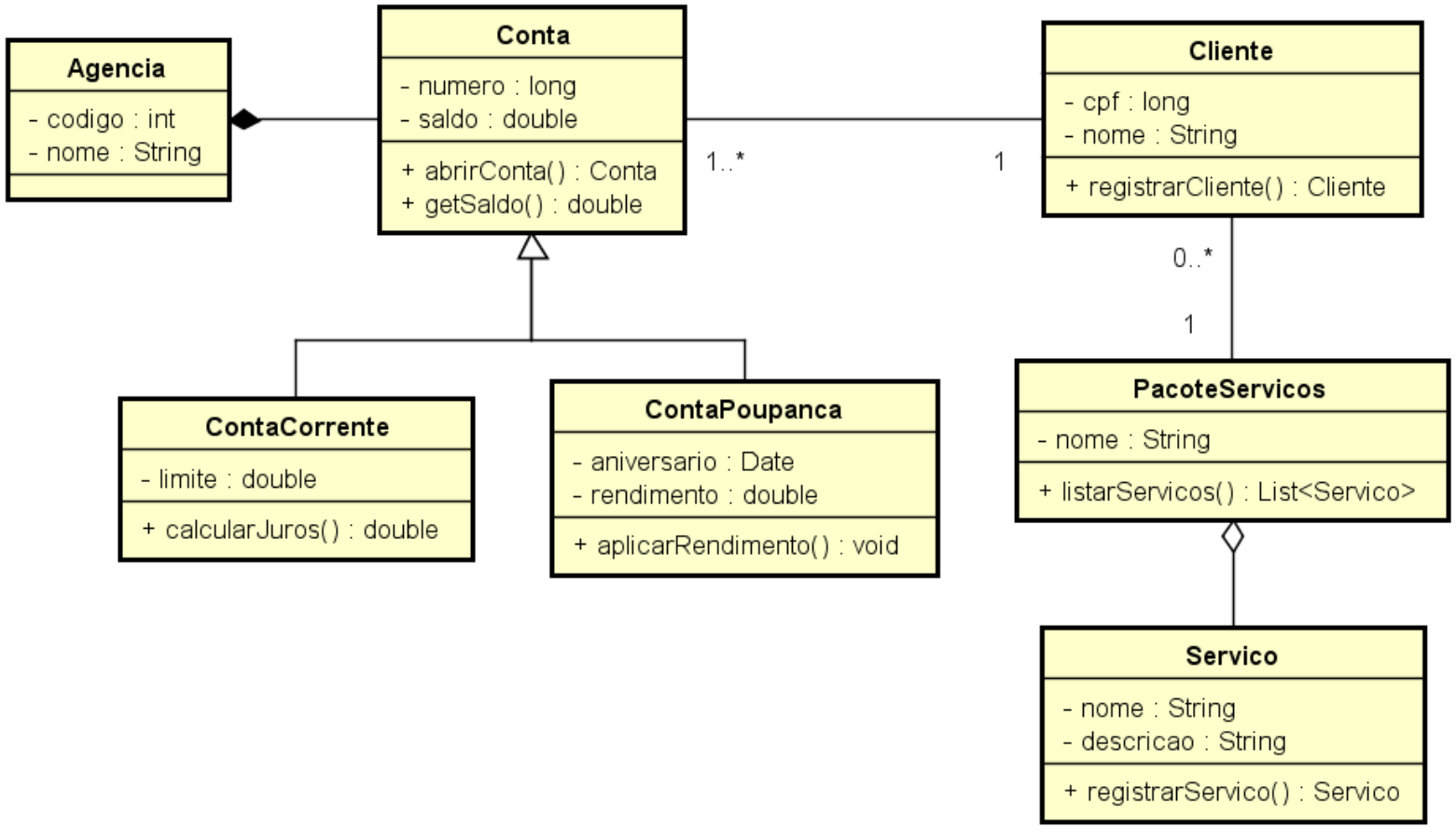
- Visões da Arquitetura de Software (Modelo 4+1)



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

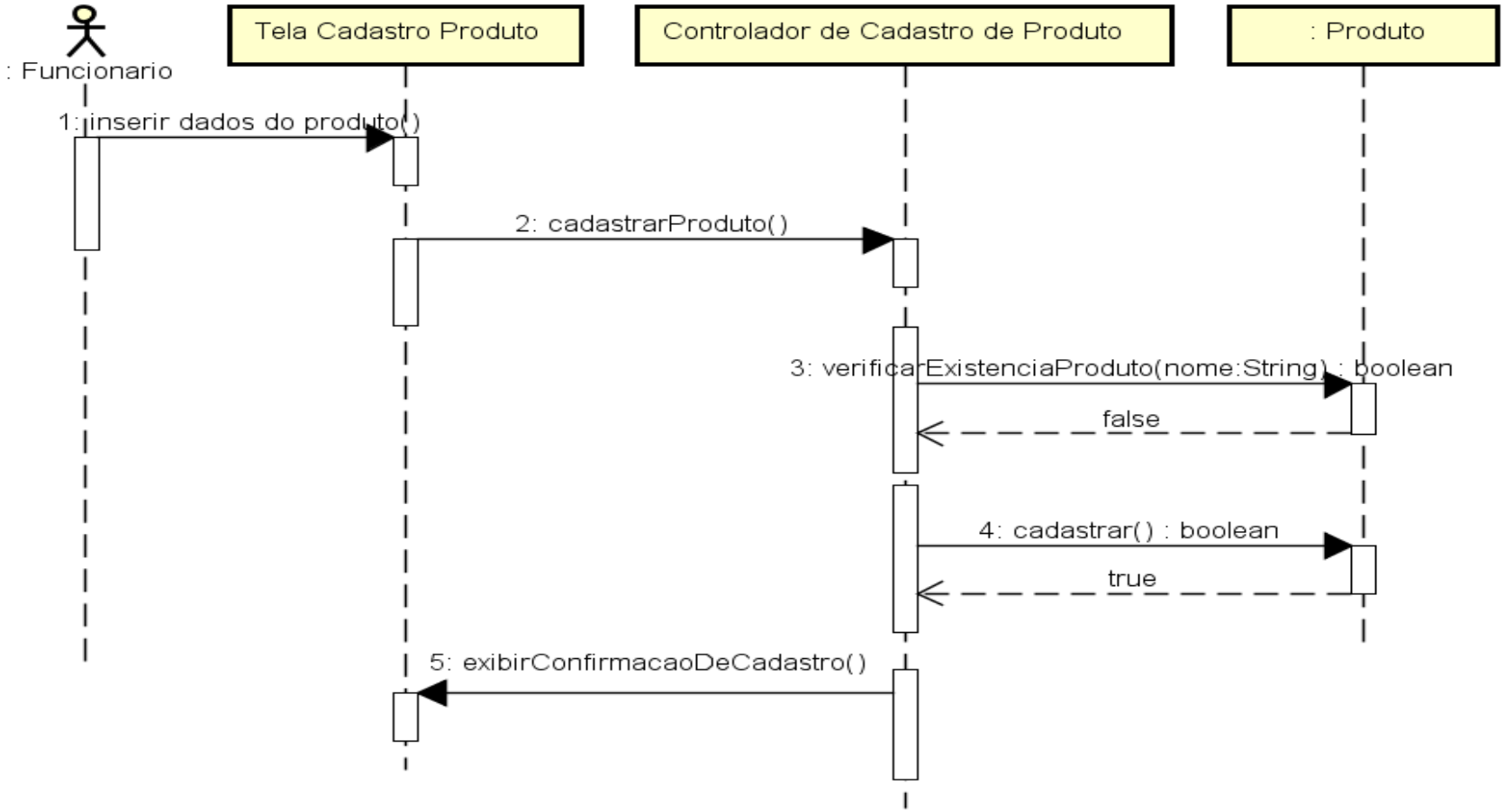
Arquitetura de Software

- Visão de Projeto/Lógica - Exemplo



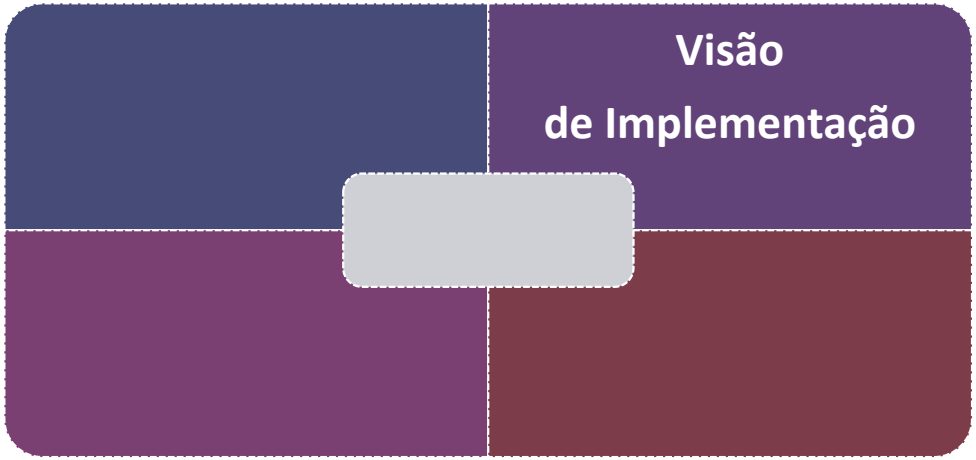
Arquitetura de Software

- Visão de Projeto/Lógica - Exemplo



Arquitetura de Software

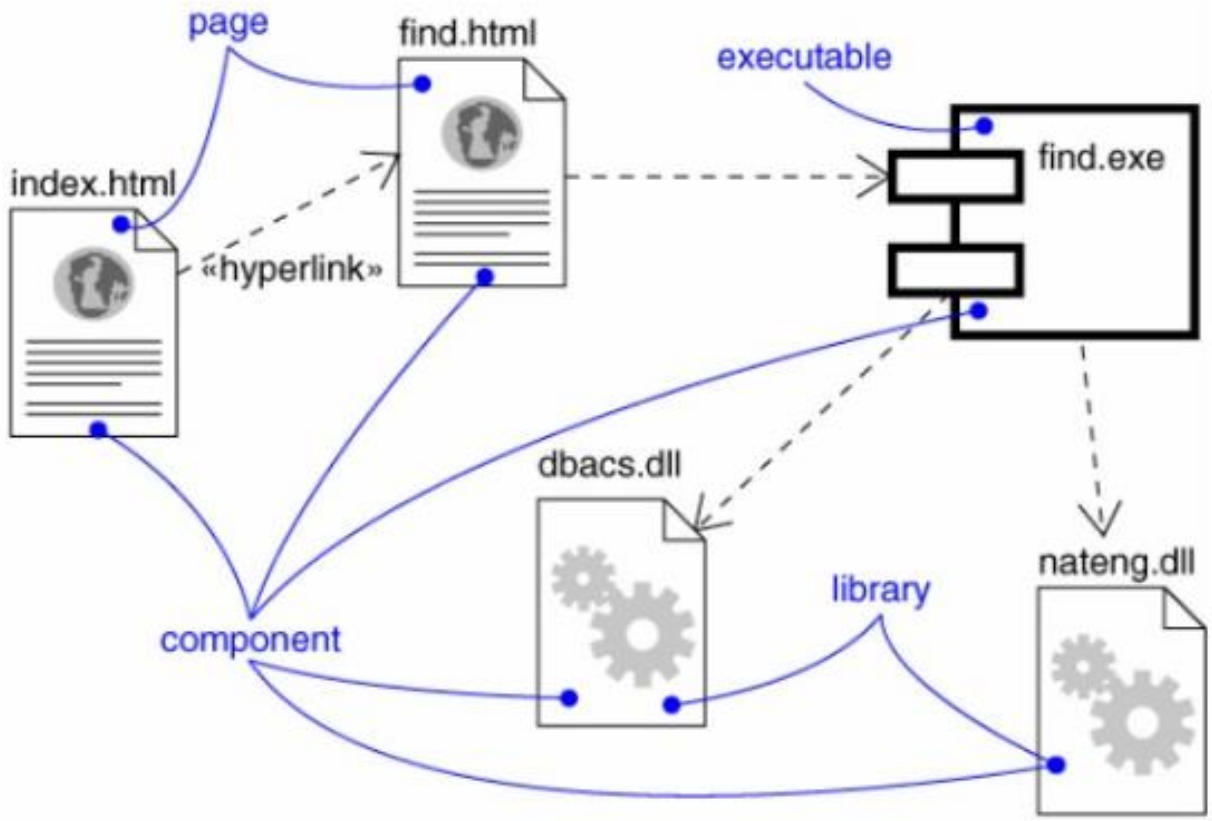
- Visões da Arquitetura de Software (Modelo 4+1)



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

Arquitetura de Software

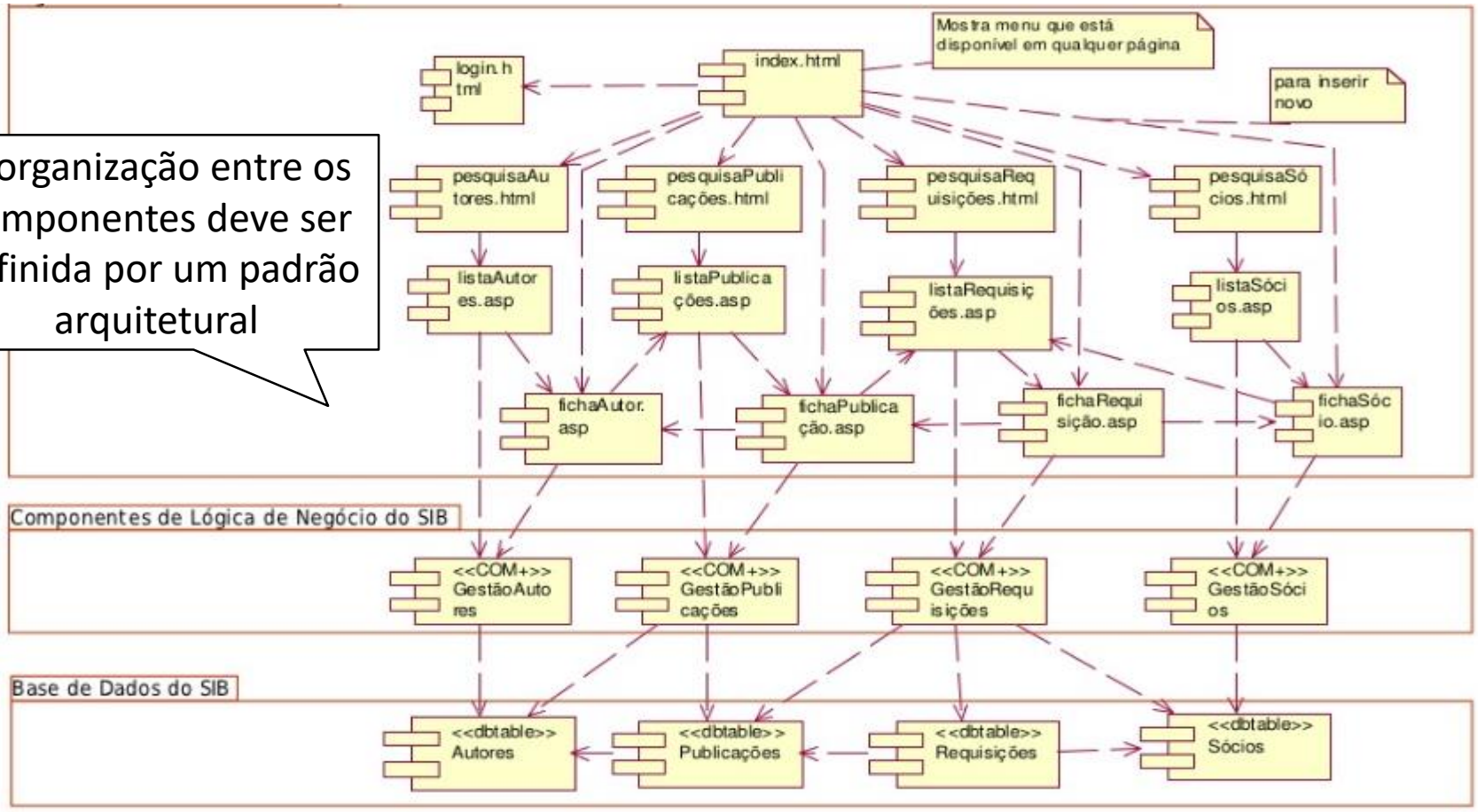
- Visão de Implementação/Componentes - Exemplo



Arquitetura de Software

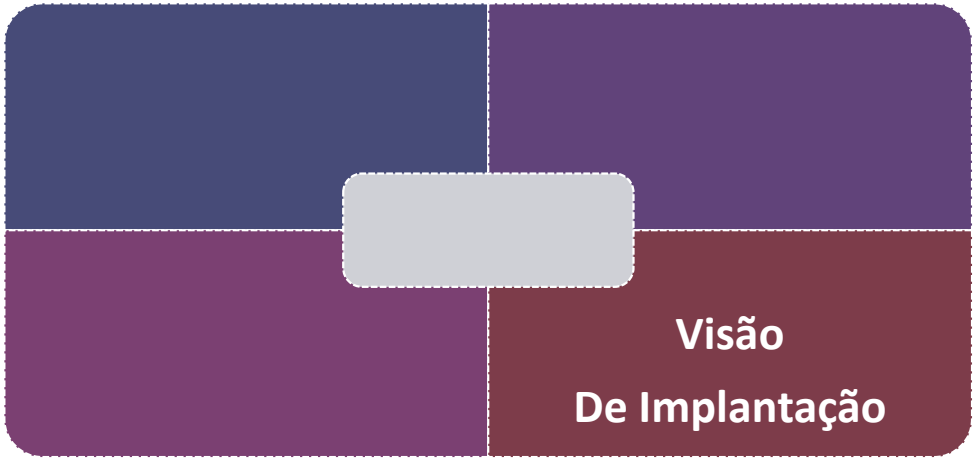
- Visão de Implementação/Componentes - Exemplo

A organização entre os componentes deve ser definida por um padrão arquitetural



Arquitetura de Software

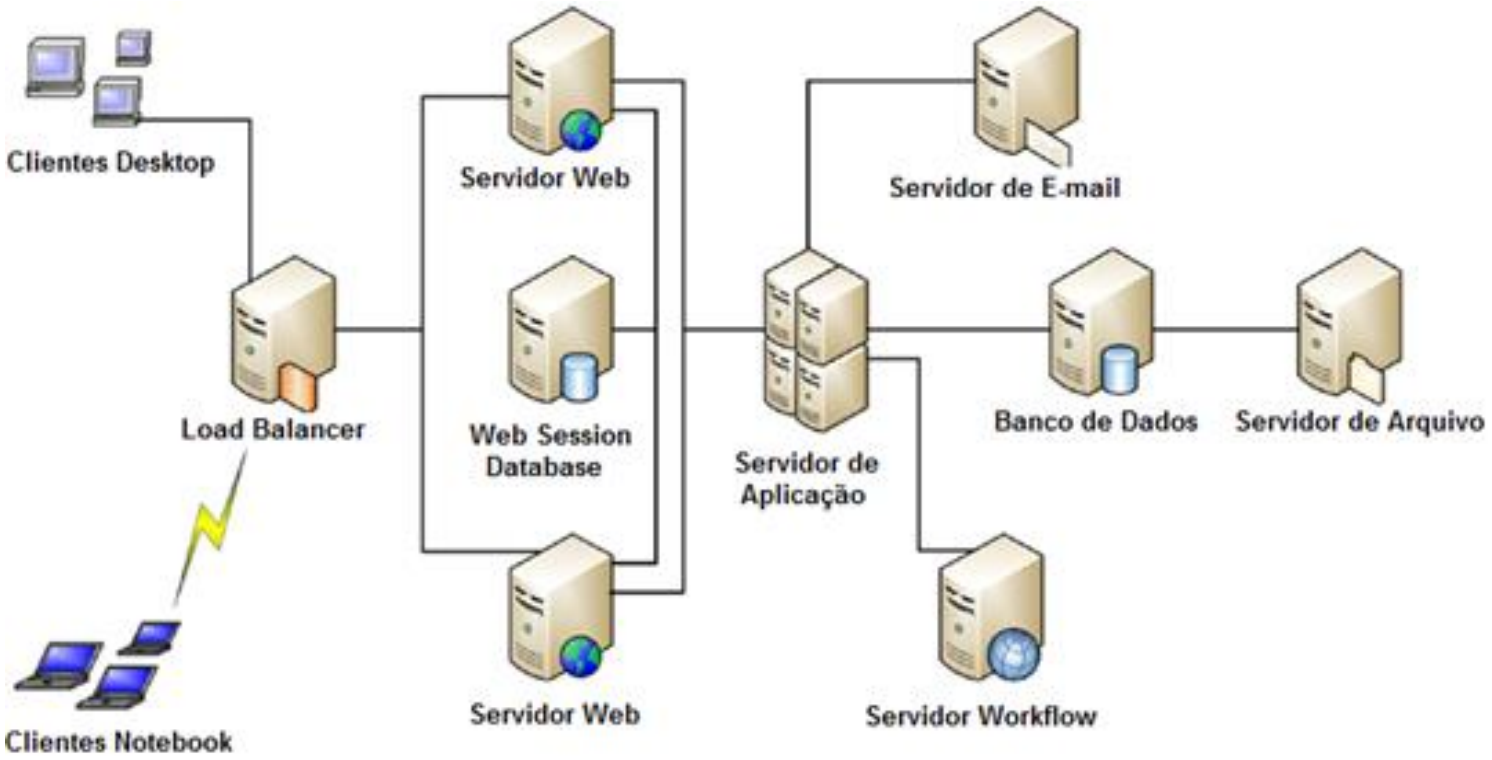
- Visões da Arquitetura de Software (Modelo 4+1)



  - 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 e instalação.

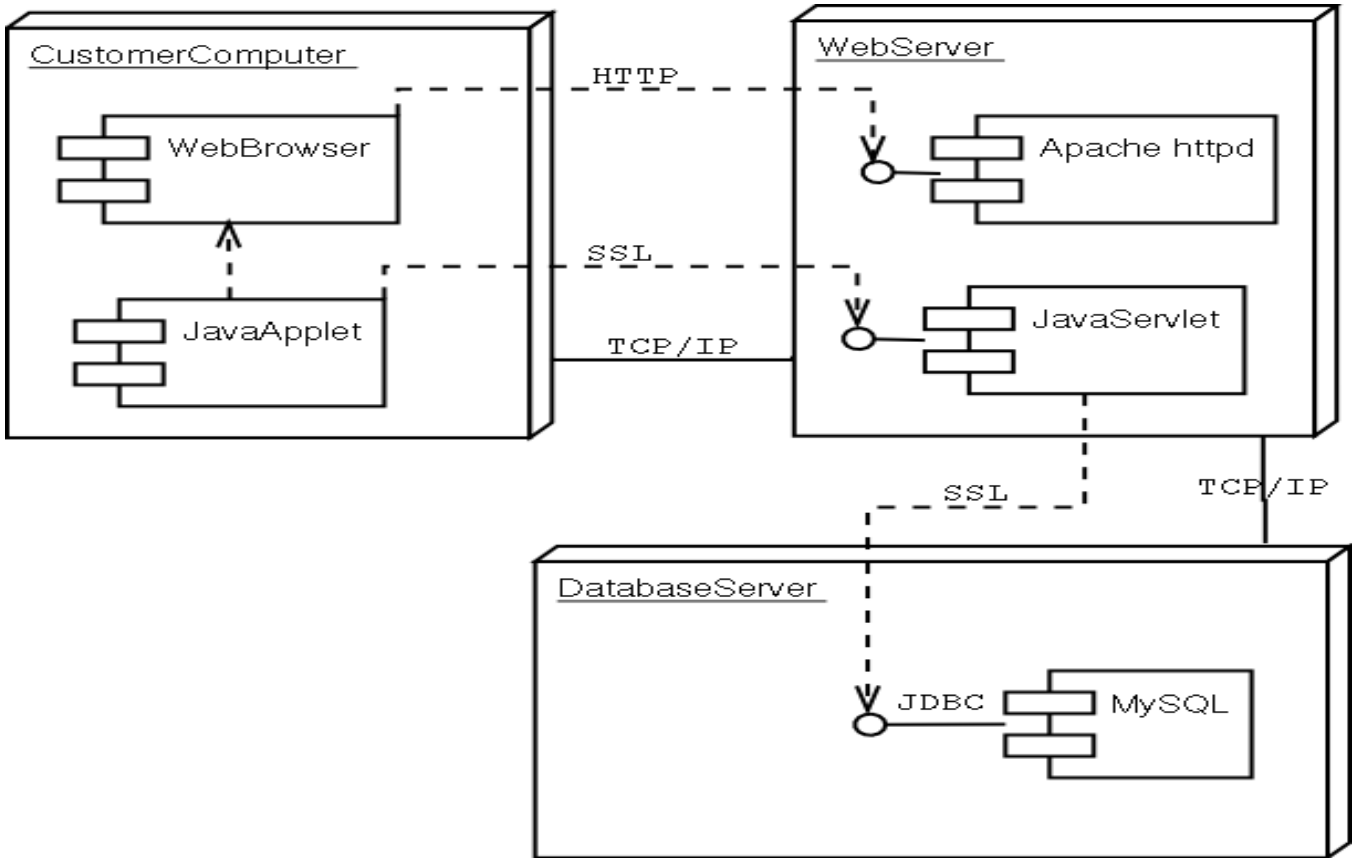
Arquitetura de Software

- Visão de Implantação/Física - Exemplo



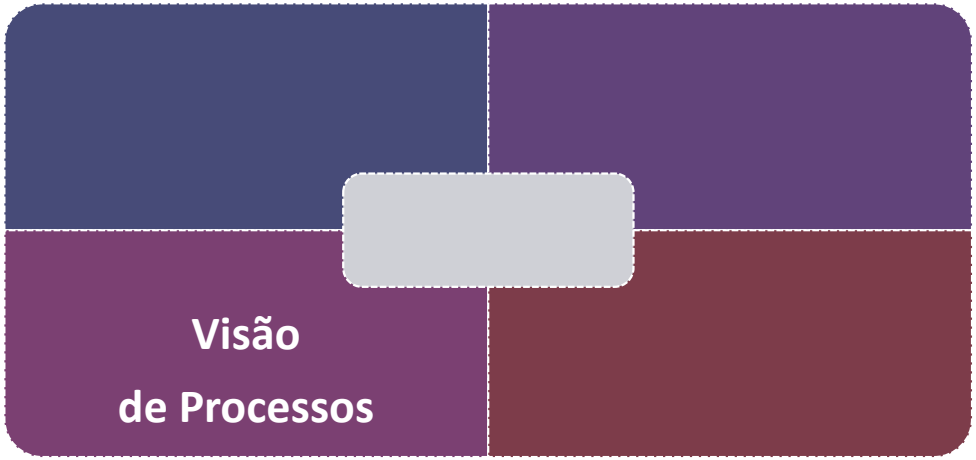
Arquitetura de Software

- Visão de Implantação/Física - Exemplo



Arquitetura de Software

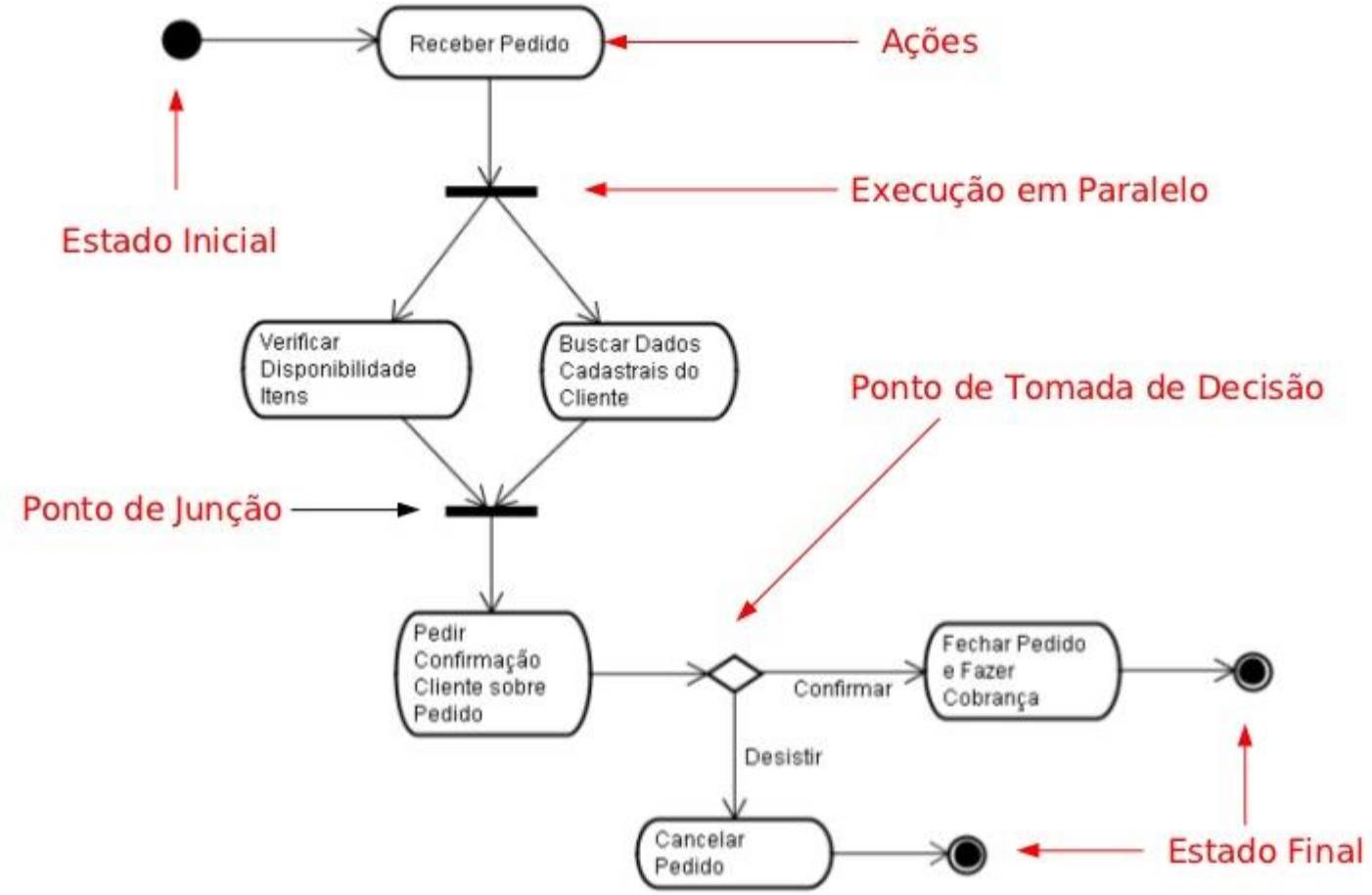
- Visões da Arquitetura de Software (Modelo 4+1)



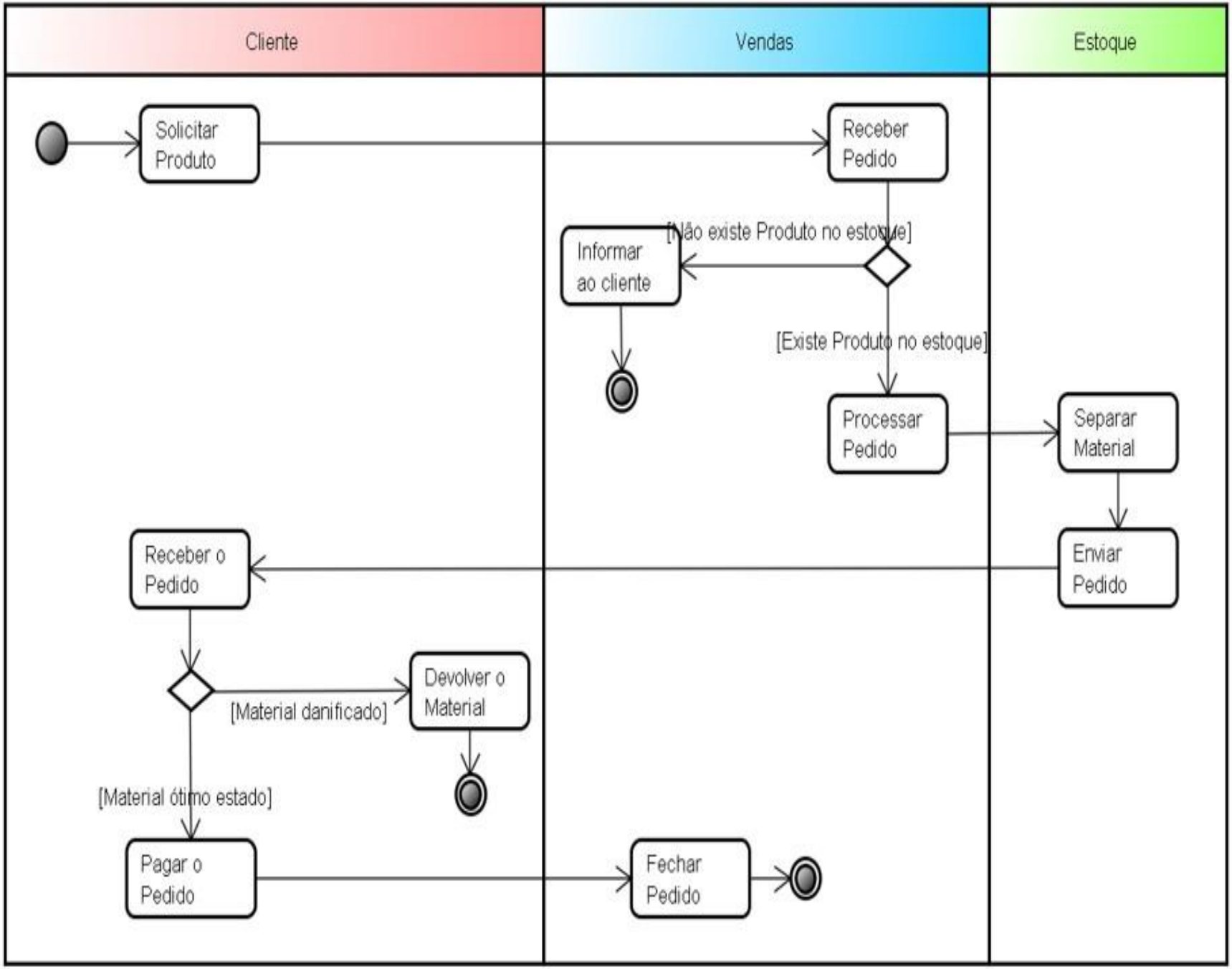
  - Perspectiva
    - Integradores de Sistema/Testadores
    - Considera questões de desempenho, escalabilidade e processamento.
      - Descrever o fluxo de atividades em um processo
      - Descreve aspectos simultâneos do sistema.
        - Processos concorrentes (threads) devem ser definidos nessa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

Arquitetura de Software

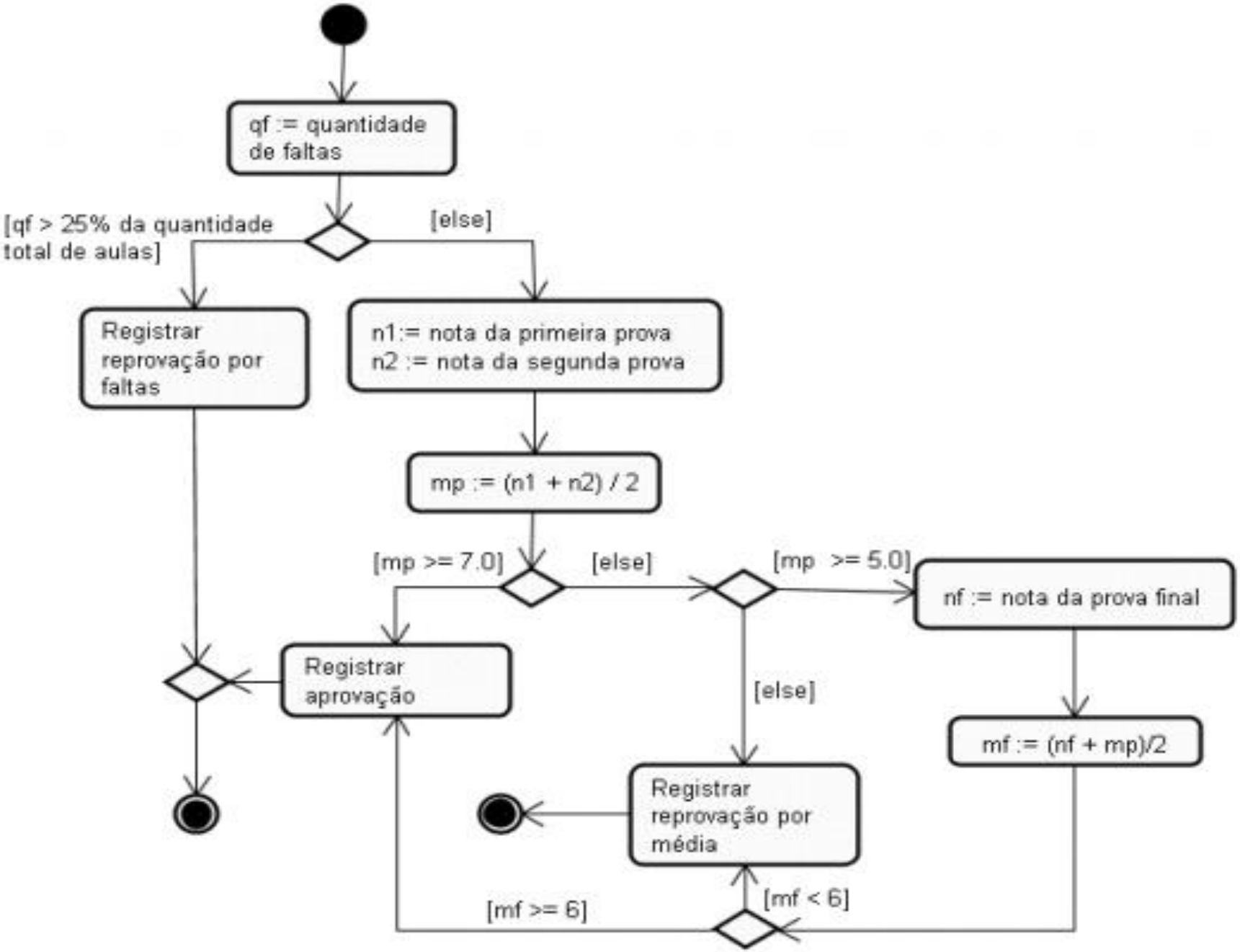
- Visão de Processos - Exemplo



- Visão de Processos



- Visão de Processos



Arquitetura de Software

- Exemplo

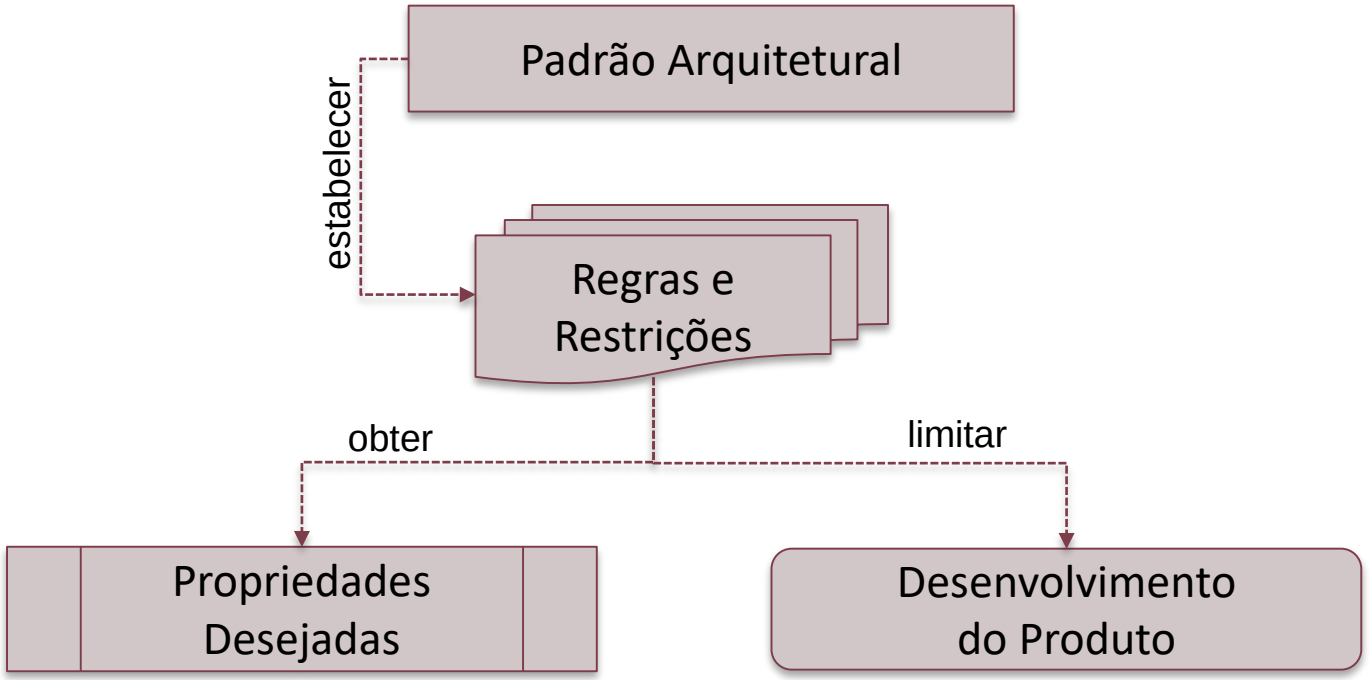
Documento de Arquitetura de Software

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.

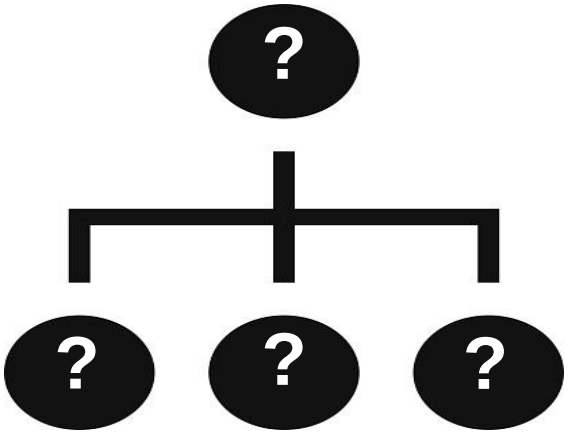
Arquitetura de Software

- Padrão Arquitetural
  - Solução estruturada de *design*, pronta para ser reutilizada para solucionar problemas recorrentes de arquitetura.



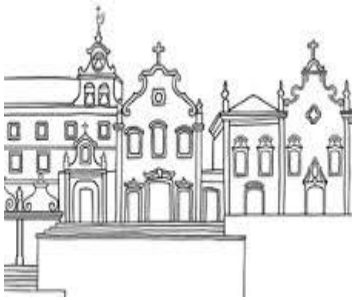
Arquitetura de Software

- 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.

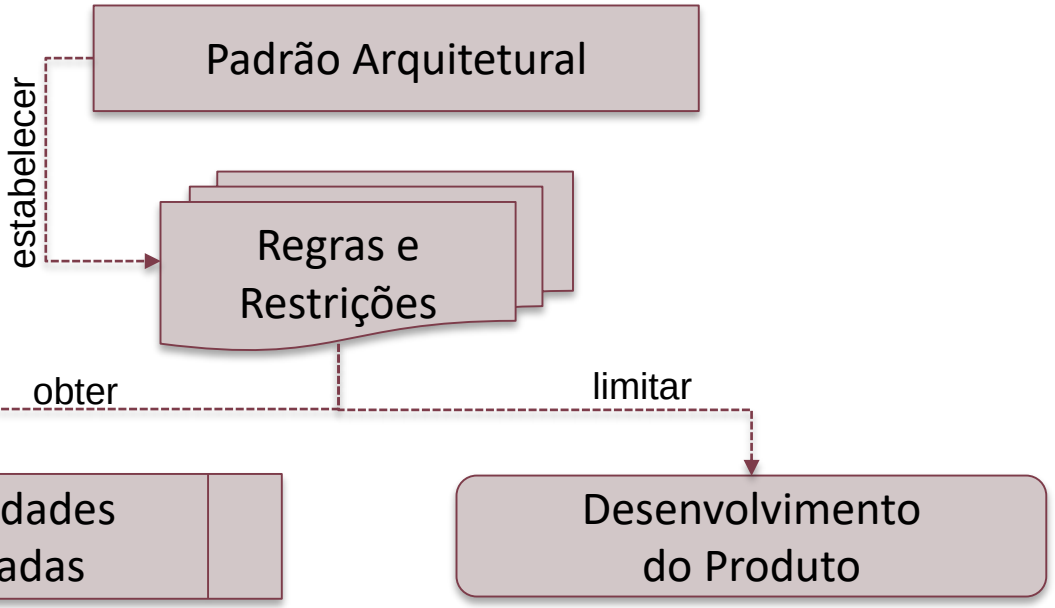


Arquitetura de Software

- Padrão Arquitetural – Na construção civil



Estilo Colonial

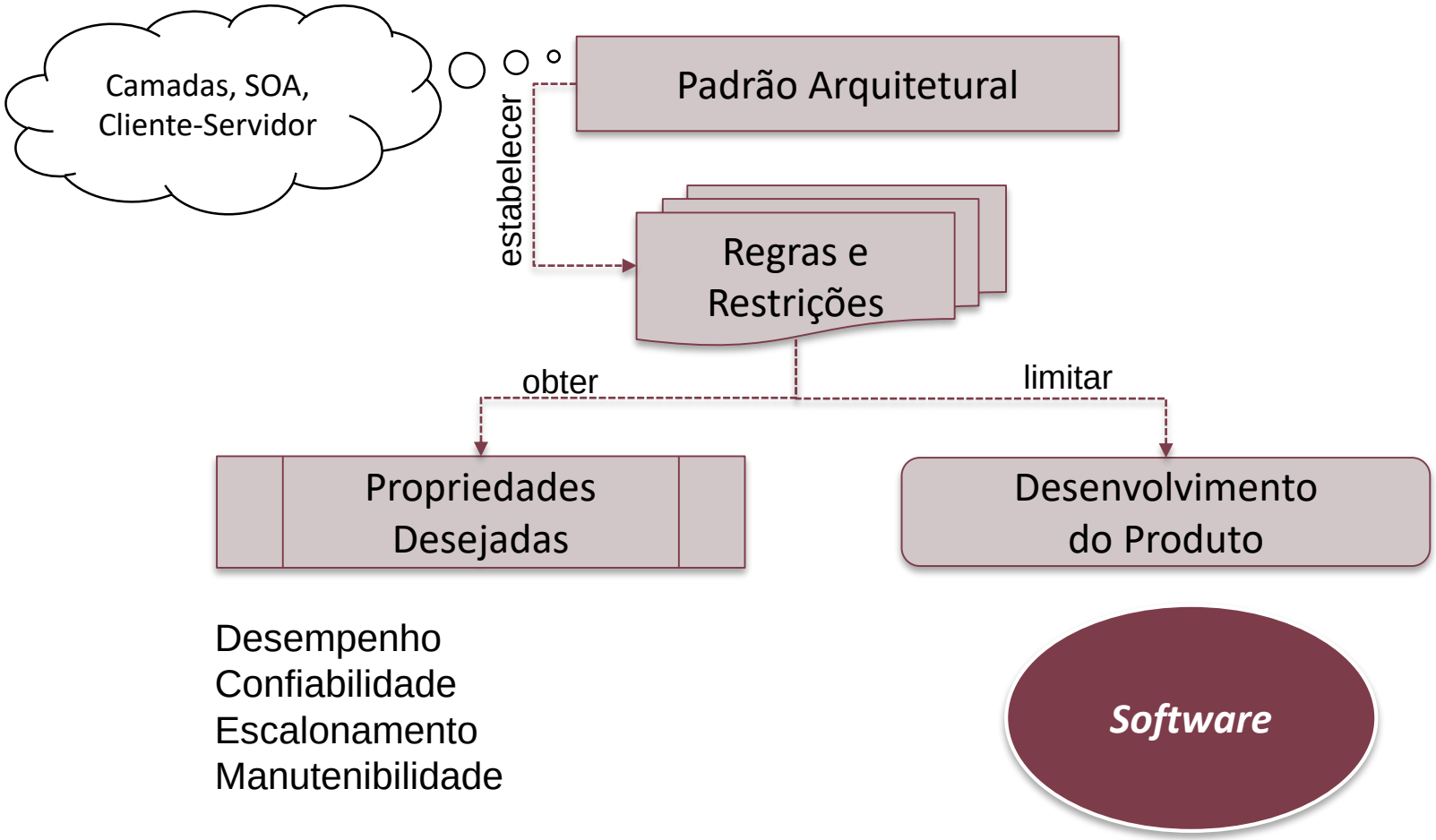


Simetria
Presença de eira e beira
Janela em arco

Arquiteturas distintas, mesmo padrão

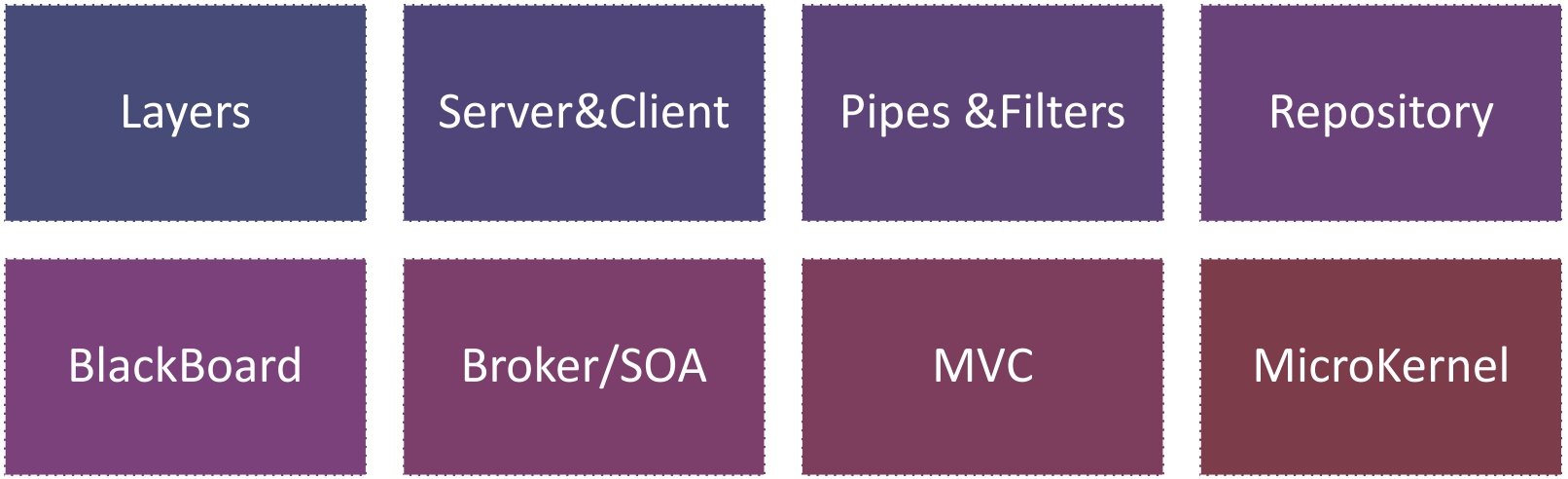
Arquitetura de Software

- Padrão Arquitetural – Na construção de software



Arquitetura de Software

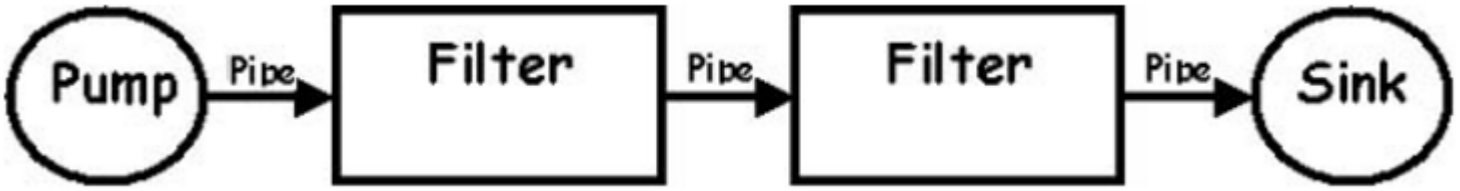
- Padrão Arquitetural
  - Cada padrão propõe uma maneira de organizar o sistema e define características que indicam quando devemos utilizá-lo.



  - Os padrões **não** são mutuamente exclusivos.

Arquitetura de Software

- *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.



Arquitetura de Software

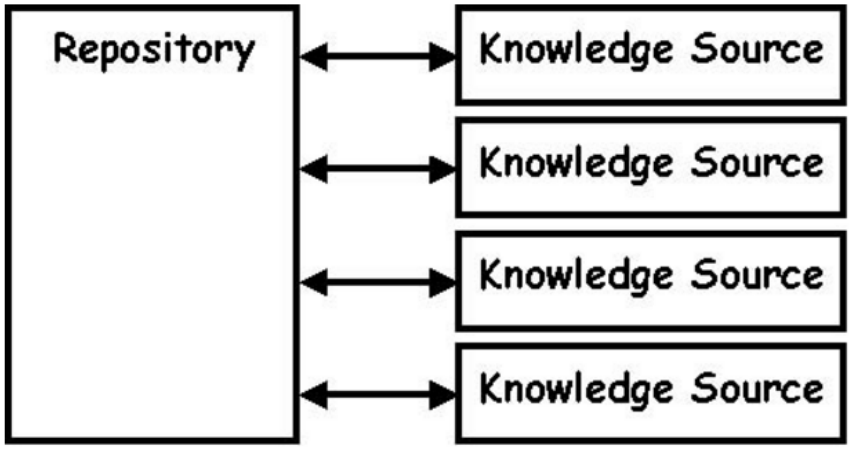
- *Pipes & Filters*
  - *Shell* do *Linux*
    - A saída de um programa pode ser “linkada” como a entrada de outro programa

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

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

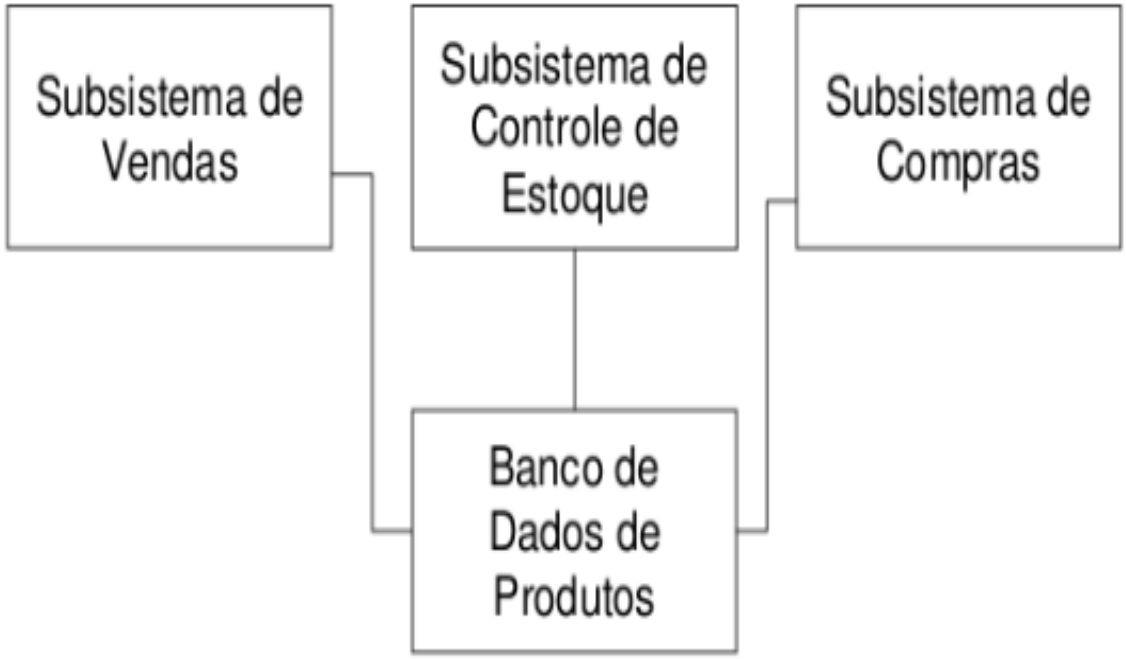
Arquitetura de Software

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



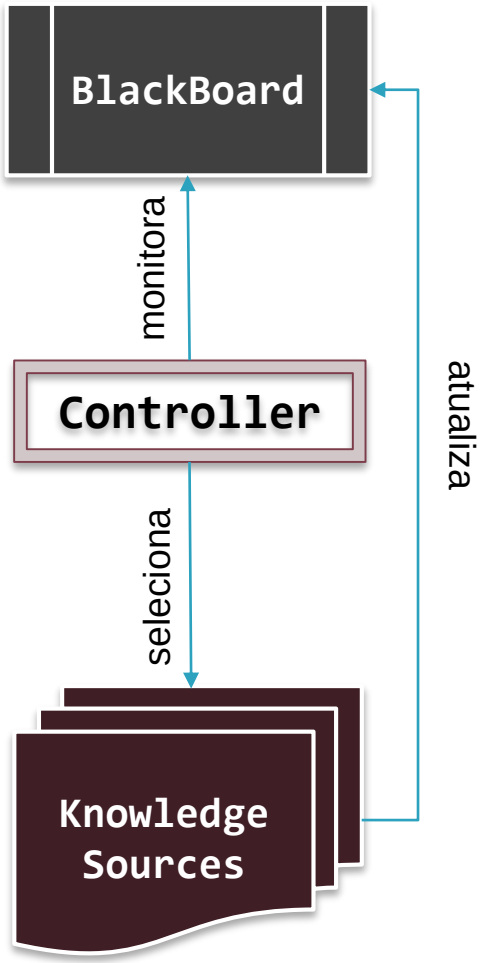
Arquitetura de Software

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



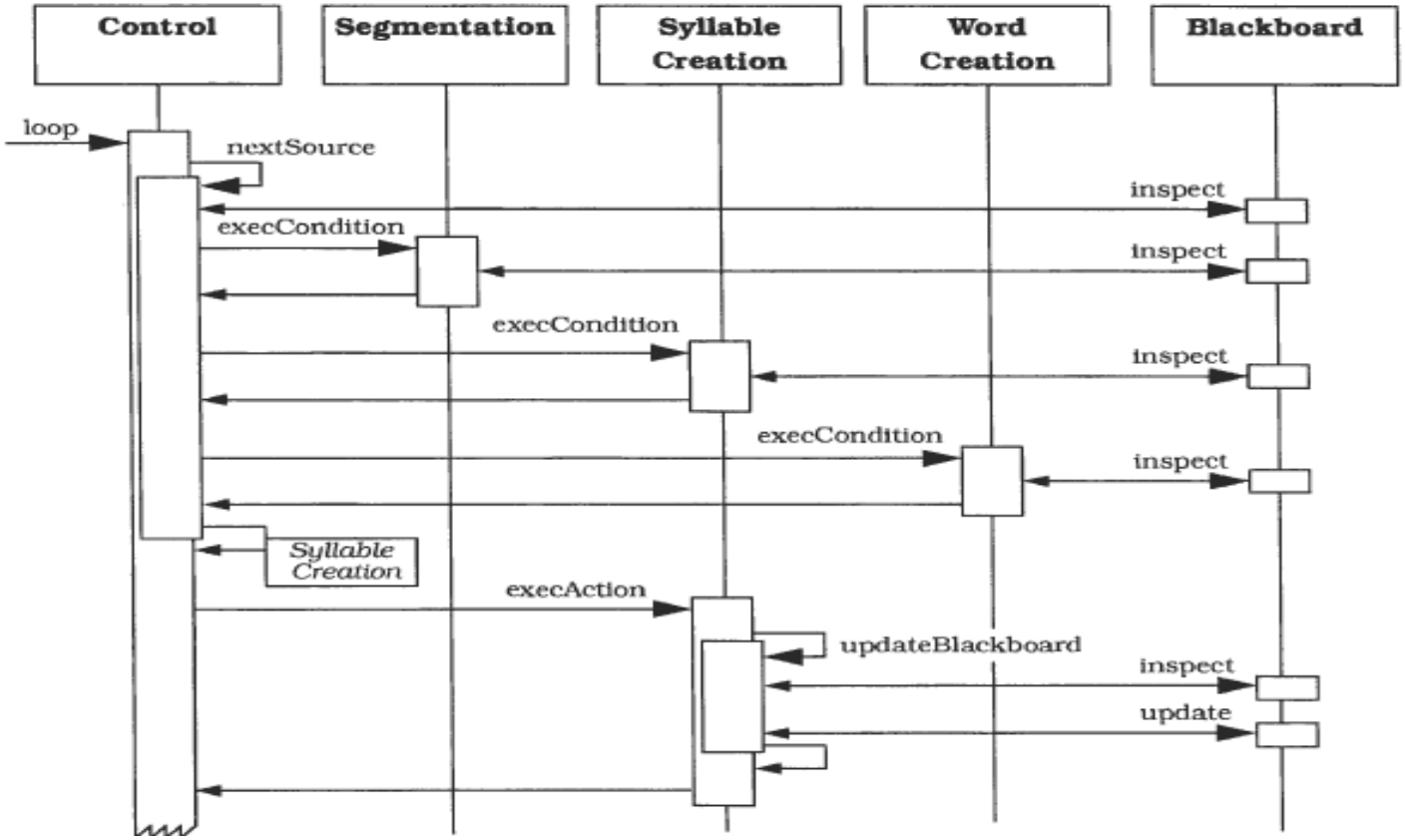
Arquitetura de Software

- *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



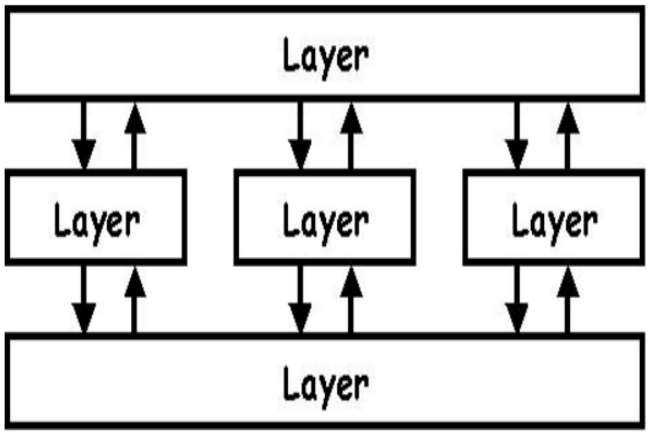
Arquitetura de Software

- *BlackBoard*
  - Programa de reconhecimento de voz.



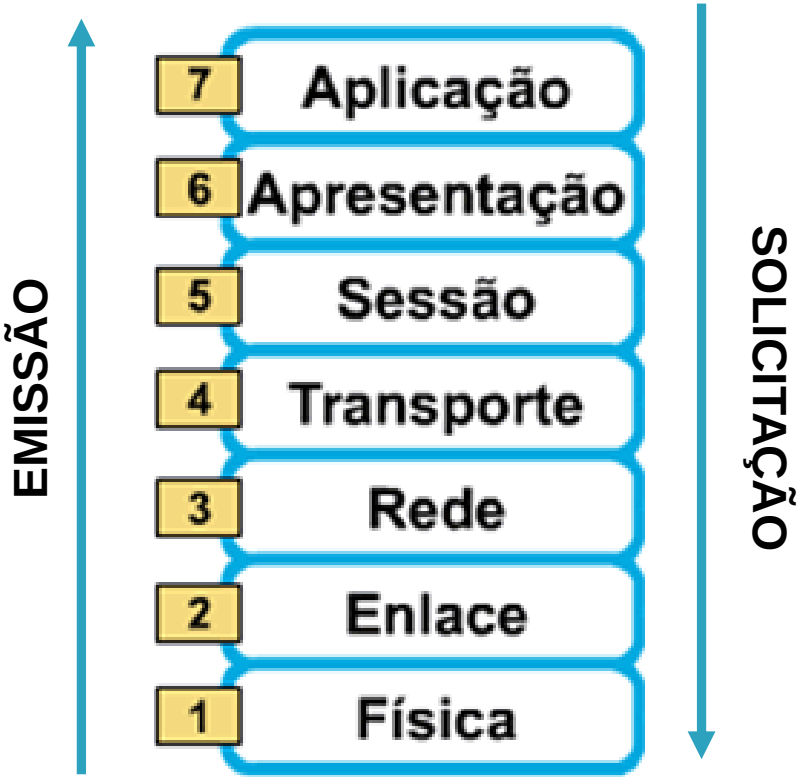
Arquitetura de Software

- *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



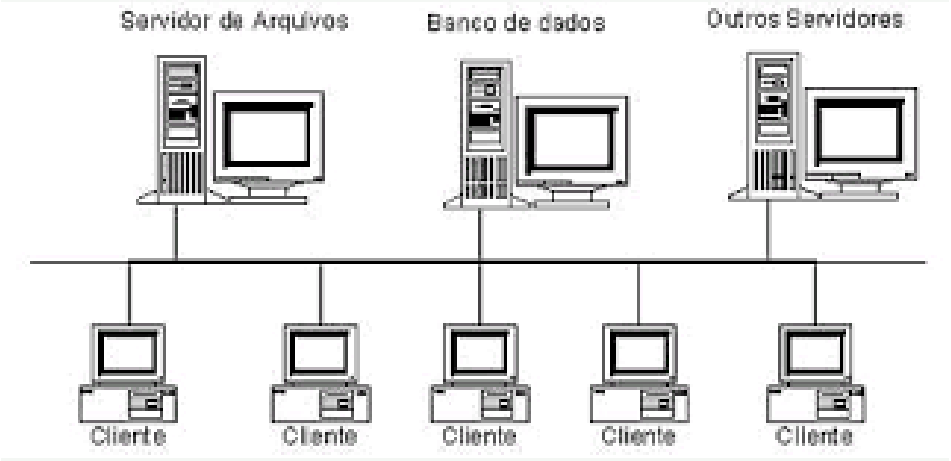
Arquitetura de Software

- *Layers* (Camadas)
  - Ex: Modelo de Referência OSI



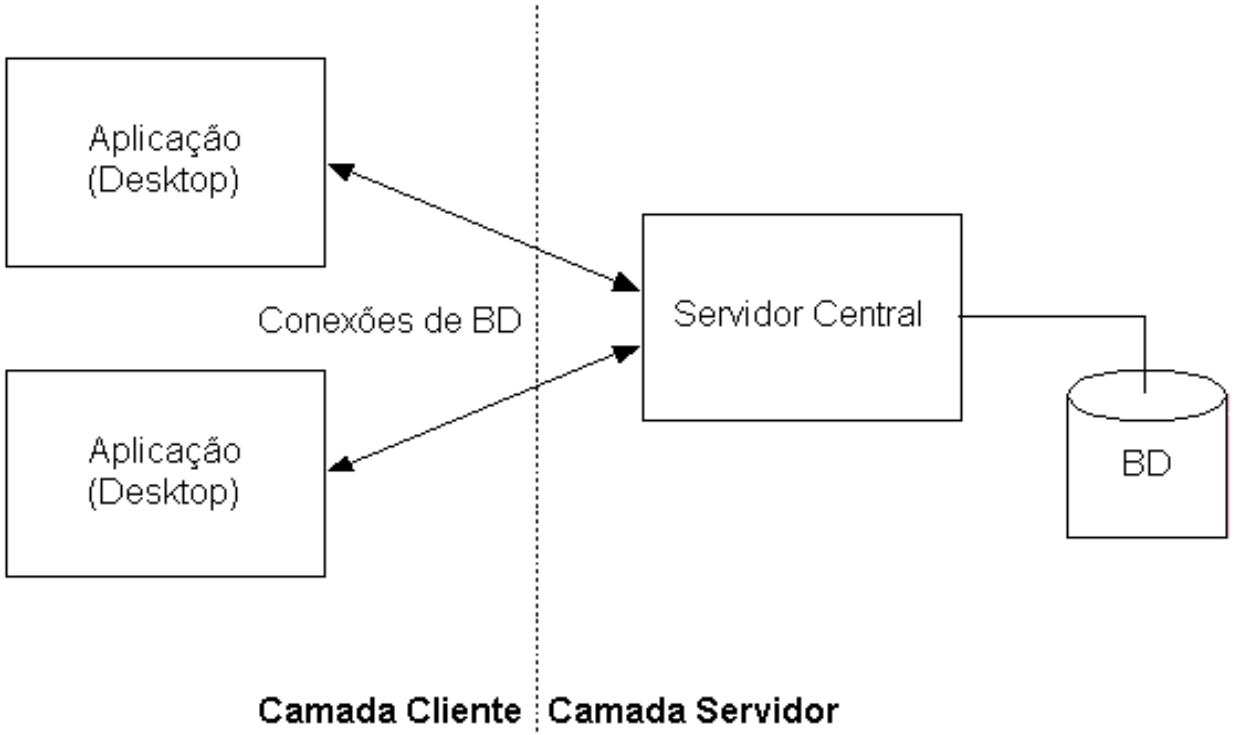
Arquitetura de Software

- 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.



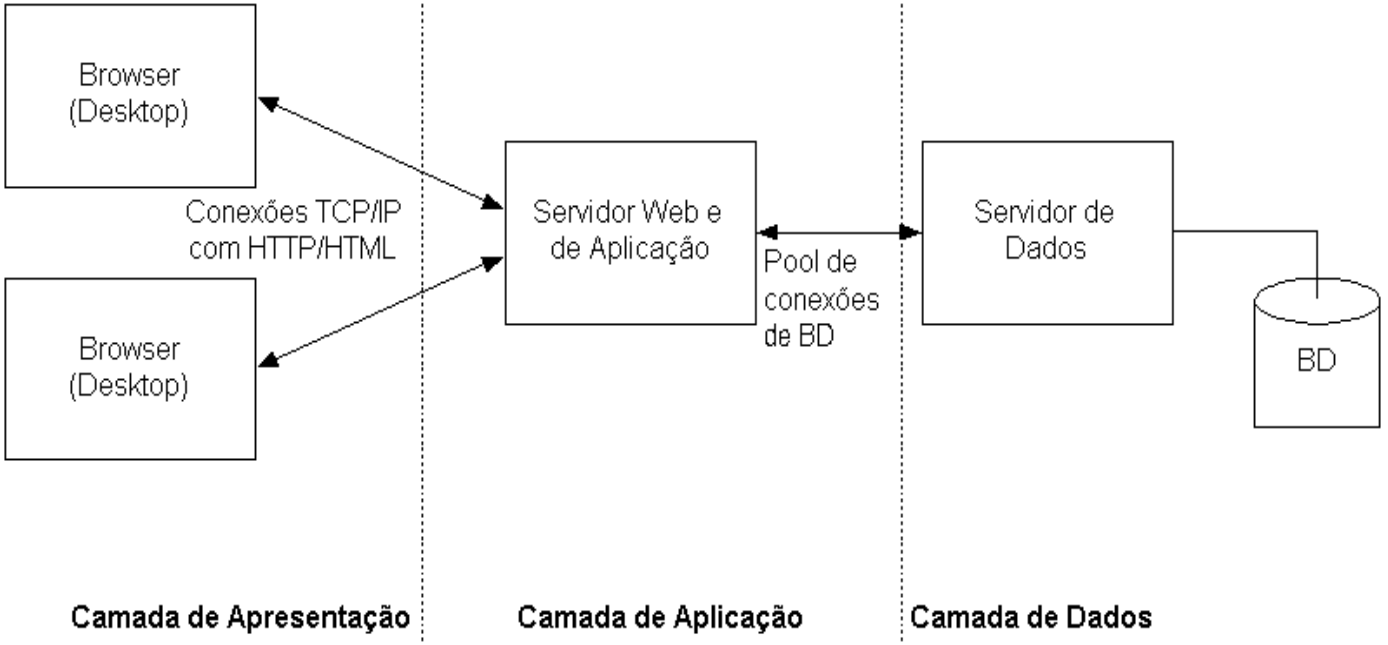
Arquitetura de Software

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



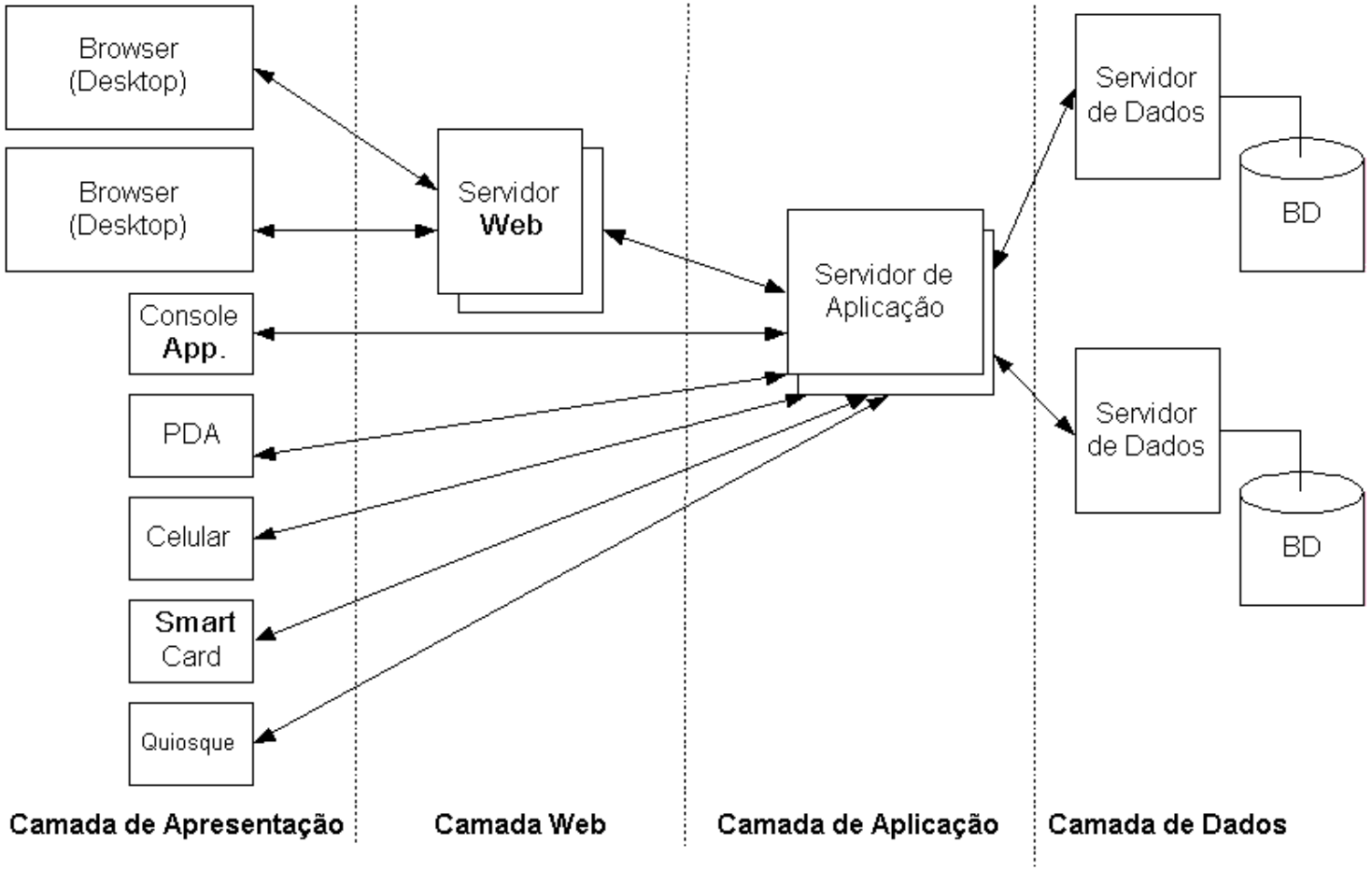
Arquitetura de Software

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



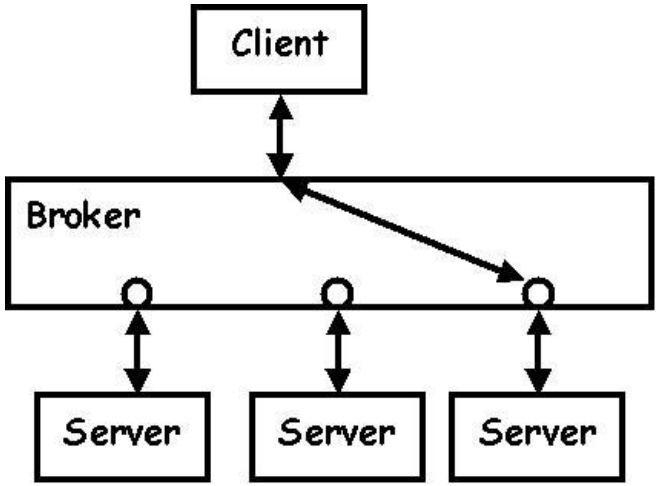
Arquitetura de Software

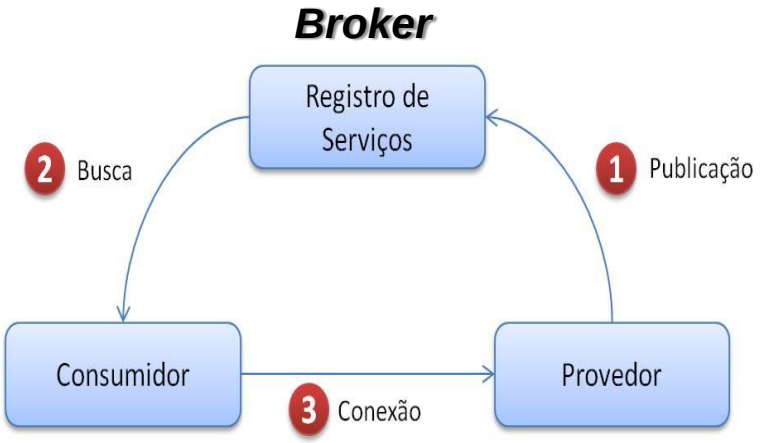
- Cliente-Servidor (N Camadas)



Arquitetura de Software

- 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*.



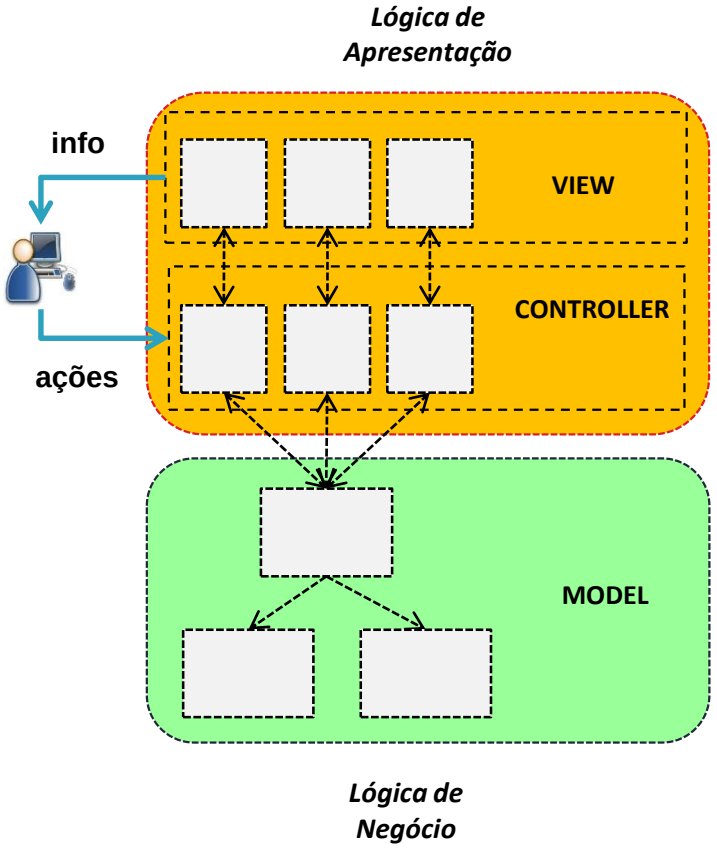


Arquitetura de Software

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

Arquitetura de Software

- MVC – *Model View Controller*
  - Contexto
    - Sistemas interativos.
    - Separação entre lógica de negócio e lógica de apresentação.
  - Características
    - Camadas
      - *Model*
        - Regras de negócio e acesso a dados
      - *View*
        - Interface com o usuário
      - *Controller*
        - Intermedia a comunicação entre Model e View.



Arquitetura de Software

- MVC – *Model View Controller*