Desenvolvimento Rápido de Aplicações

Padrões de 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...

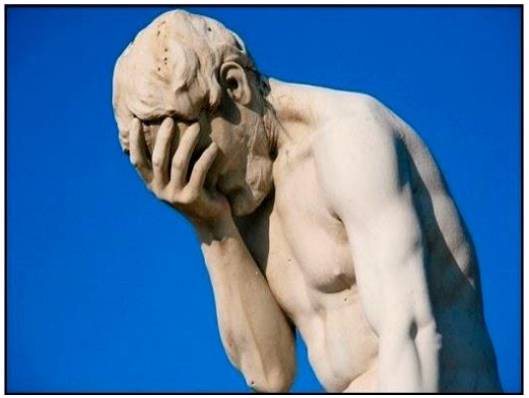
Arquitetura de Software

- Uma historinha...

  - Analisando a solução de João:
    - Para agilizar o desenvolvimento, ele tomou a sábia decisão de usar códigos prontos da internet;
    - O código foi criado sem comentários, indentação e modularização;
    - João não se preocupou com a aplicação de um padrão de arquitetura nem com a implementação de padrões de projetos.

Arquitetura de Software

- Uma historinha...
  - Desesperado, João sai mexendo no código aleatoriamente a fim de encontrar e resolver o problema.
  - Dessa forma, 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!

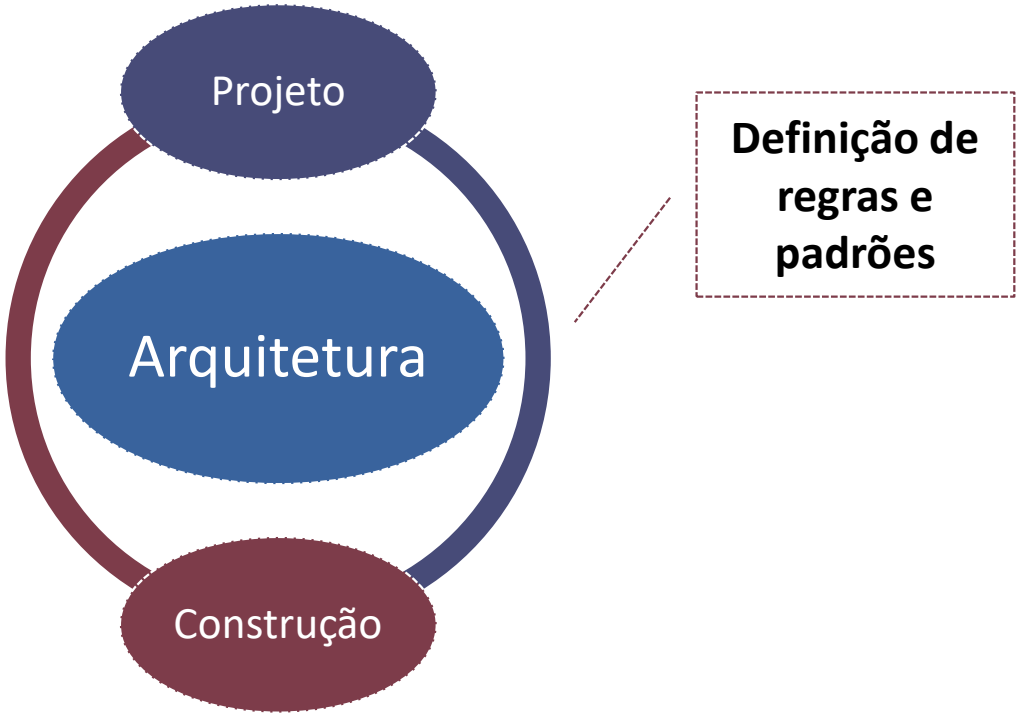


Arquitetura de Software

Desenvolver rápido não está relacionado apenas a criar o software no menor espaço de tempo, mas em desenvolver um software de qualidade em um tempo factível.

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

- Definição formal

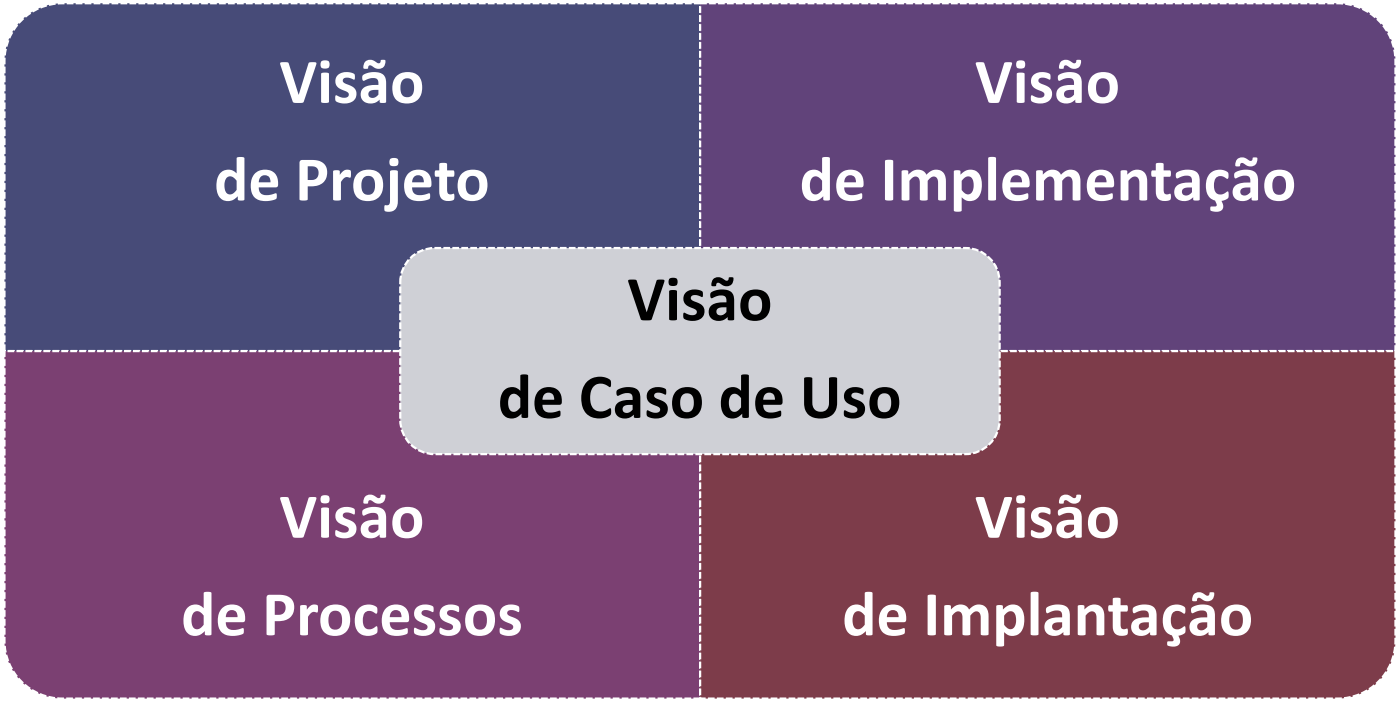
> *Organização fundamental de um sistema, expressa nos seus **componentes**, nos **relacionamentos** entre eles e com o ambiente, e nos **princípios** que governam seu projeto e sua evolução.*

Arquitetura de Software

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

Arquitetura de Software

- Visões da Arquitetura de Software
  - Modelo 4+1



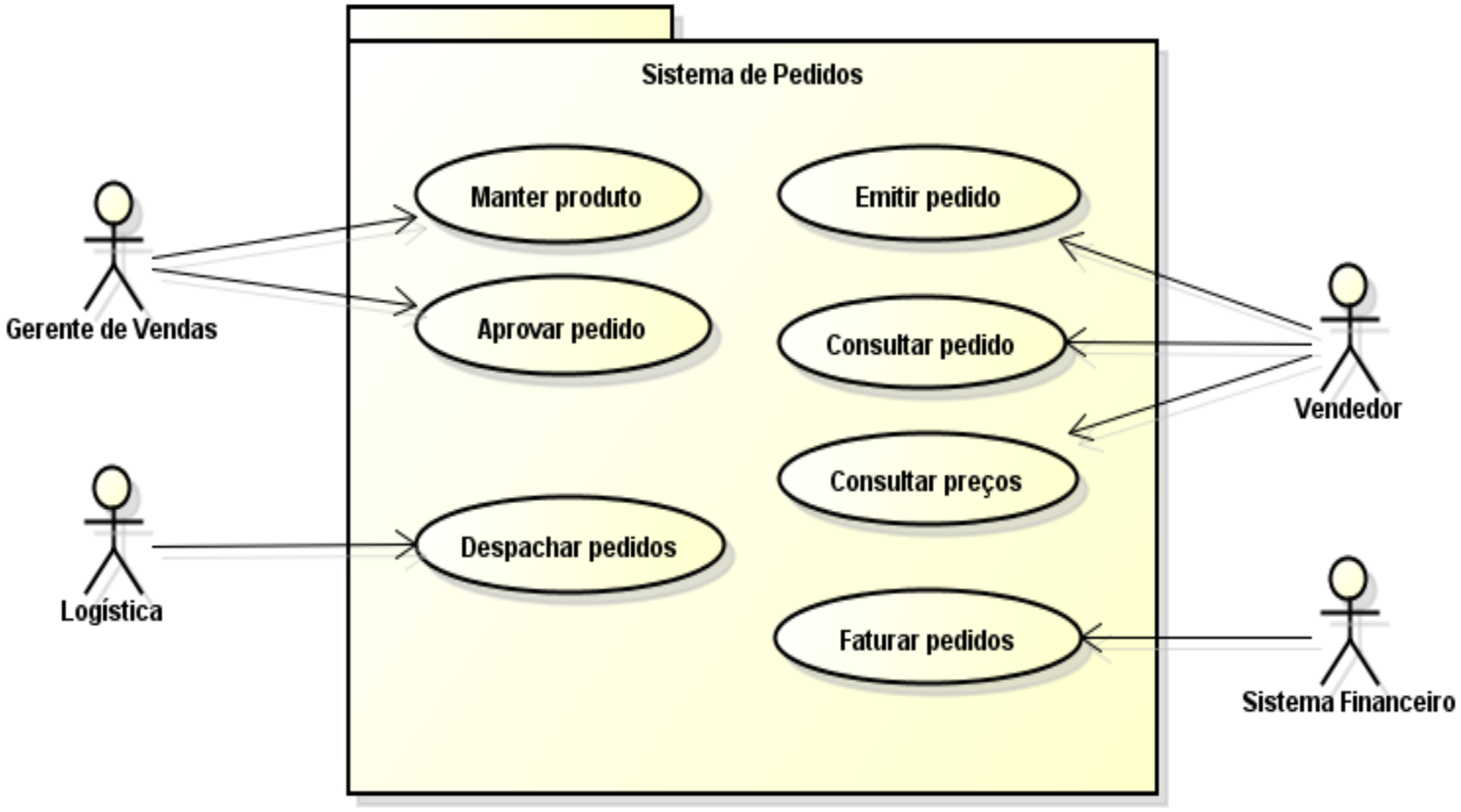
Arquitetura de Software

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

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

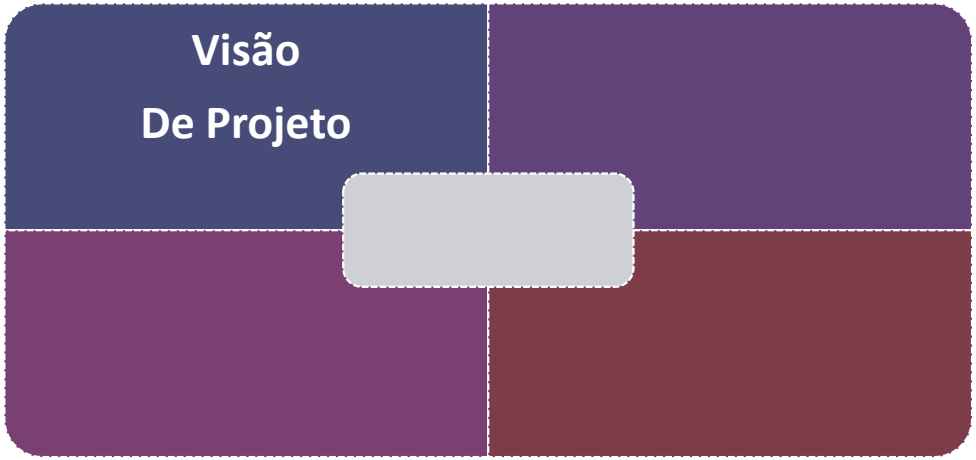
Arquitetura de Software

- Visão de Caso de Uso - Exemplo



Arquitetura de Software

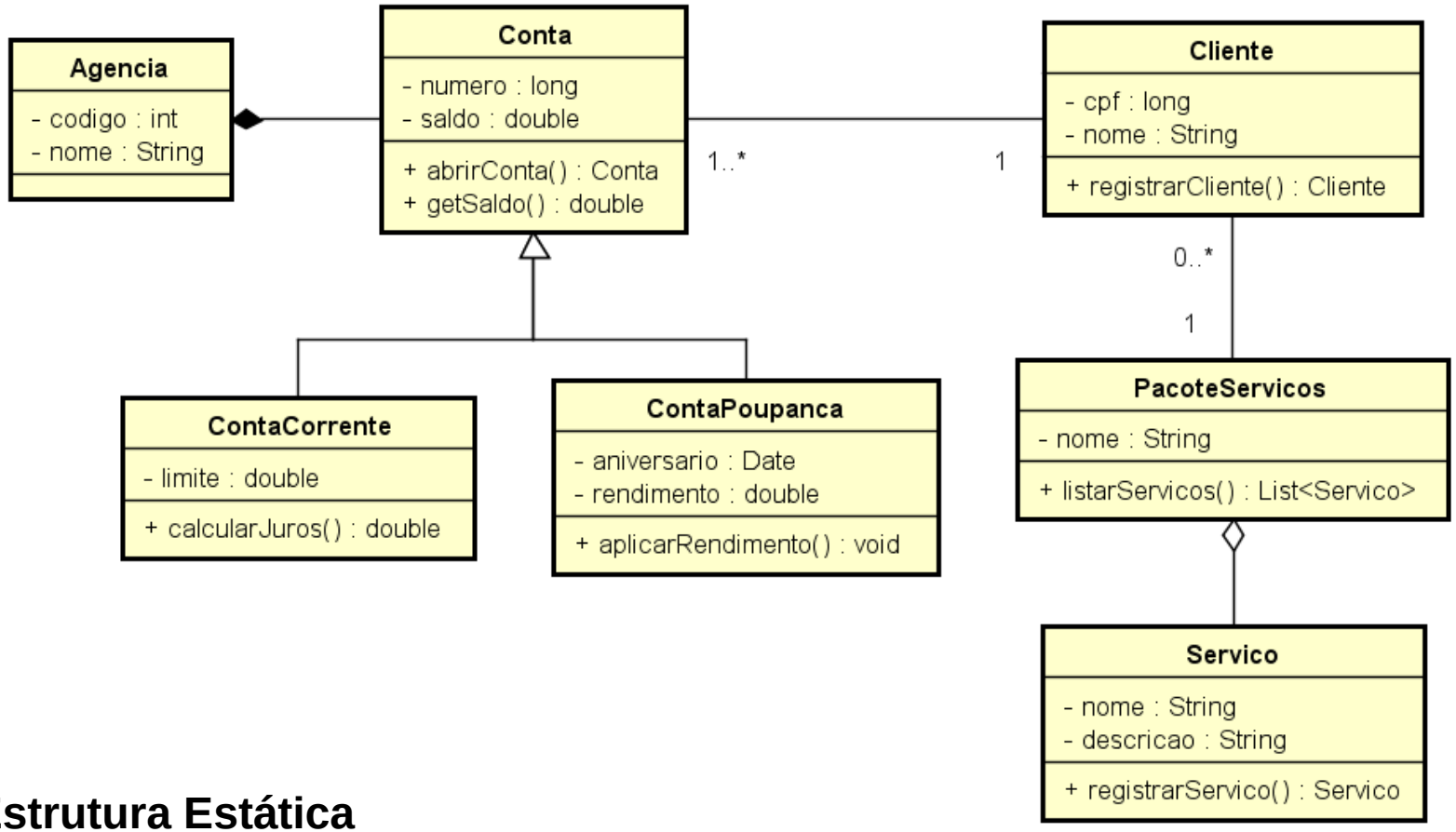
- Visões da Arquitetura de Software (Modelo 4+1)



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

Arquitetura de Software

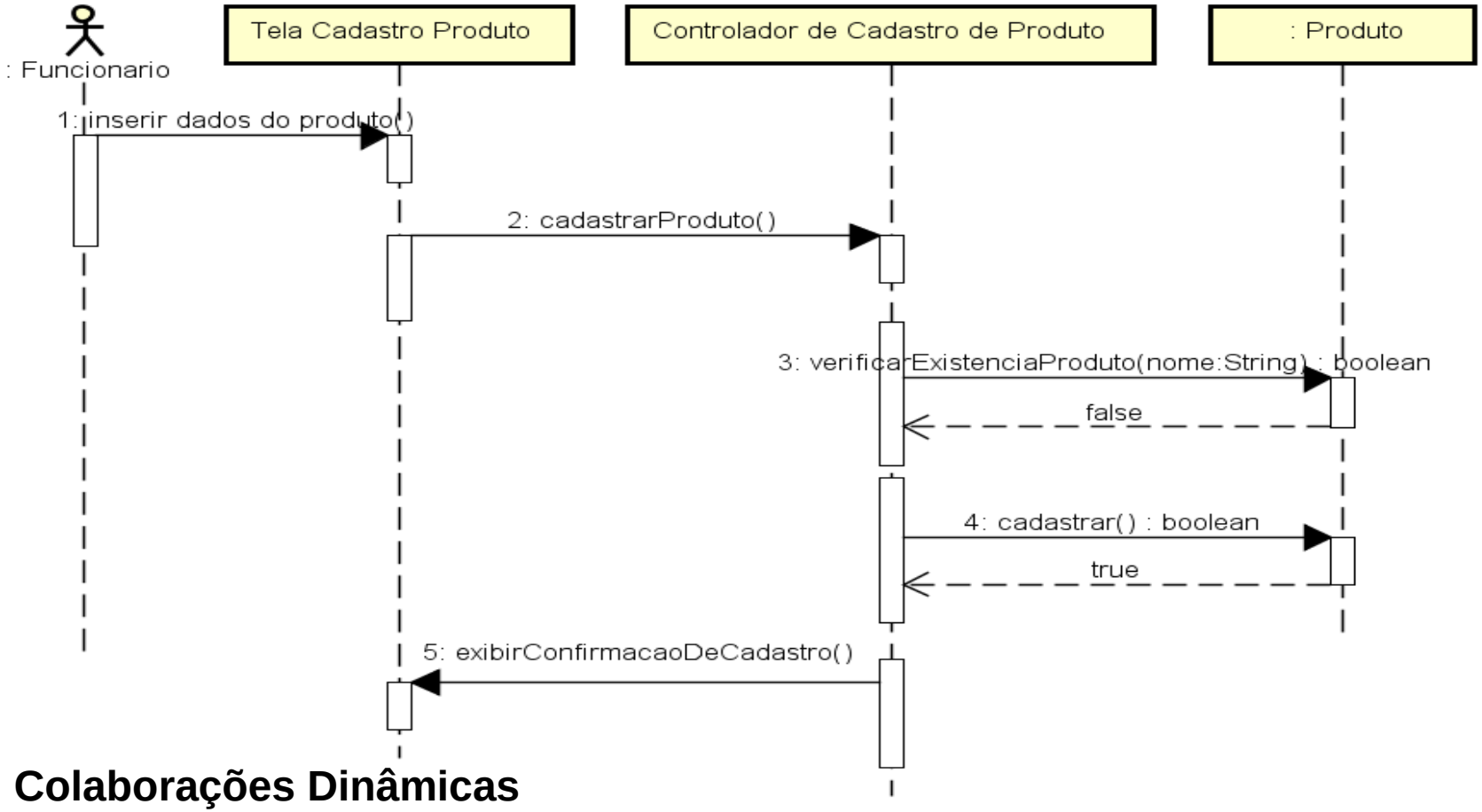
- Visão de Projeto/Lógica - Exemplo



Estrutura Estática

Arquitetura de Software

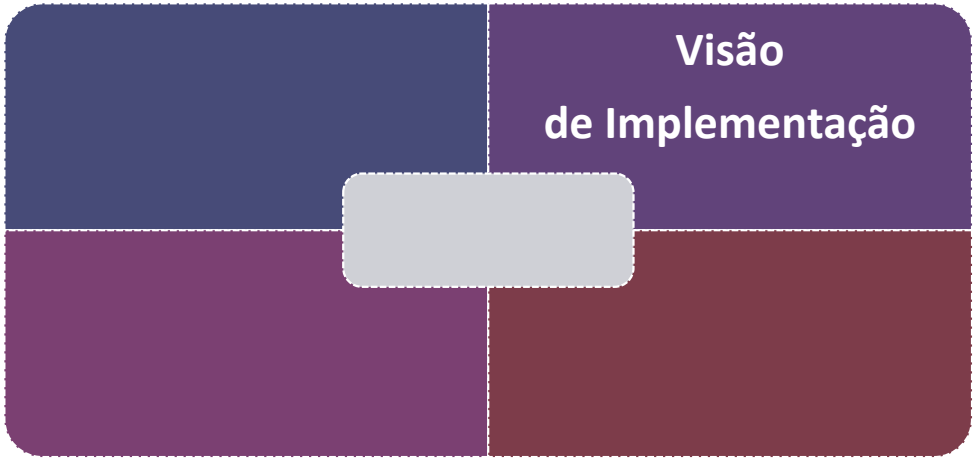
- Visão de Projeto/Lógica - Exemplo



Colaborações Dinâmicas

Arquitetura de Software

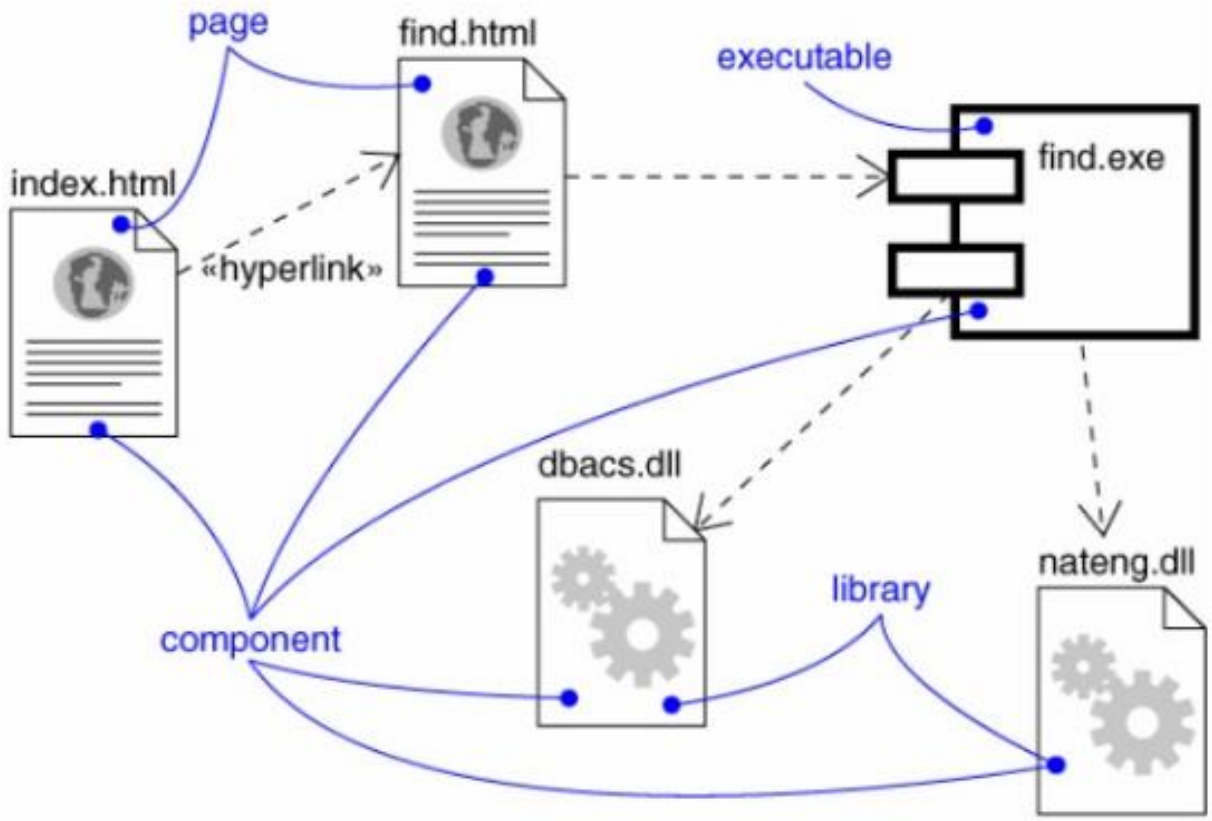
- Visões da Arquitetura de Software (Modelo 4+1)



  - 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

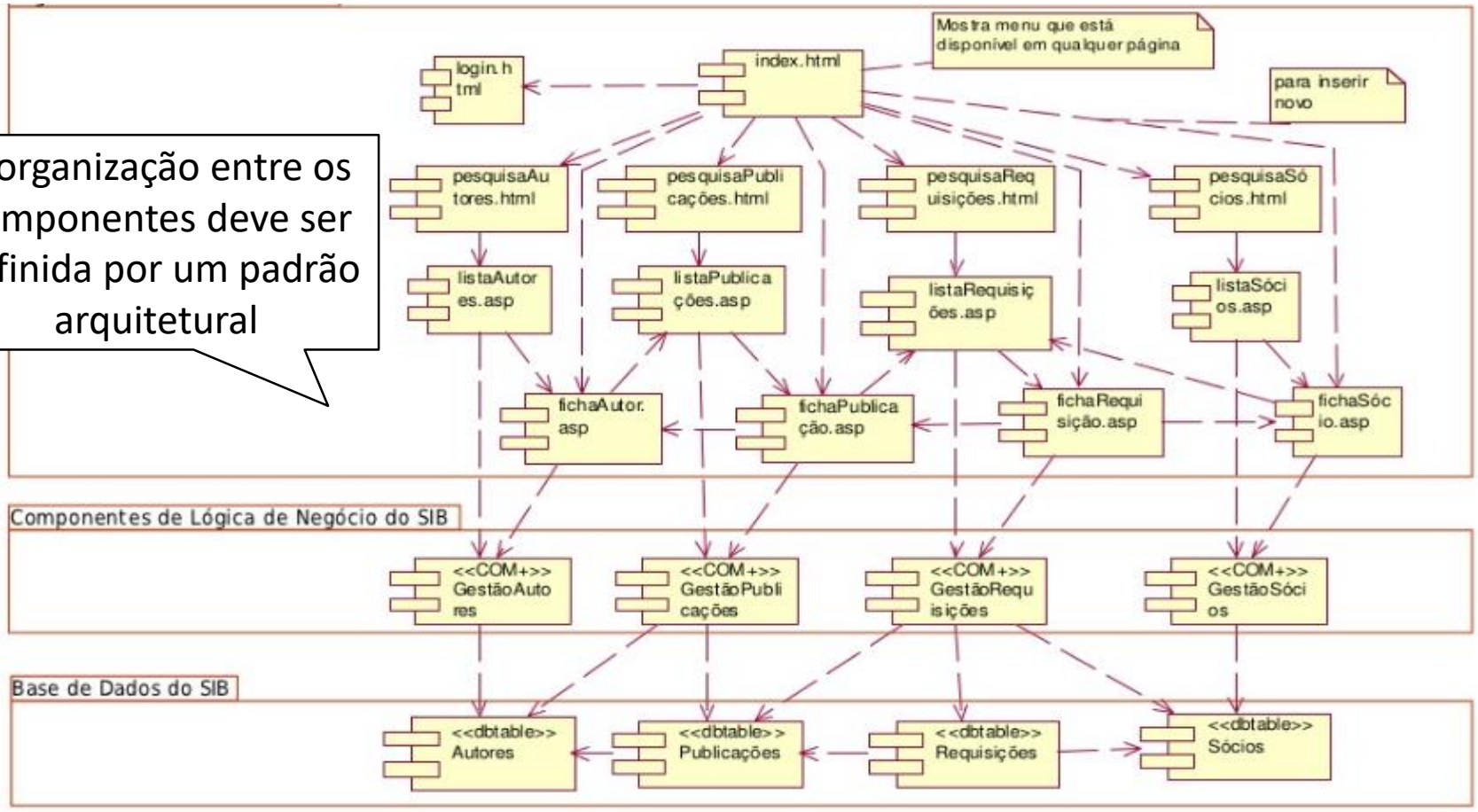
- Visão de Implementação/Componentes - Exemplo



Arquitetura de Software

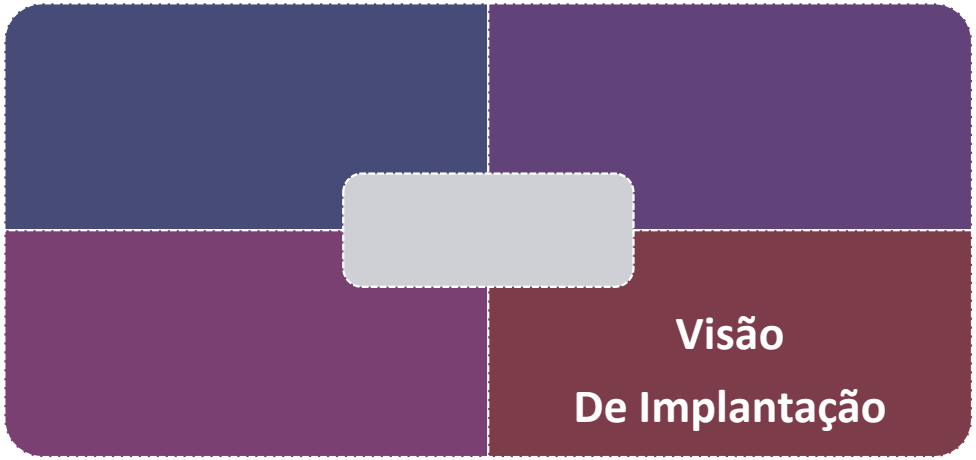
- Visão de Implementação/Componentes - Exemplo

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



Arquitetura de Software

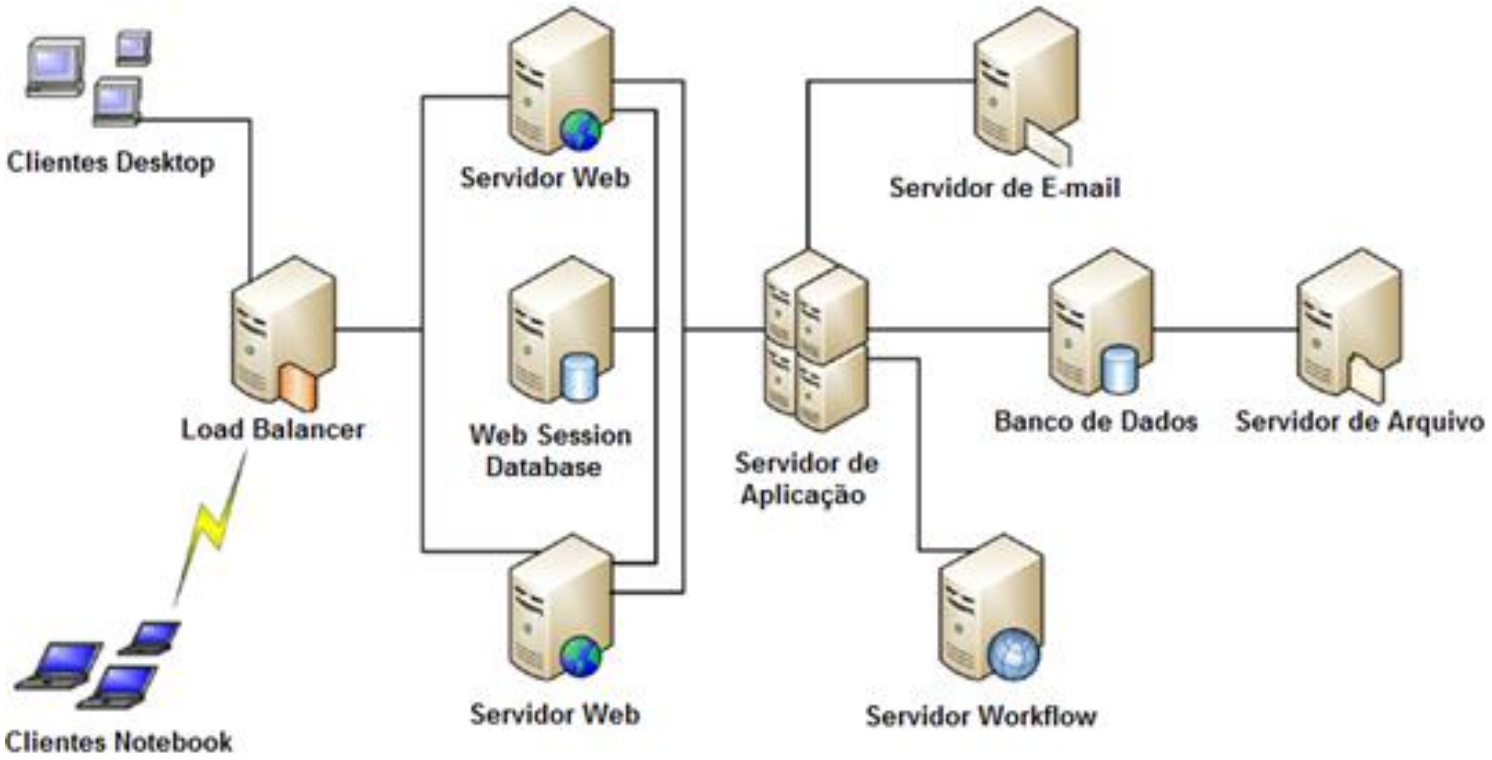
- Visões da Arquitetura de Software (Modelo 4+1)



  - 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

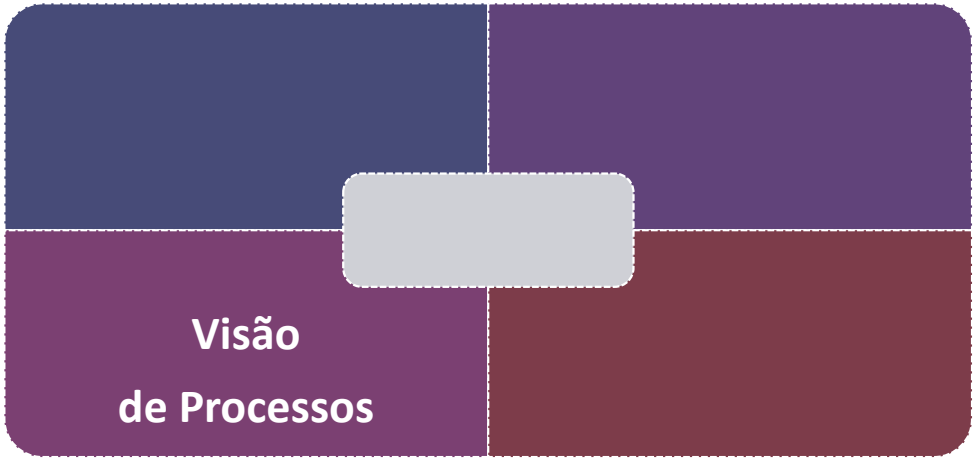
- Visão de Implantação/Física - Exemplo



Topologia de hardware

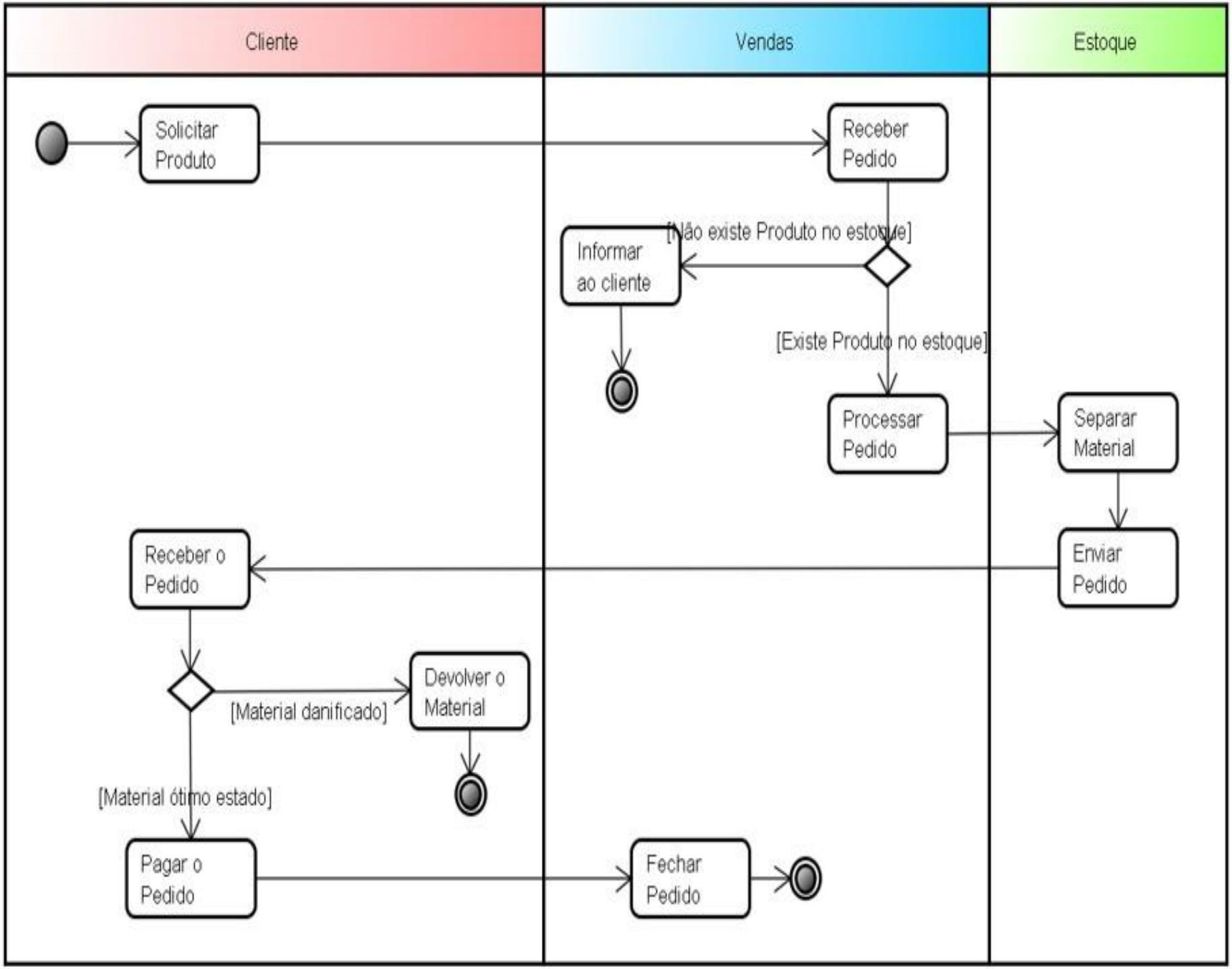
Arquitetura de Software

- 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.

- Visão de Processos



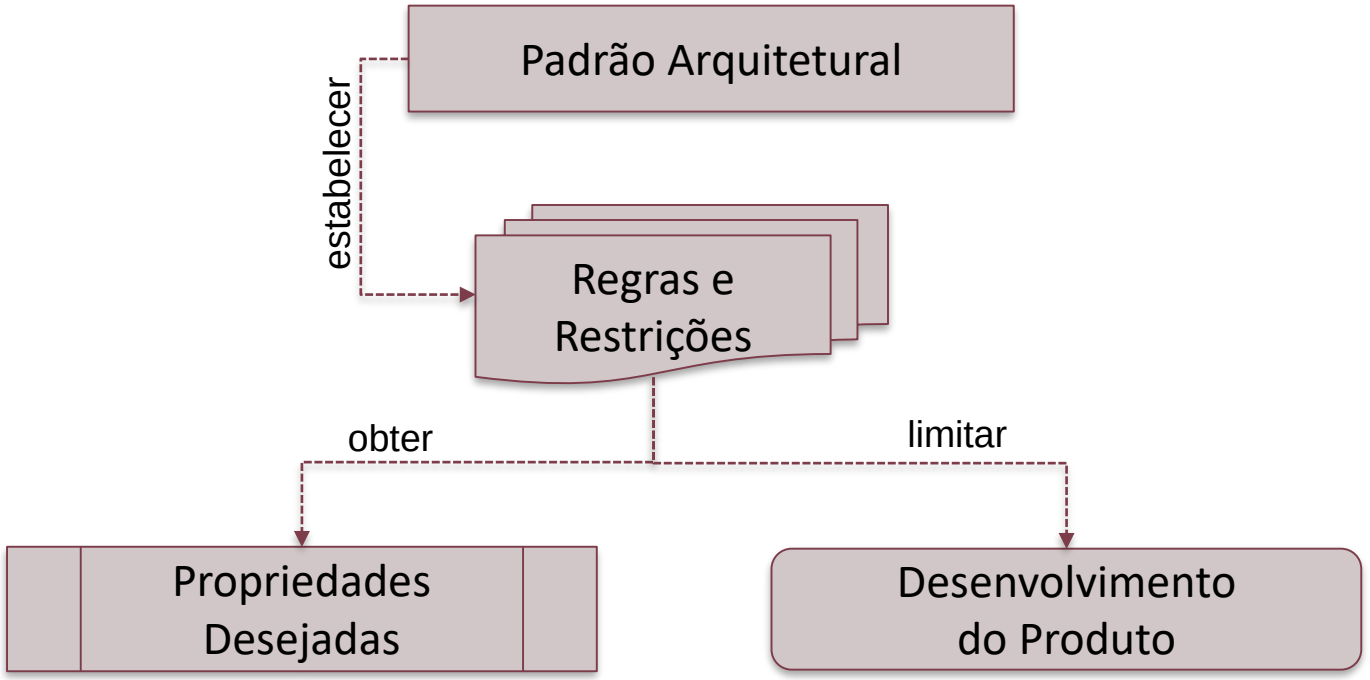
Arquitetura de Software

- Exemplo

Documento de Arquitetura de Software

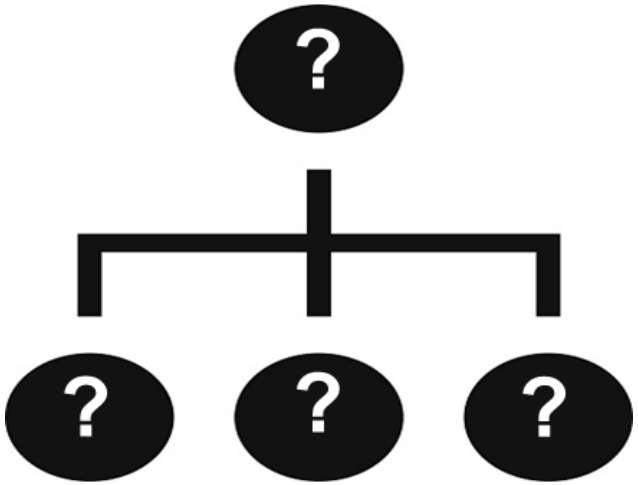
Arquitetura de Software

- Padrão Arquitetural
  - Solução estruturada pronta para ser reutilizada na solução de problemas recorrentes de arquitetura.



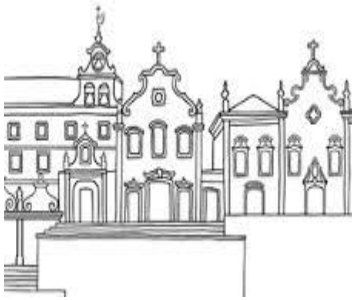
Arquitetura de Software

- Padrão Arquitetural
  - Deve ser abstrato.
  - É um *template/modelo* 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



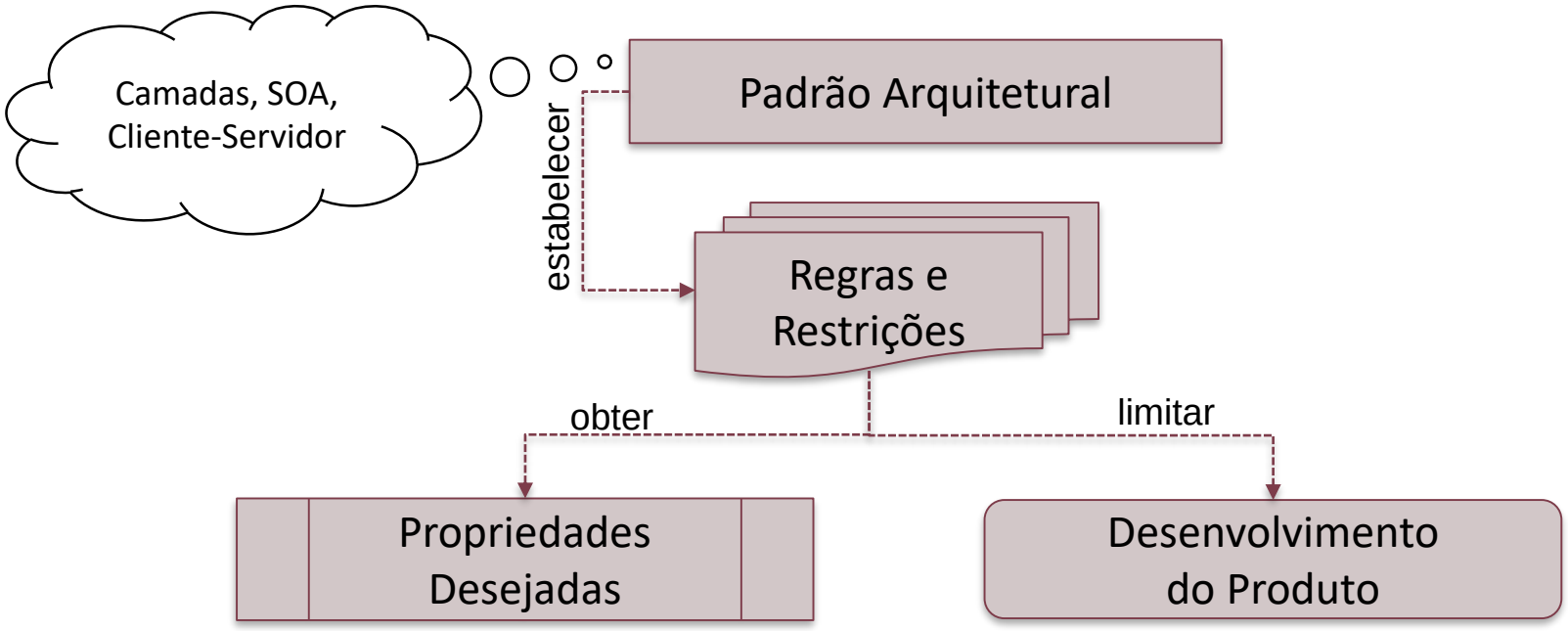
Estilo Colonial

Simetria
Presença de eira e beira
Janela em arco

Arquiteturas distintas, mesmo padrão

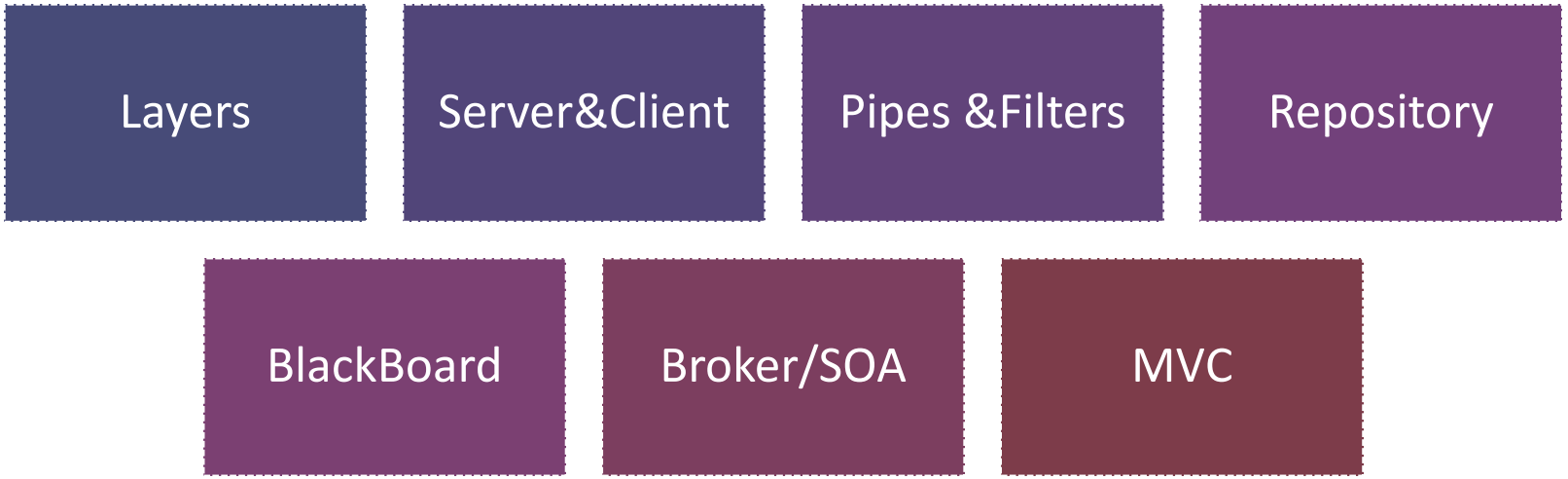
Arquitetura de Software

- Padrão Arquitetural – Na construção de software



Arquitetura de Software

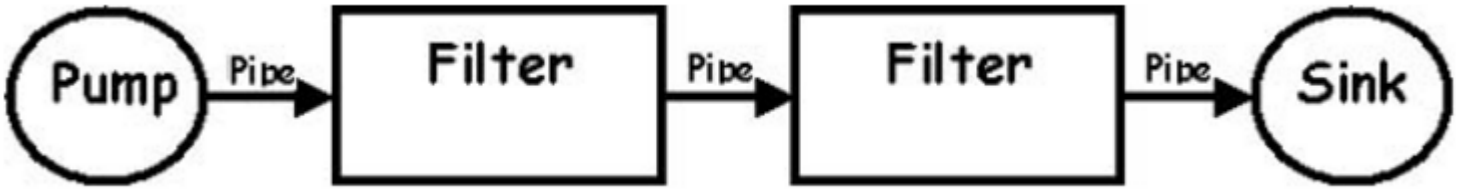
- Padrão Arquitetural
  - Cada padrão propõe uma maneira de organizar o sistema e define características que indicam quando devemos utilizá-lo.



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

Arquitetura de Software

- *Pipes & Filters*
  - Contexto
    - Divisão de uma tarefa de processamento em uma sequência de passos (Filters) que são conectados por canais (Pipes).
    - Sequência de transformações sobre uma fonte de dados.
  - 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

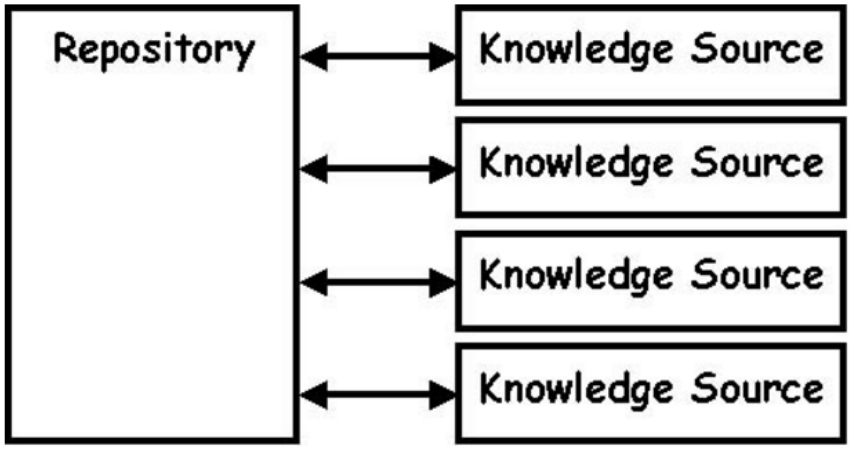
```
$ ls | grep b | sort -r | tee arquivo.out | wc -l
```

A saída será a quantidade de arquivos que contém a letra “b” e o nome desses arquivos salvos em "arquivo.out".

    - 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.

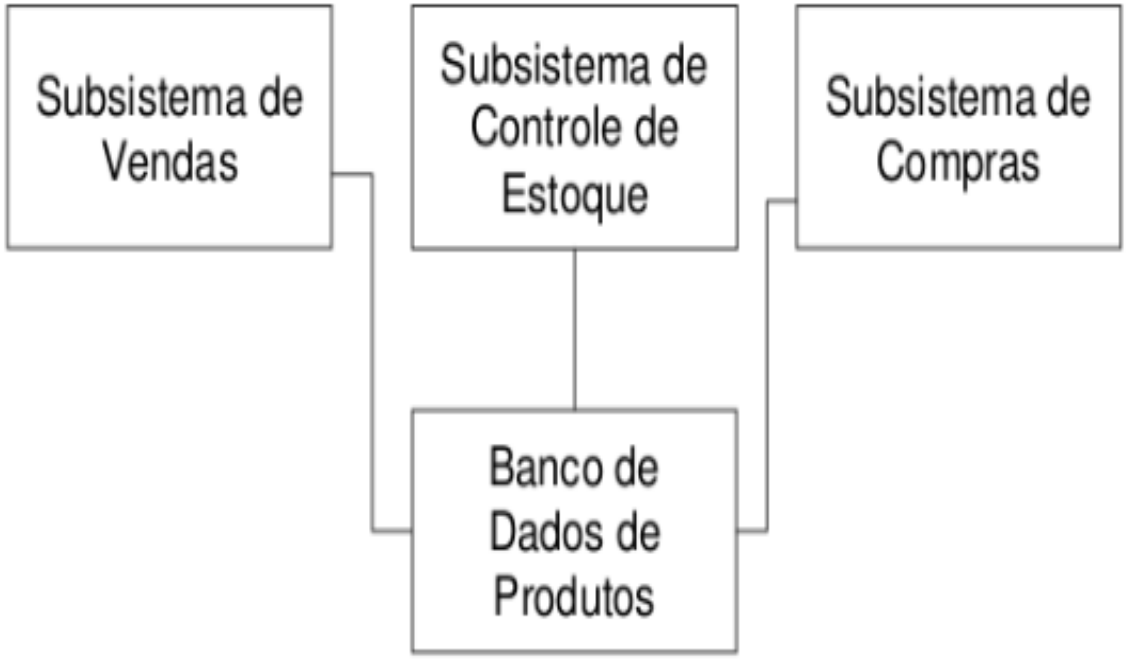
Arquitetura de Software

- *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.



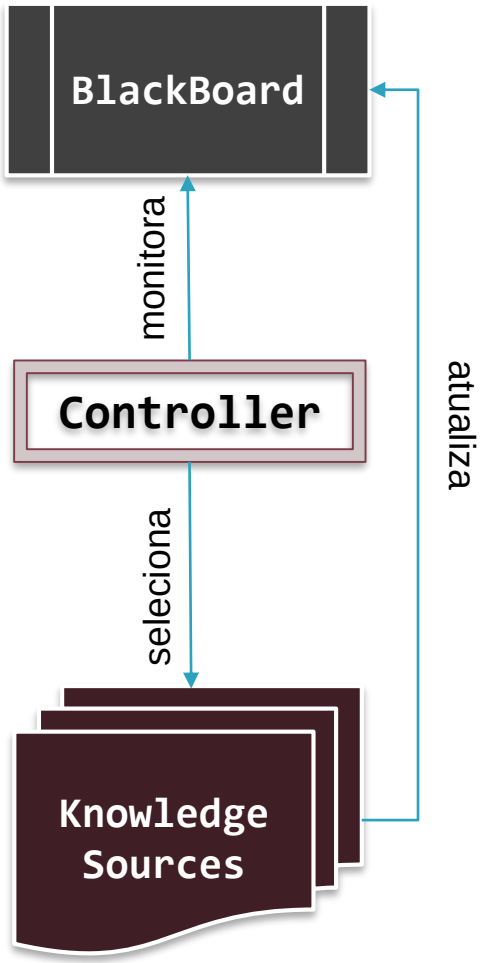
Arquitetura de Software

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



Arquitetura de Software

- *BlackBoard*
  - Contexto
    - Solução para problemas não determinísticos.
    - 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

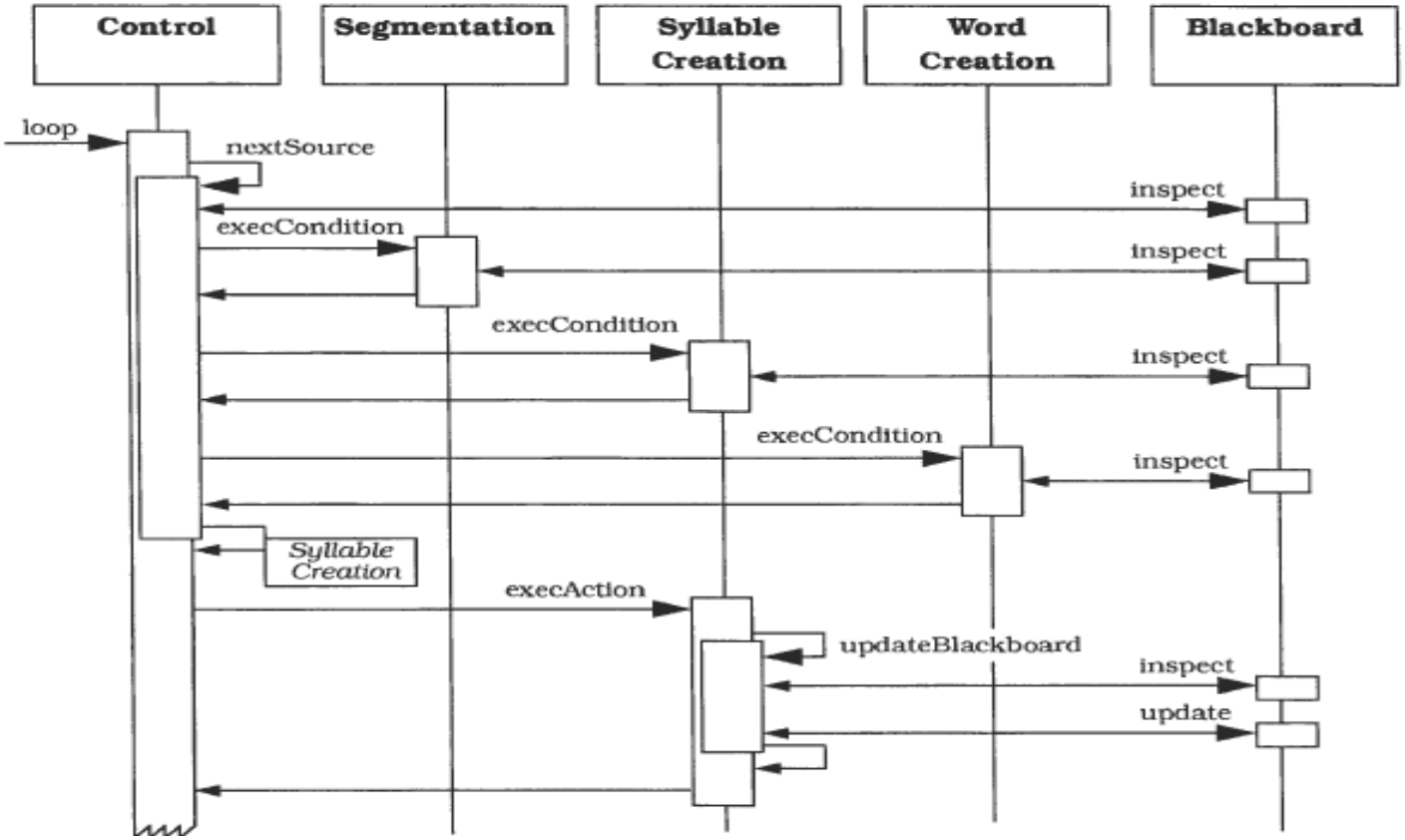


Arquitetura de Software

- *BlackBoard*

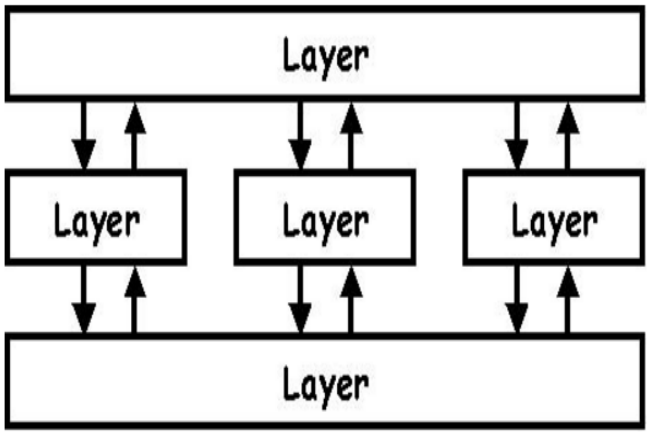
cada fonte vai criar uma hipótese sobre a possível solução

  - Programa de reconhecimento de voz.



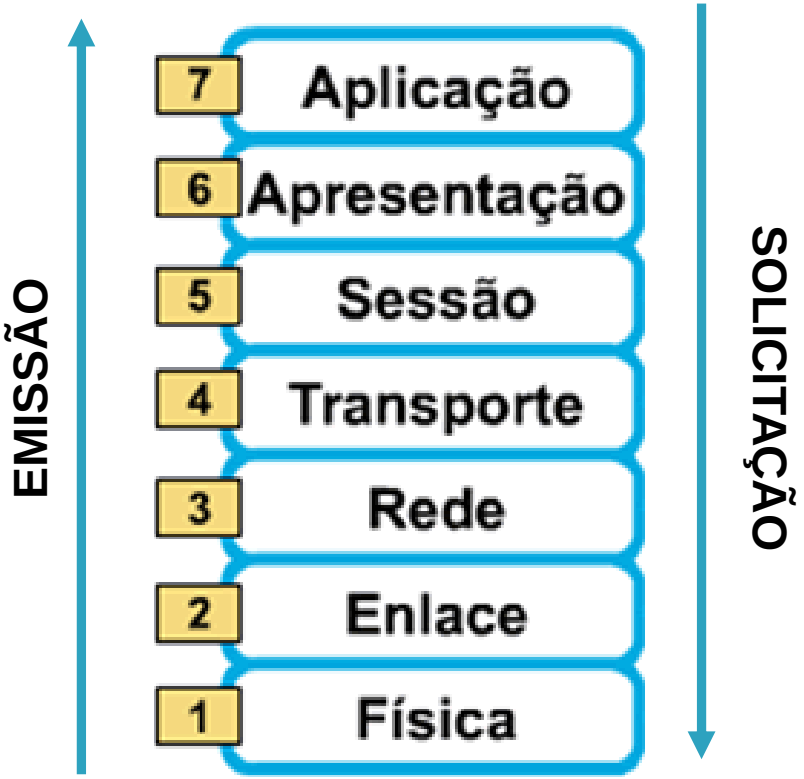
Arquitetura de Software

- *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



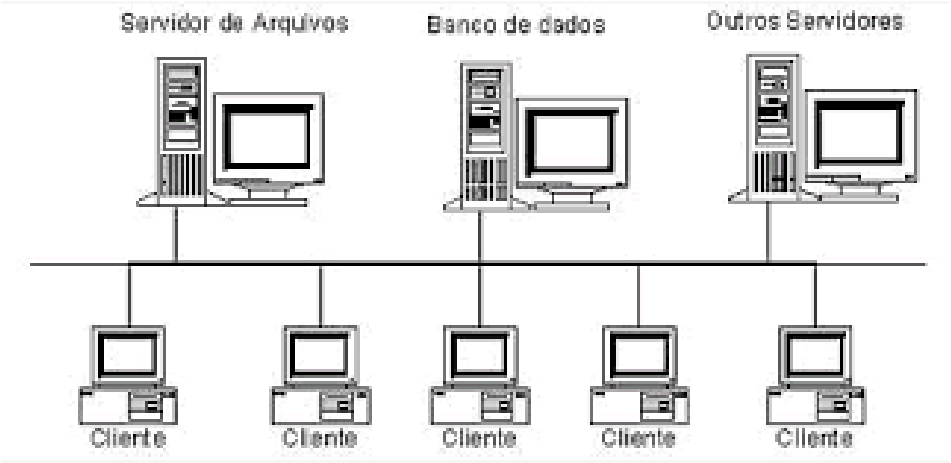
Arquitetura de Software

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



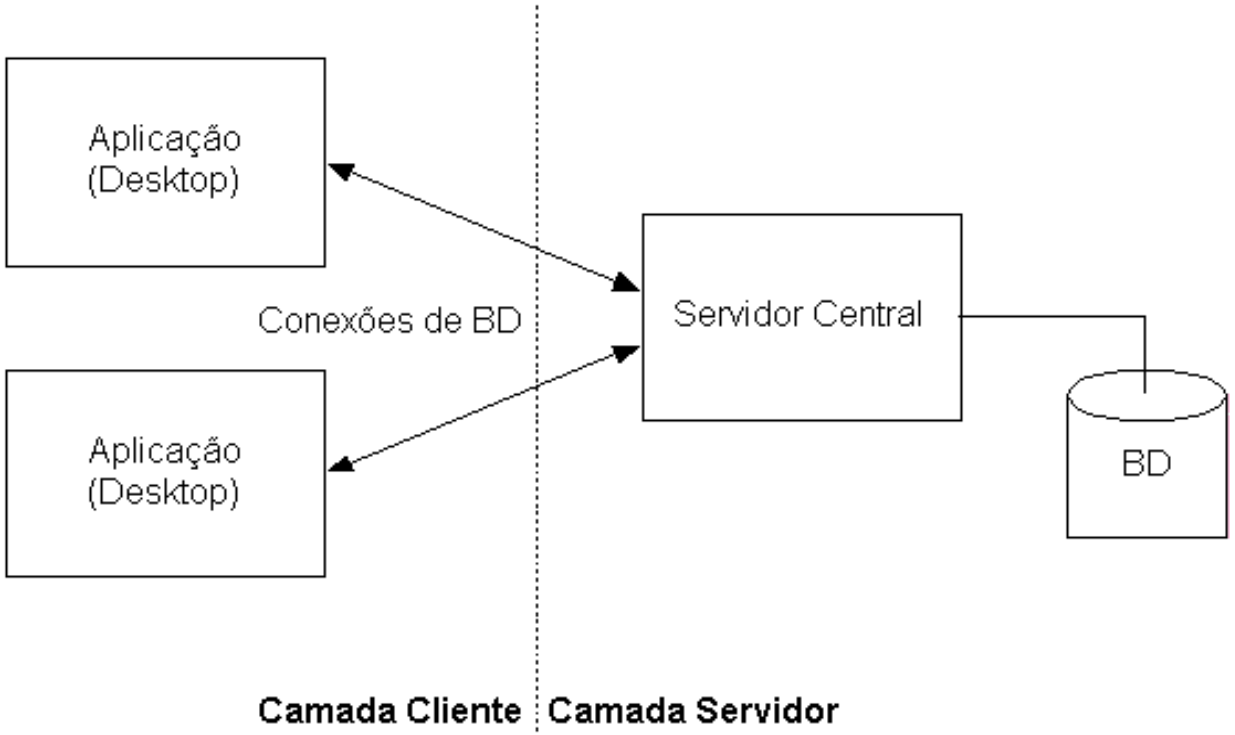
Arquitetura de Software

- Cliente-Servidor (2 Camadas)
  - Contexto
    - Sistemas distribuídos 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.



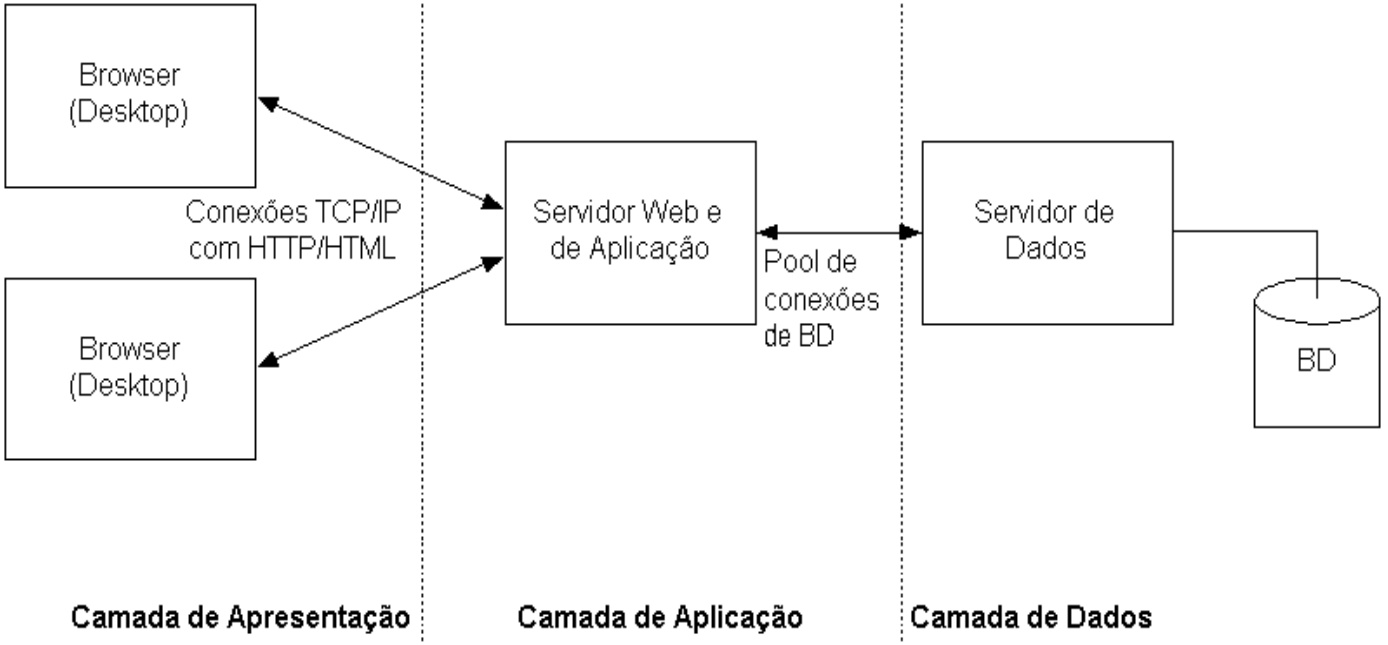
Arquitetura de Software

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



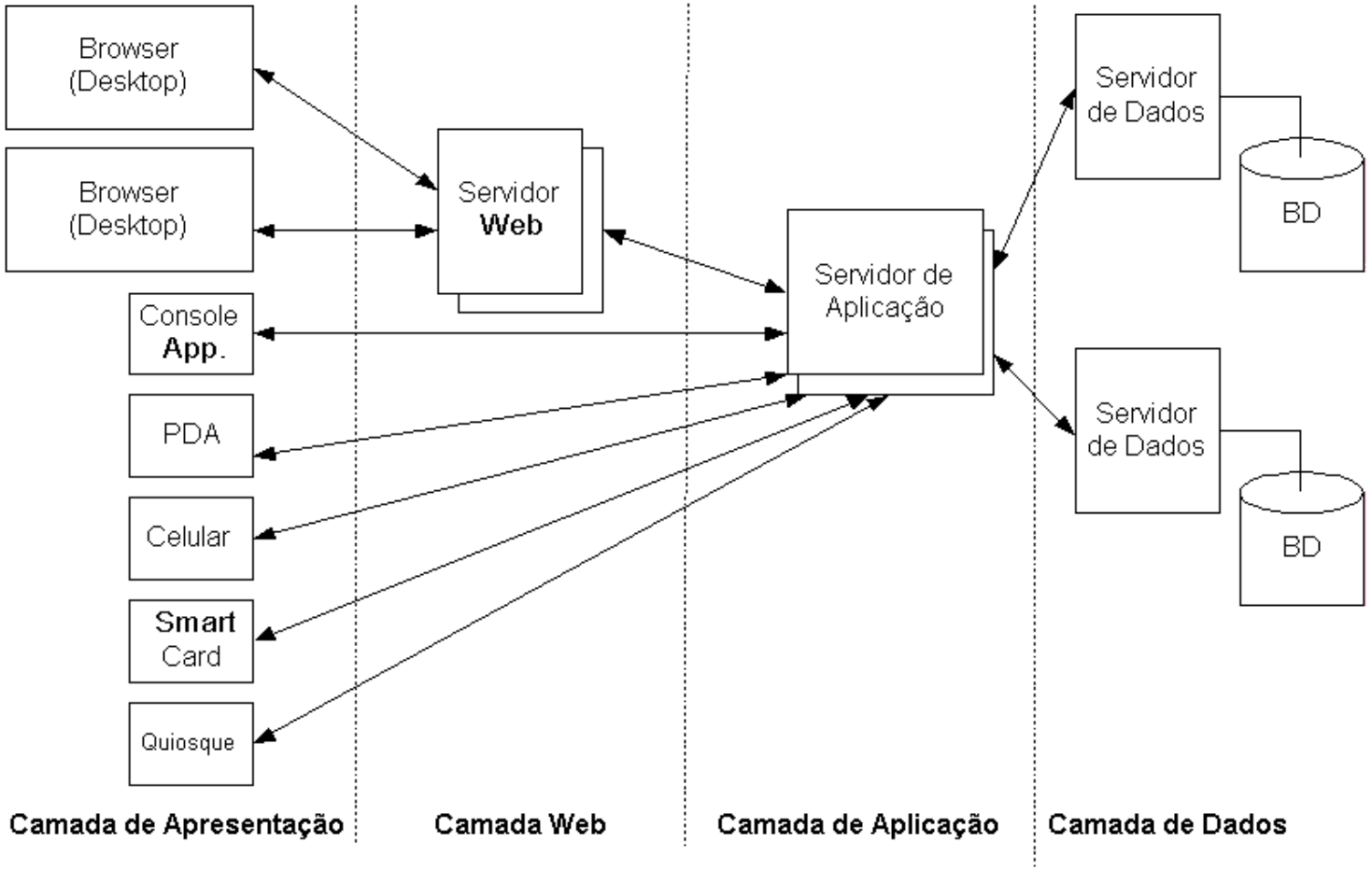
Arquitetura de Software

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



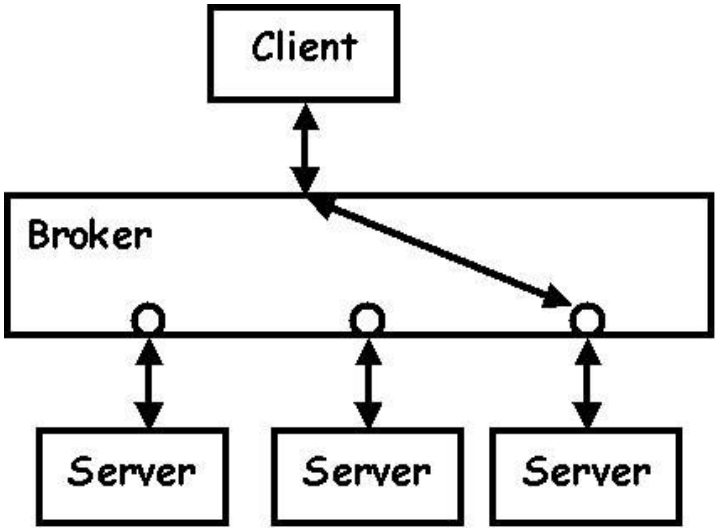
Arquitetura de Software

- 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*.

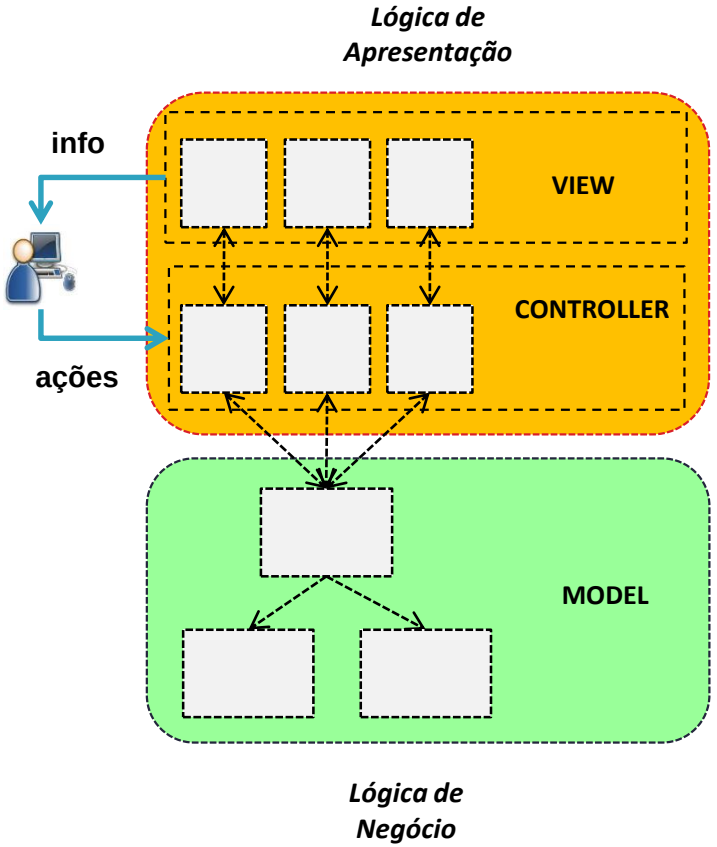


Arquitetura de Software

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

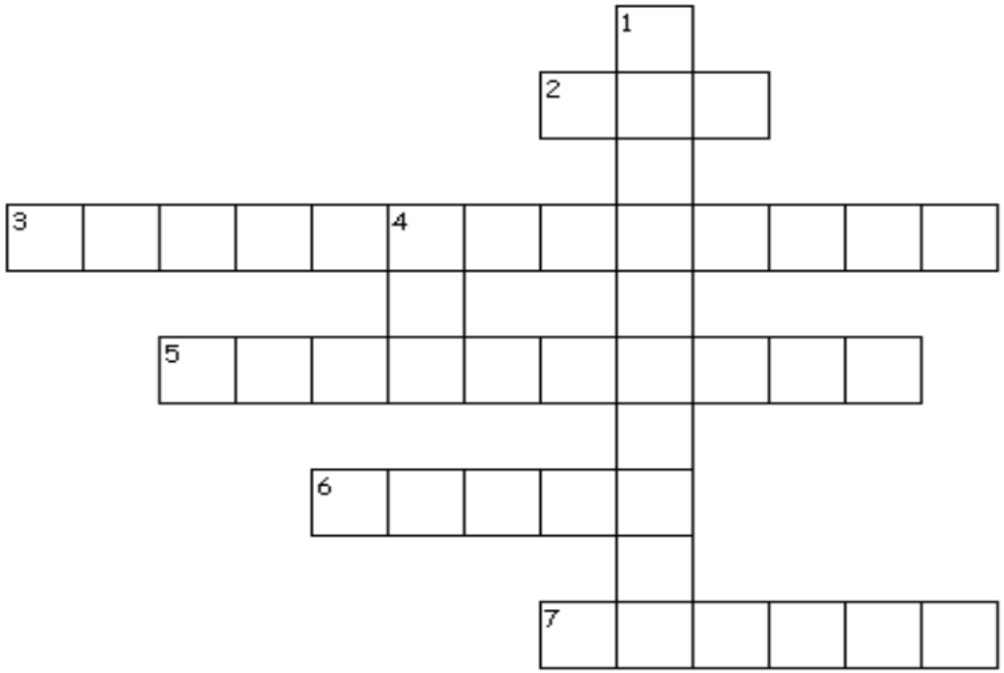
Arquitetura de Software

- MVC – *Model View Controller*
  - Contexto
    - 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

- Exercício de Fixação de Conteúdo



Across
2. Padrão de Arquiteura aplicado a webservices
3. Visão da arquitetura que define a organização dos componentes de uma aplicação segundo um padrão arquitetural
5. Padrão de Arquitetura composta por subsistemas especializados para solucionar problemas não determinísticos
6. Camada da arquitetura MVC responsável por implementar as regras de negócio da aplicação
7. Elemento da Arquitetura Orientada a Serviços que intermedia a comunicação entre clientes e servidores

Down
1. Camada da arquitetura MVC responsável por processar as requisições do usuário
4. Padrão de Arquitetura que promove a separação entre camadas visando separar a lógica de apresentação das regras de negócio