PUC
GOIÁS

Curso: Análise e Desenvolvimento de Sistemas

Estilos e padrões arquiteturais

Classificação

Invocação/Retorno (*Call/Return*)

- Programa principal/Subrotina (*Main Program/Subroutine*)
- Invocação remota de procedimento (*Remote Procedure Call - RPC*)
- Camadas (*Layered*)

Componentes independentes (*Independent Components*)

- Comunicação de processos (*Communicating Processes*)
- Baseado em eventos

Máquina virtual (*Virtual Machine*)

- Interpretador (*Interpreter*)
- Baseado em regras (*Rule-based*)

Centrado em dados (*Data-Centered*)

- Repositório (*Repository*)
- Quadro negro (*Blackboard*)

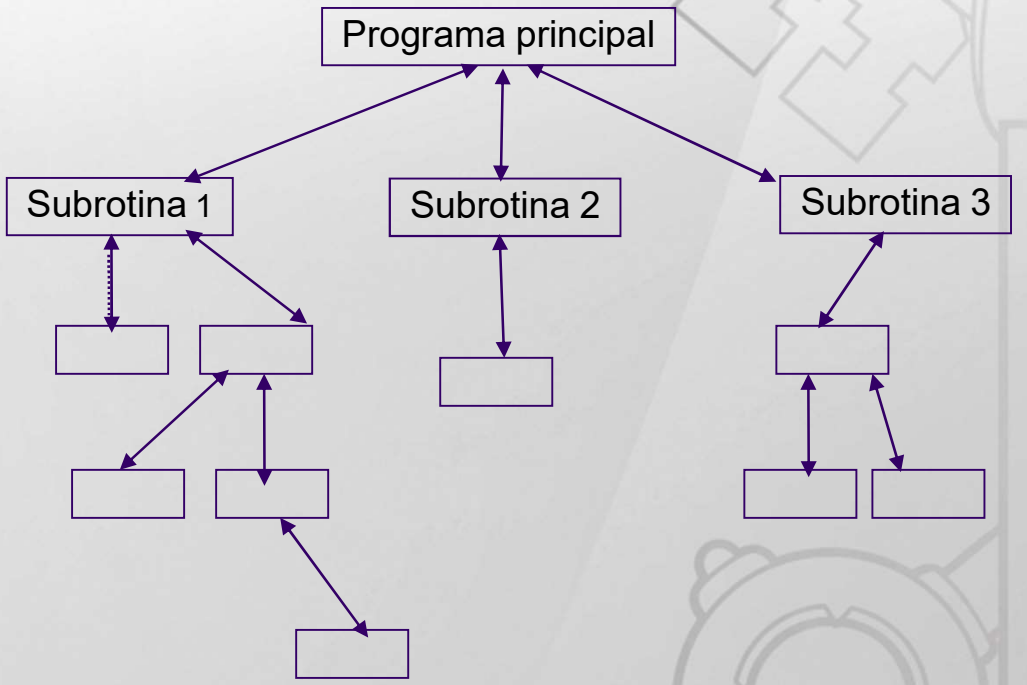
Fluxo de dados (*Data-Flow*)

- Seqüencial (*Batch Sequential*)
- Tubos e filtros (*Pipe and Filter*)

Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

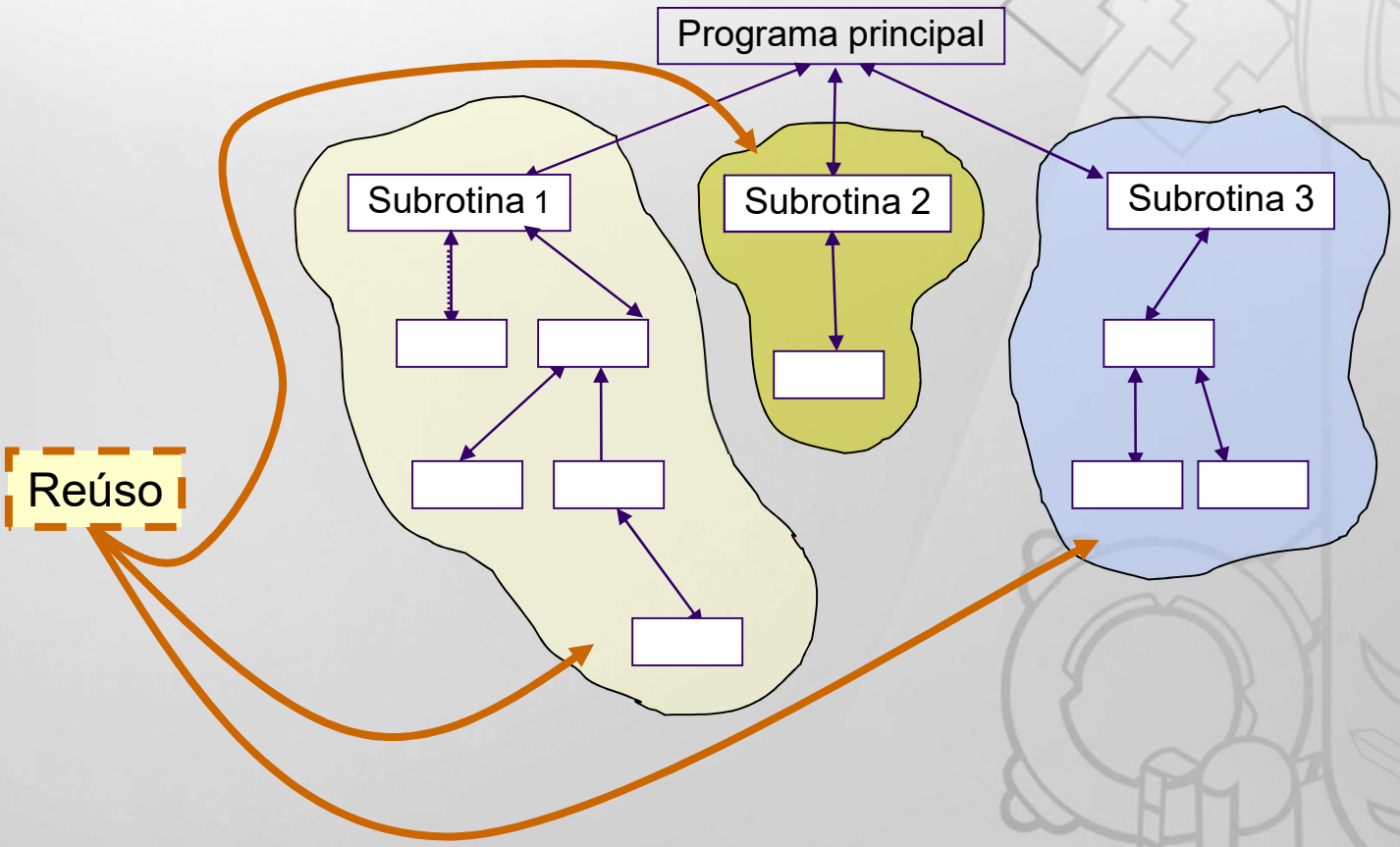
- Programa principal/Subrotina (*Main Program/Subroutine*)



Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

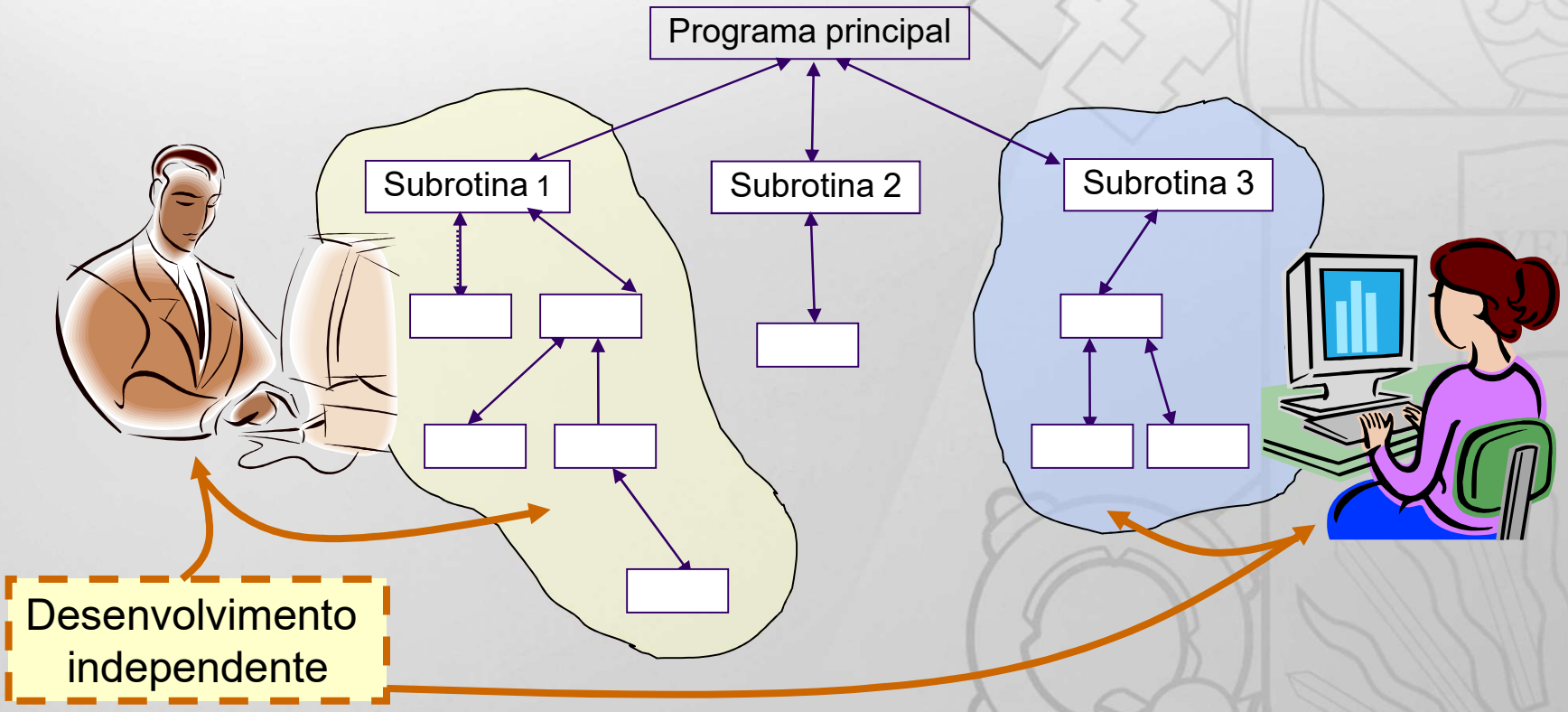
- Programa principal/Subrotina (*Main Program/Subroutine*)
  - **Objetivos**



Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

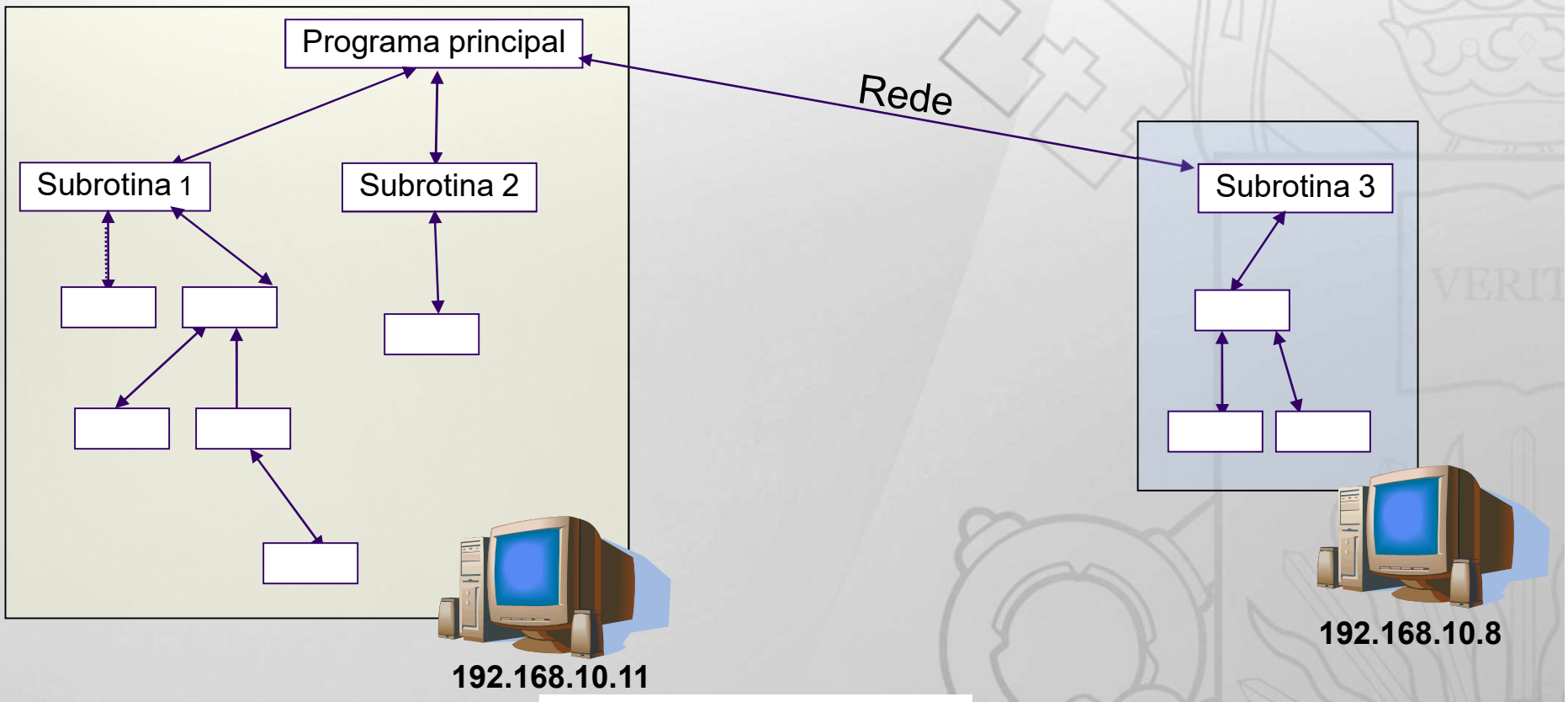
- Programa principal/Subrotina (*Main Program/Subroutine*)
  - **Objetivos**



Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

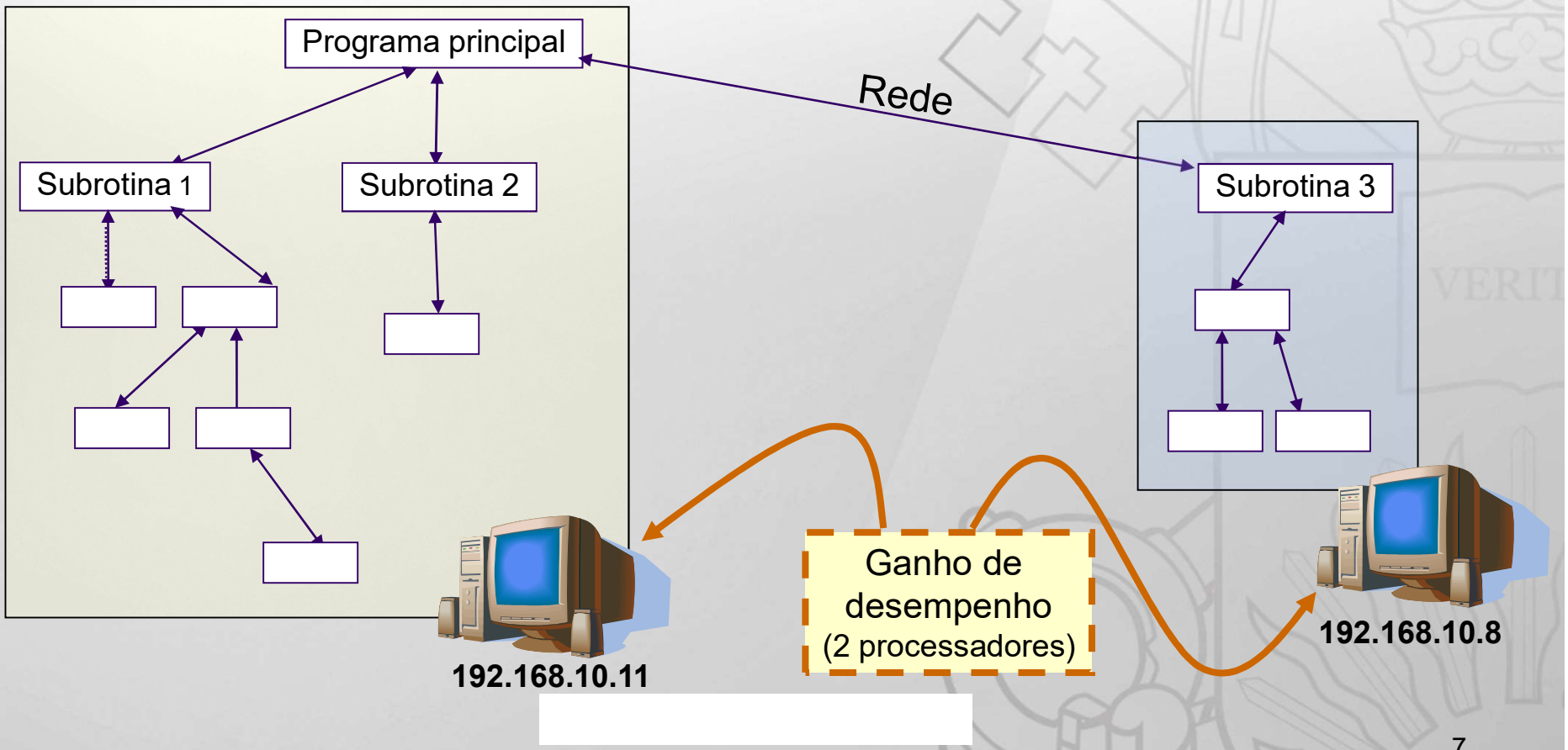
- Invocação remota de procedimento (*RPC*)



Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

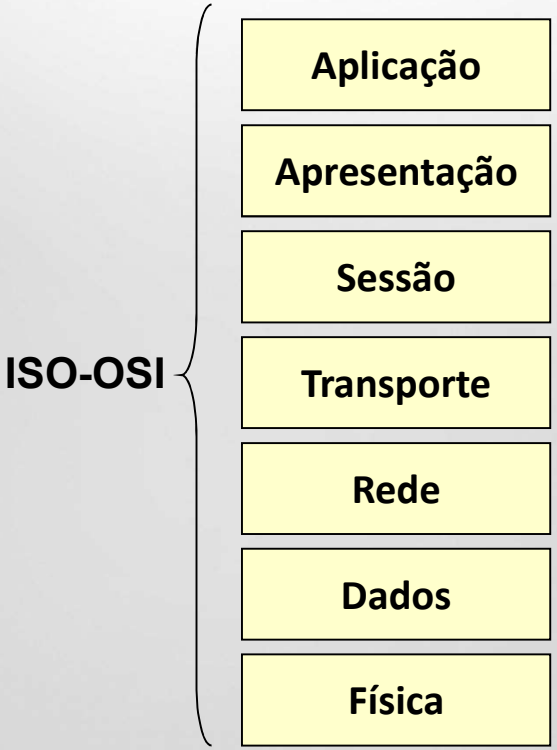
- Invocação remota de procedimento (*RPC*)

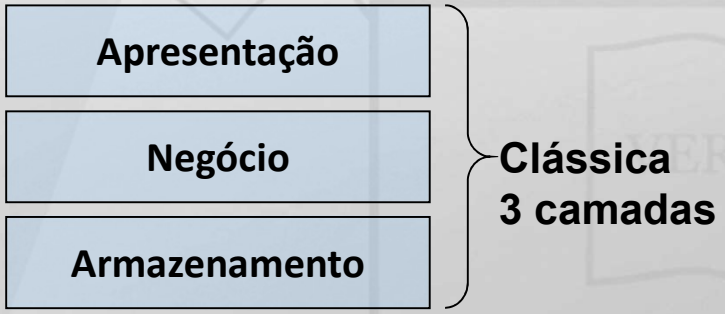


Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

- Camadas (*Layered*)





Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

- Camadas (*Layered*)
  - Camadas se comunicam apenas com outras adjacentes

Estilos e padrões arquiteturais

Invocação/Retorno (Call/Return)

- Camadas (*Layered*)
  - Alterações locais não são propagadas

Estilos e padrões arquiteturais

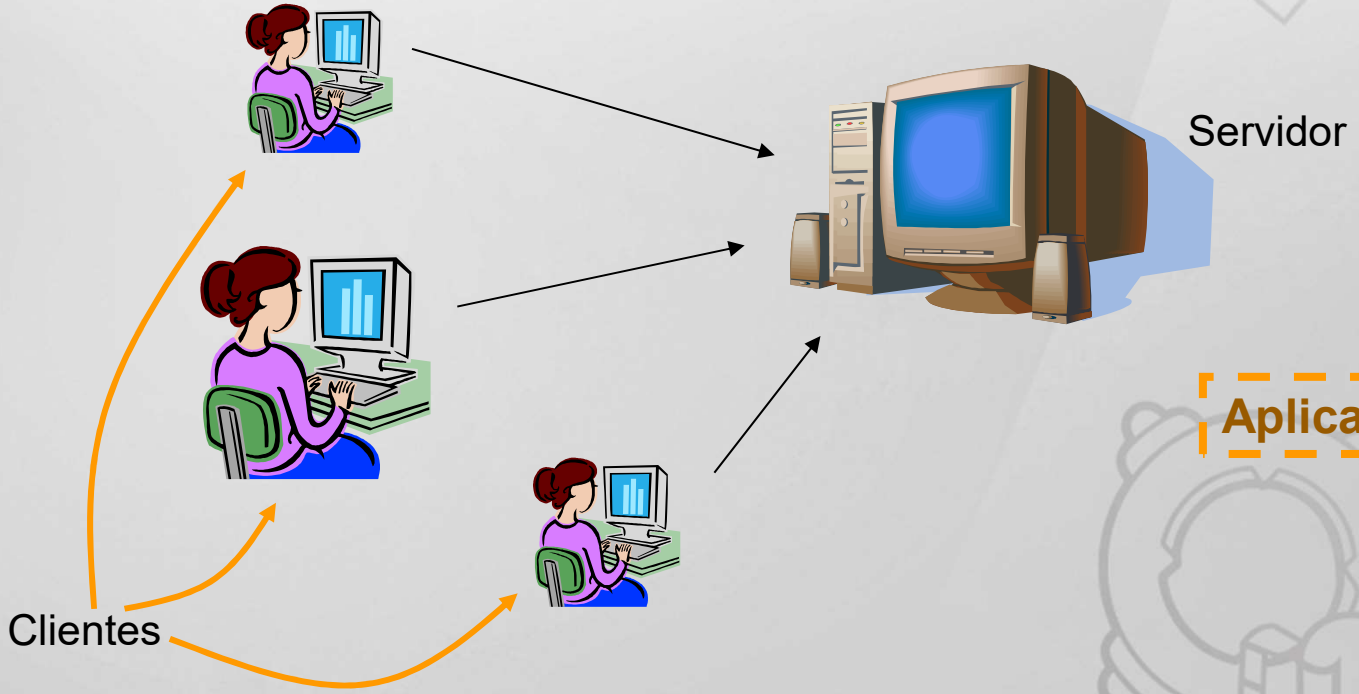
Componentes independentes

- Comunicação de processos (*Communicating Processes*)
  - Baseado na comunicação via troca de mensagens entre processos
    - Em geral, via rede
  - Cliente - Servidor
  - Ponto a ponto (*Peer to Peer* – P2P)

Estilos e padrões arquiteturais

Componentes independentes

- Comunicação de processos (*Communicating Processes*)
  - Cliente – Servidor
    - Já vimos anteriormente

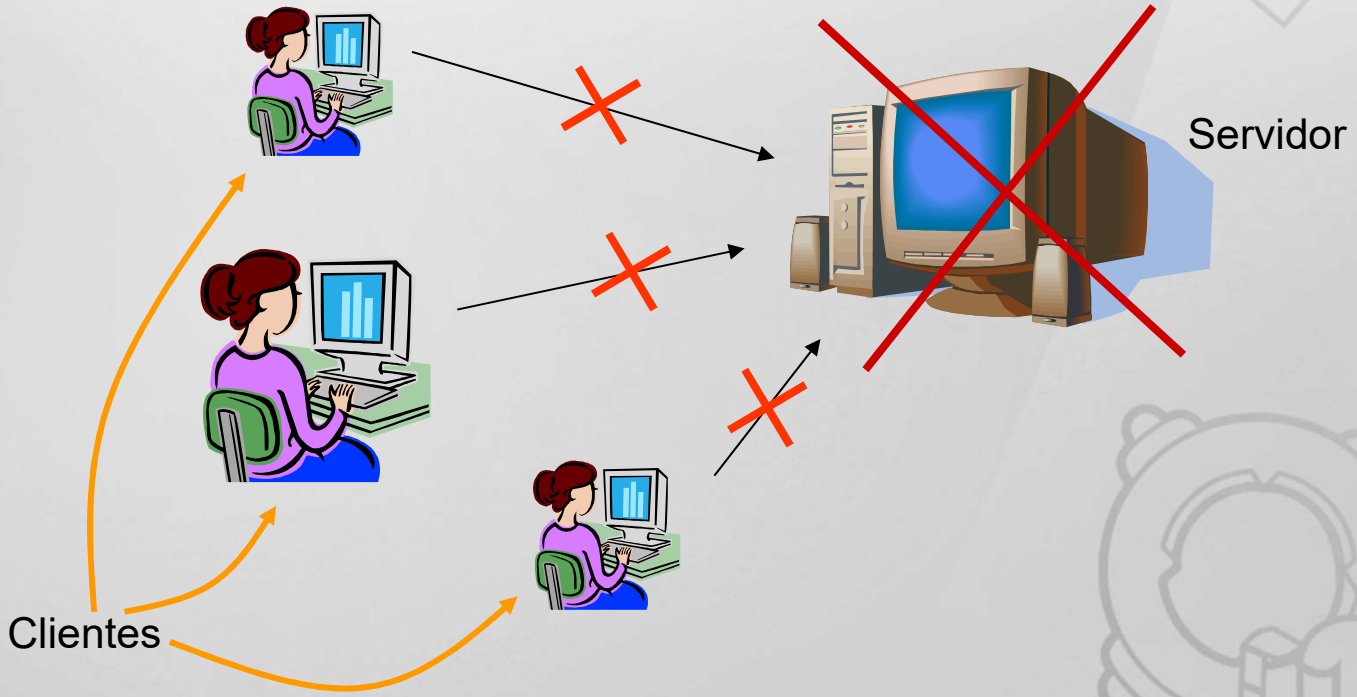


Aplicação: Internet

Estilos e padrões arquiteturais

Componentes independentes

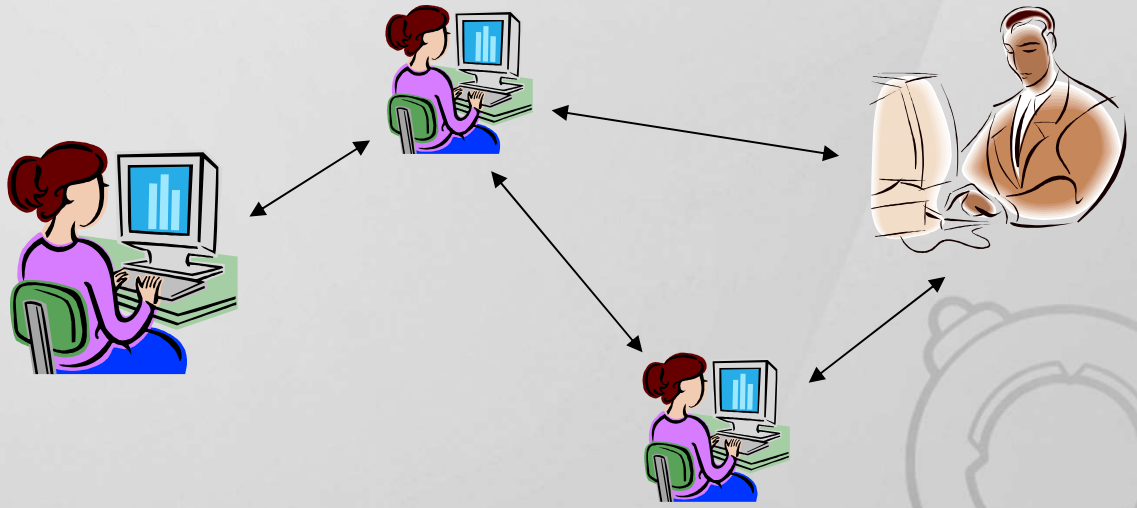
- Comunicação de processos (*Communicating Processes*)
  - Cliente – Servidor
    - Problema: servidor é ponto de falha!



Estilos e padrões arquiteturais

Componentes independentes

- Comunicação de processos (*Communicating Processes*)
  - Ponto a Ponto (P2P)
    - Não há distinção entre nós
    - Cada nó mantém seus próprios dados e endereços conhecidos
    - Cada nó é “cliente e servidor ao mesmo tempo”



Estilos e padrões arquiteturais

Componentes independentes

- Comunicação de processos (*Communicating Processes*)
  - Ponto a Ponto (P2P)
    - Exemplos de redes P2P: gnutella, freenet
    - Exemplos de aplicação: Kazaa, eMule
    - Vantagem: não há ponto de falha
    - Desvantagem: tempo de consulta

Estilos e padrões arquiteturais

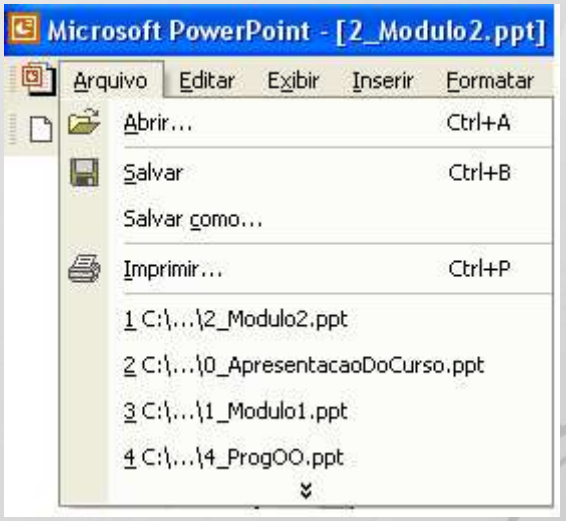
Componentes independentes

- Baseado em eventos
  - Aplicação comum
    - Interface gráfica

onKeyDown

onKeyUp

onMouseOver



onMouseReleased

onMouseClick

menuDown

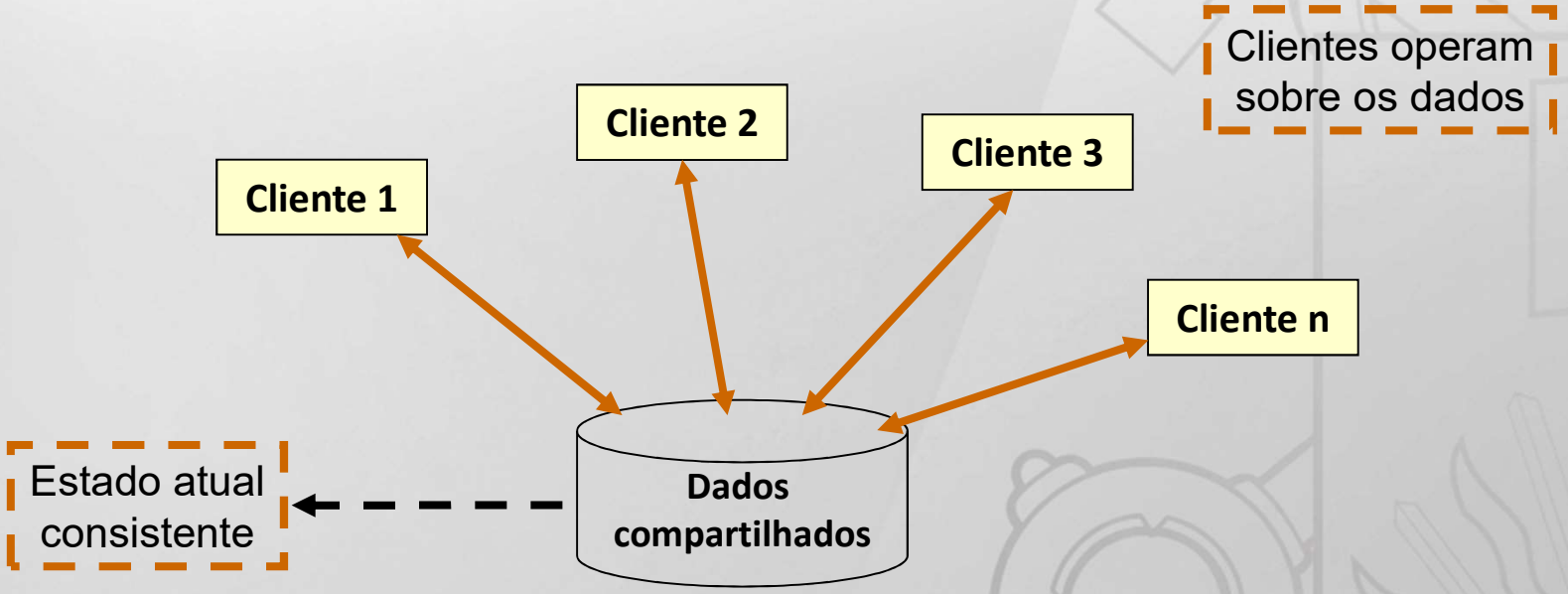
onMousePressed

onSelected

Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

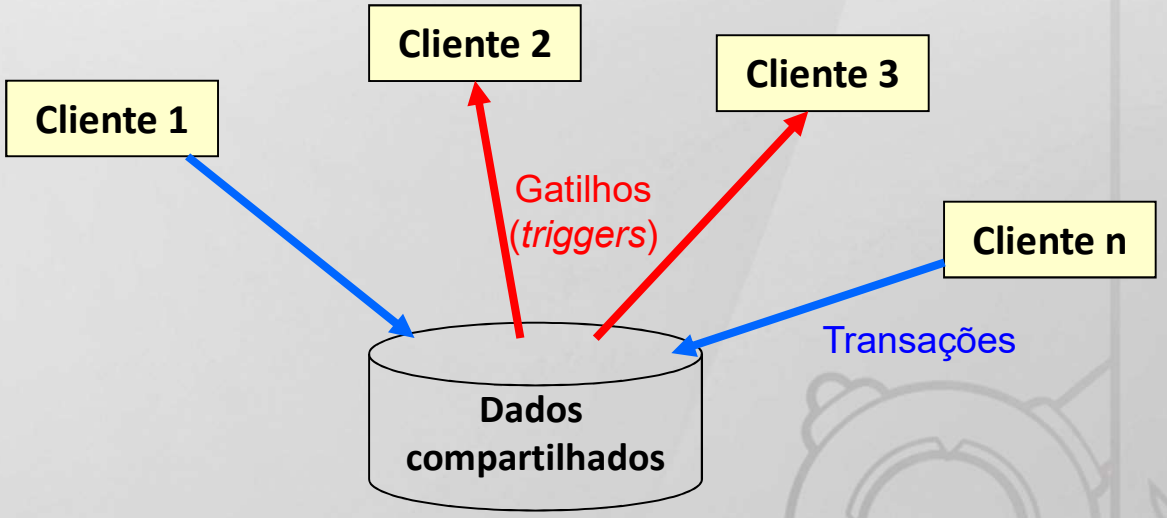
- Repositório (*Repository*)
  - Integridade, escalabilidade (novos clientes, novos dados)



Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

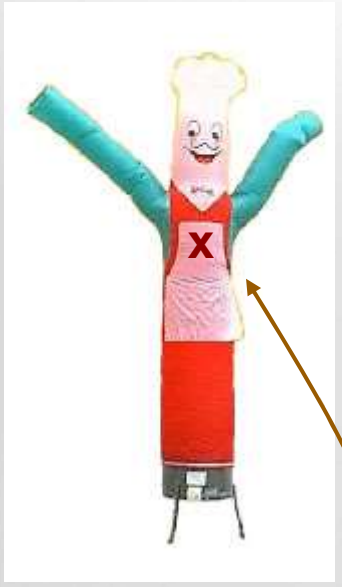
- Repositório (*Repository*)
  - Exemplo: banco de dados tradicional

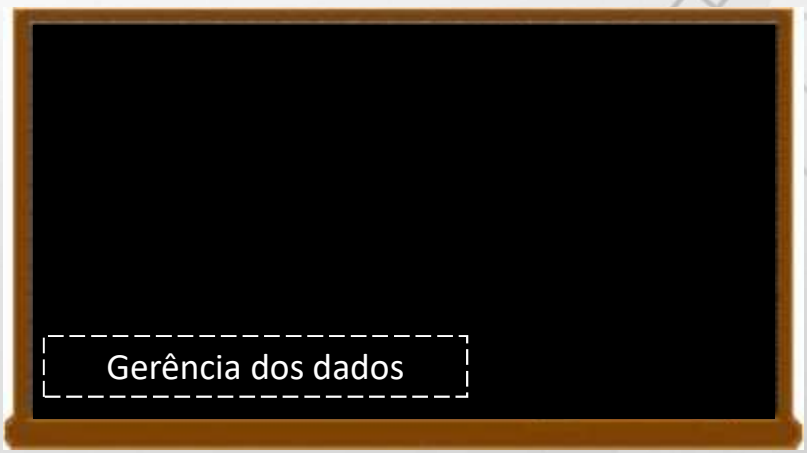


Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

- Quadro negro (*Blackboard*)



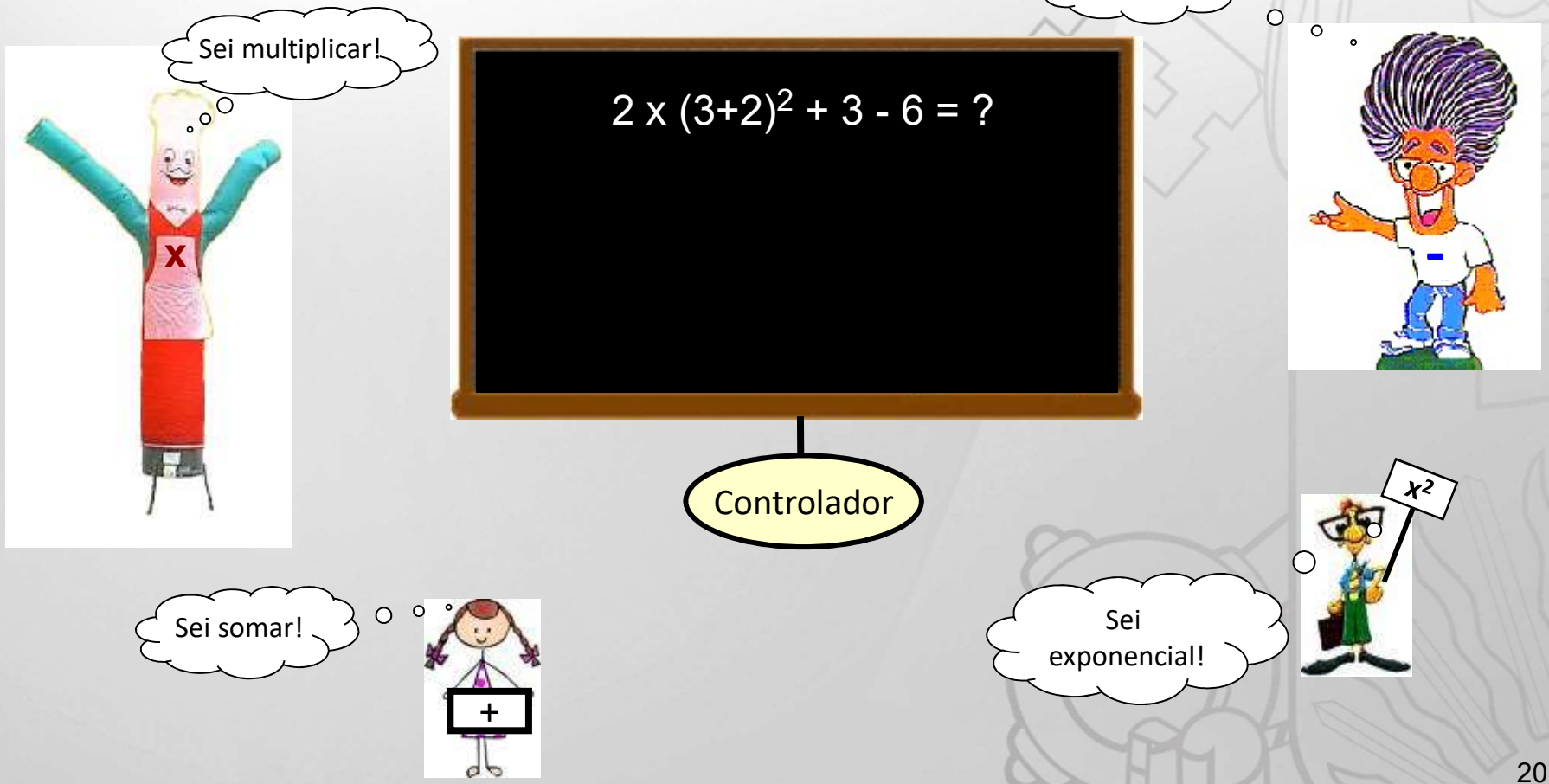


Fontes de conhecimento

Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

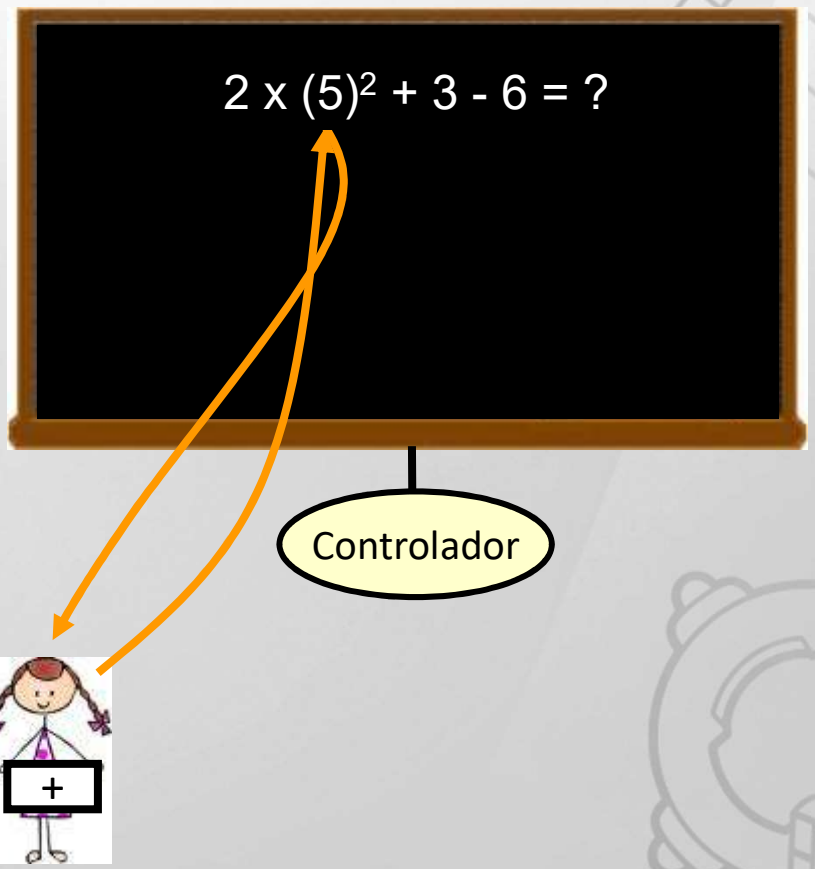
- Quadro negro (*Blackboard*)



Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

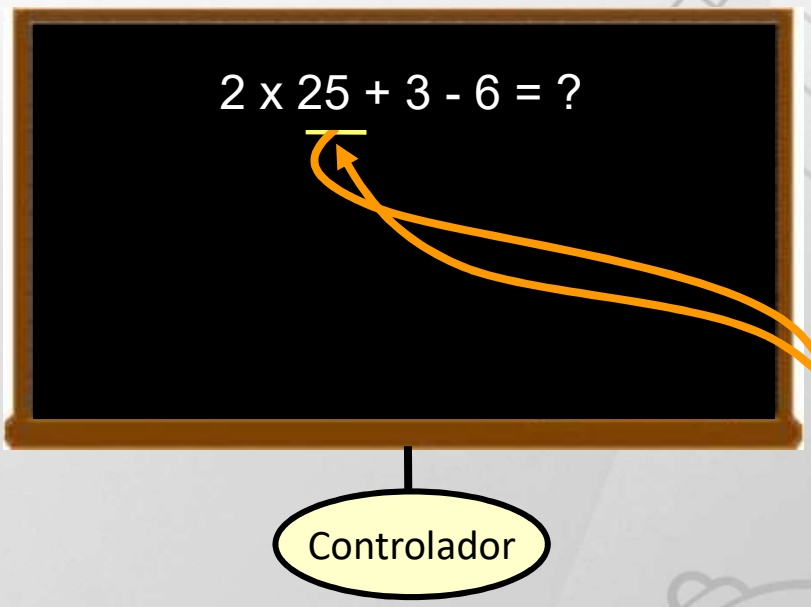
- Quadro negro (*Blackboard*)



Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

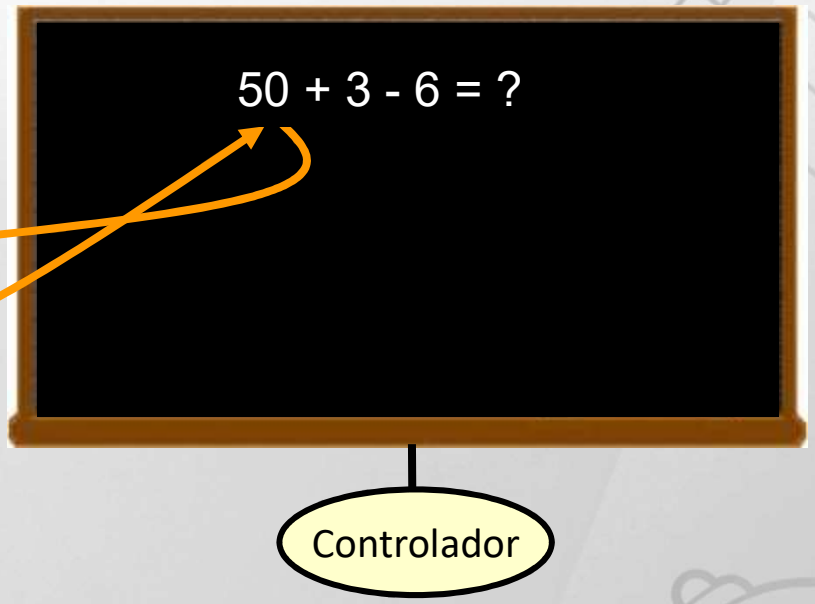
- Quadro negro (*Blackboard*)



Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

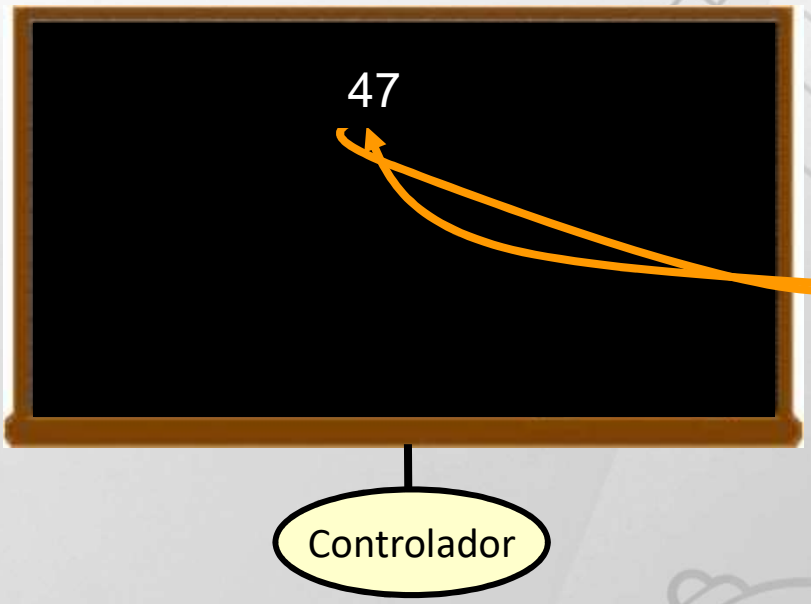
- Quadro negro (*Blackboard*)



Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

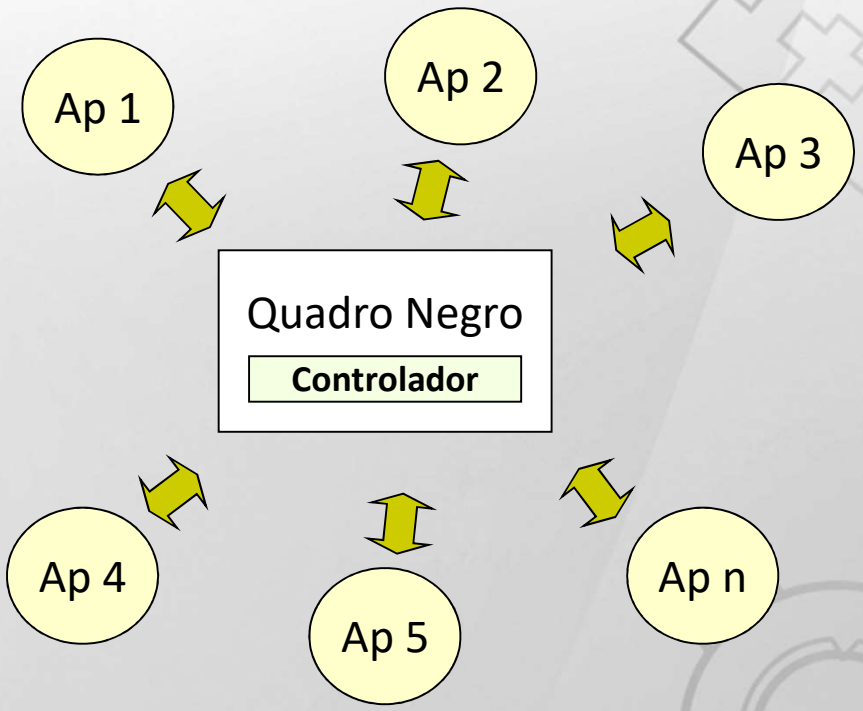
- Quadro negro (*Blackboard*)



Estilos e padrões arquiteturais

Centrado em dados (Data-centered)

- Quadro negro (*Blackboard*)



Estilos e padrões arquiteturais

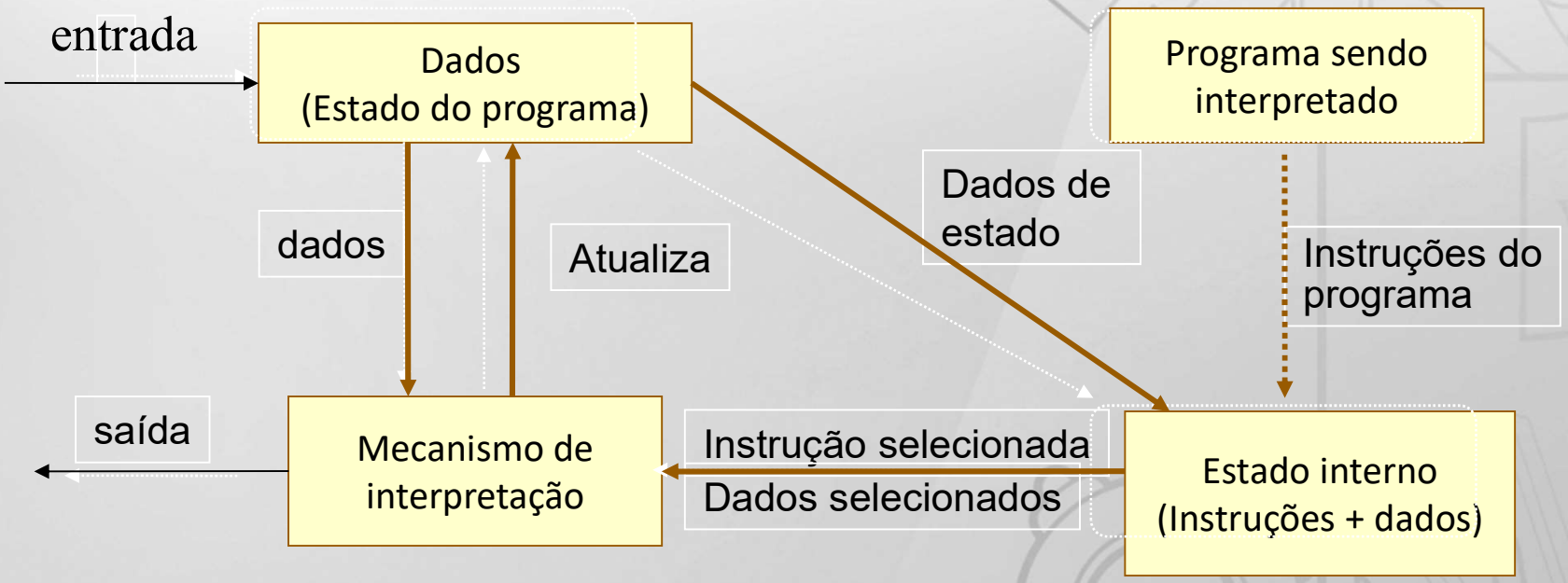
Centrado em dados (Data-centered)

- Quadro negro (*Blackboard*)
  - Sistemas complexos
    - Resolução Distribuída de Problemas - RDP
  - Aplicações independentes
    - Escalabilidade
  - Ponto de falha!!!
    - Quadro negro
  - Arquitetura usada no paradigma de *agentes*

Estilos e padrões arquiteturais

Máquina virtual (Virtual Machine)

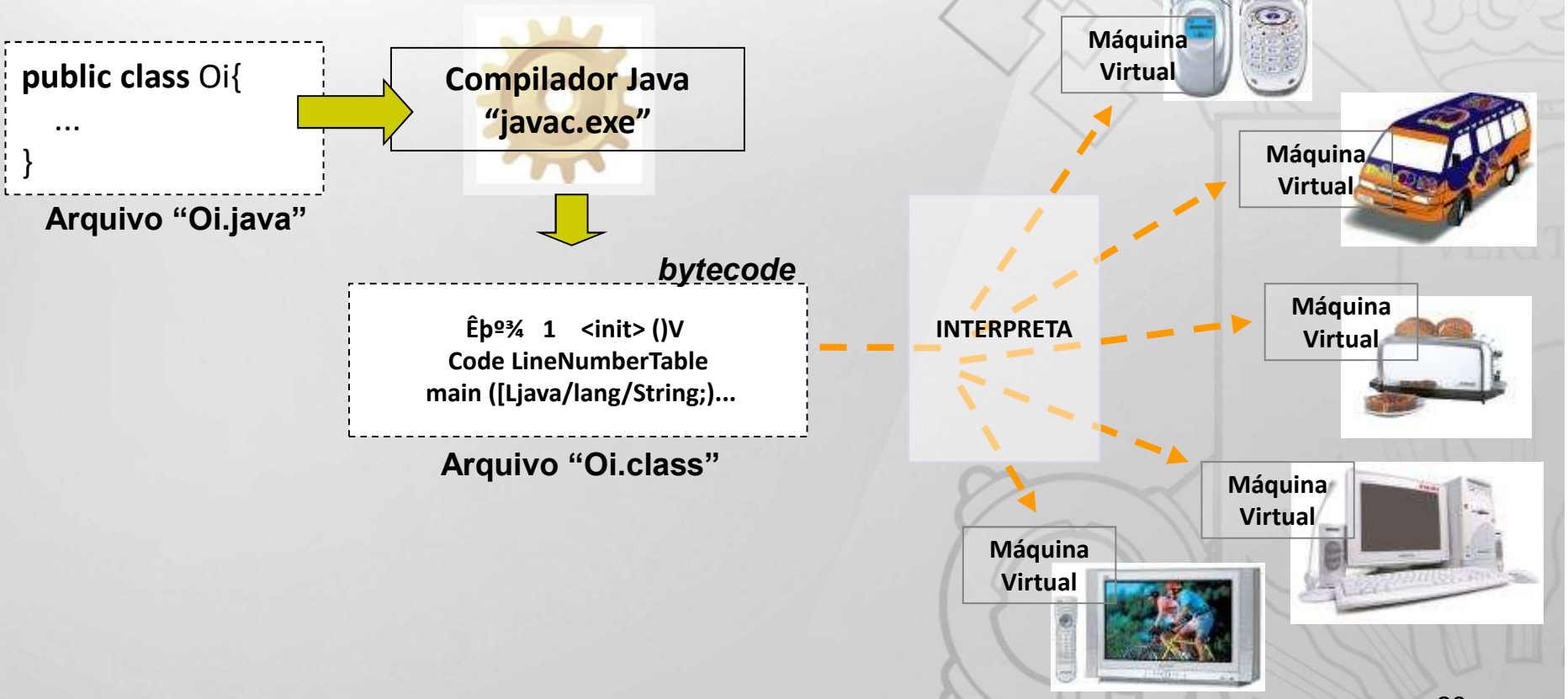
- Interpretador (*Interpreter*)
  - Simular funcionalidade não nativa para obter portabilidade



Estilos e padrões arquiteturais

Máquina virtual (Virtual Machine)

- Interpretador (*Interpreter*)
  - Exemplo: Java



Estilos e padrões arquiteturais

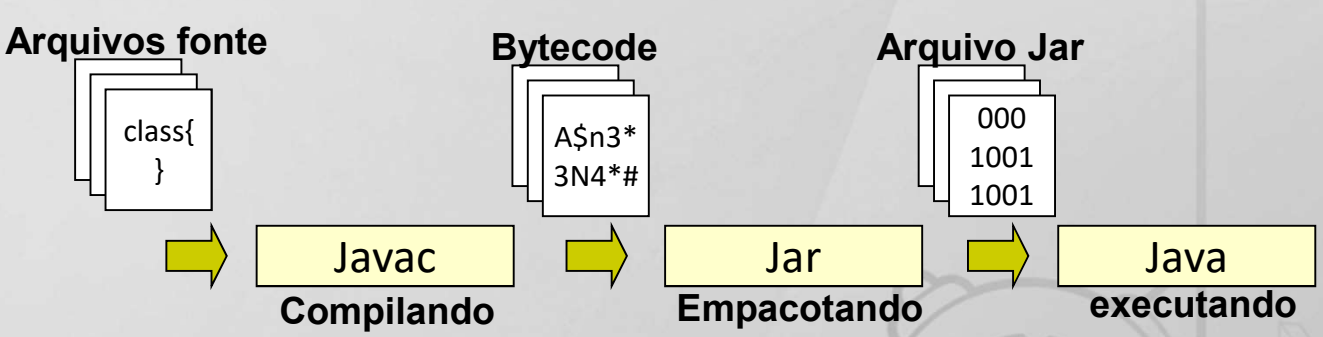
Máquina virtual (Virtual Machine)

- Interpretador (*Interpreter*)
  - Problema
    - Desempenho
  - Algumas pesquisas apontam que algumas das linguagens interpretadas já conseguem ser mais rápidas que C
    - Java, por exemplo
  - Máquina virtual nativa – Intel®

Estilos e padrões arquiteturais

Fluxo de dados (Data Flow)

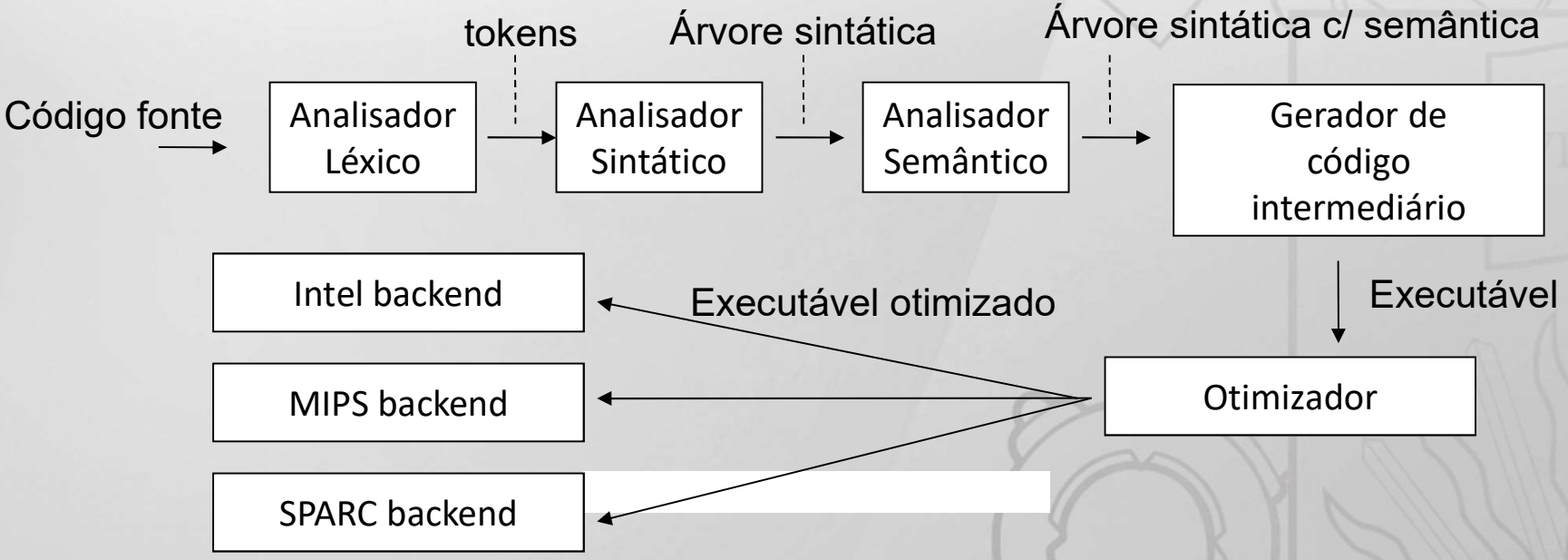
- Seqüencial (*Batch Sequential*)
  - Programas independentes executados em seqüência
    - Um após o outro
    - Dado transmitido por completo entre um programa e outro



Estilos e padrões arquiteturais

Fluxo de dados (Data Flow)

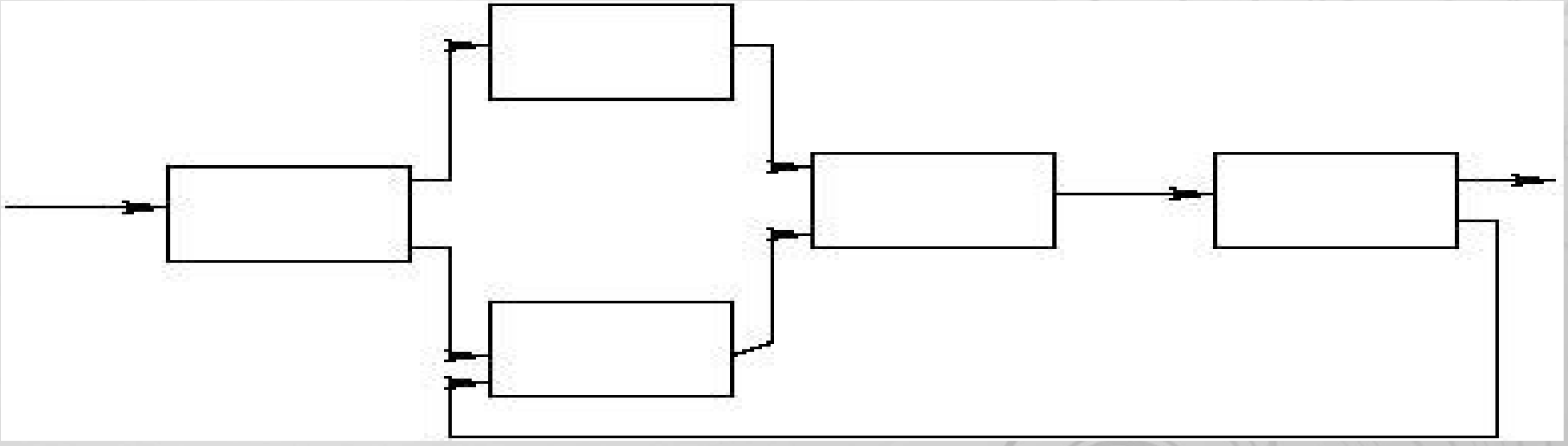
- Tubos e filtros (*Pipe and Filter*)
  - Já vimos na última aula
  - Exemplo: compilador



Estilos e padrões arquiteturais

Fluxo de dados (Data Flow)

- Tubos e filtros (*Pipe and Filter*)
  - Não precisa ser sequencial



Seleção de estilos

- Como selecionar um estilo arquitetural?

1. Identificar os principais elementos da arquitetura
2. Identificar o estilo arquitetural dominante
3. Considerar responsabilidades adicionais associadas com a escolha do estilo
4. Modificar o estilo para atingir objetivos adicionais

Seleção de estilos

- 1. Identificar os principais elementos da arquitetura
  - Cada elemento arquitetural tem um estilo arquitetural dominante que reflete as qualidades importantes que devem ser alcançadas no contexto daquele elemento
  - A escolha do estilo arquitetural dominante é baseada nos principais elementos arquiteturais
  - Os atributos de qualidade sobre cada elemento arquitetural podem acarretar a utilização ou não de um estilo

Seleção de estilos

- 2. Identificar o estilo arquitetural dominante
  - O estilo dominante pode ser modificado para alcançar objetivos particulares
  - Se nenhum estilo conhecido parece ser apropriado, o arquiteto deve projetar e documentar um novo estilo
  - As decisões sobre escolhas baseadas em atributos de qualidade dentro de um estilo devem ser documentadas

Fonte:

Carnegie Mellon
Software Engineering Institute

Seleção de estilos

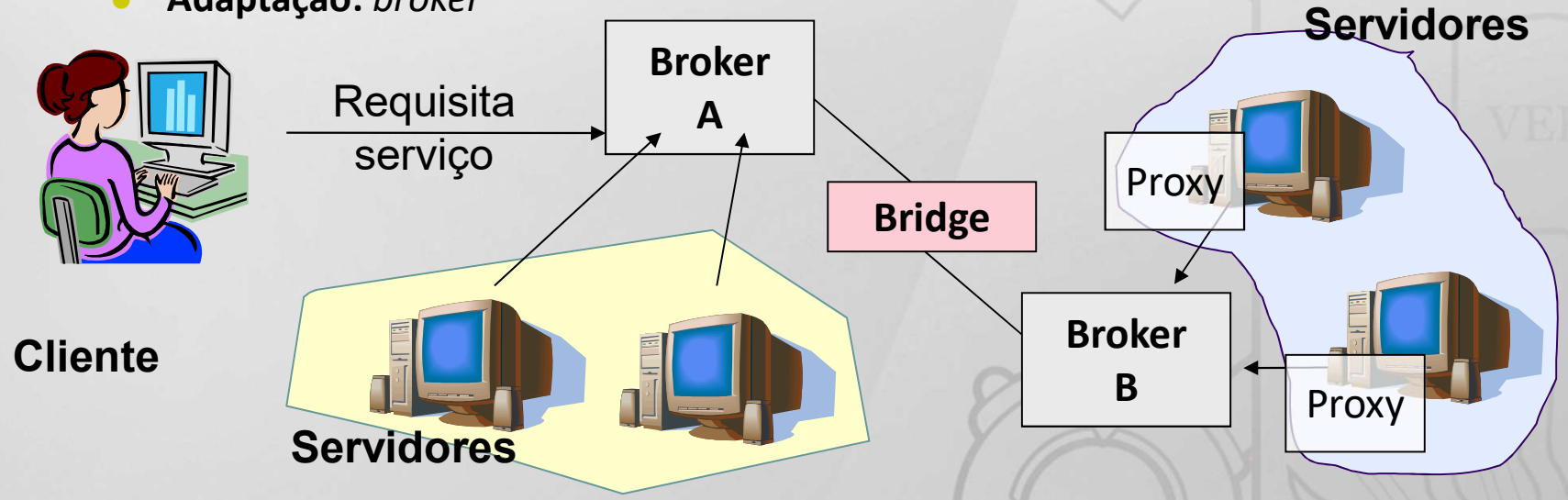
- 3. Considerar responsabilidades adicionais associadas com a escolha do estilo
  - A escolha de um estilo arquitetural introduzirá responsabilidades adicionais
  - Por exemplo:
    - Se o estilo é “Quadro negro”, então deve-se gerenciar os mecanismos para o controle do quadro negro
    - Se o estilo é “cliente-servidor”, deve-se gerenciar os protocolos de interação
  - Responsabilidades adicionais devem ser atribuídas a elementos arquiteturais existentes ou a novos elementos criados para este fim

Fonte:

Carnegie Mellon Software Engineering Institute

Seleção de estilos

- 4. Modificar o estilo para atingir objetivos adicionais
  - Pode-se alterar o estilo arquitetural caso este necessite ser adaptado devido a atributos de qualidade ou até mesmo funcionalidade
  - Exemplo: cliente-servidor
    - **Adaptação:** *broker*



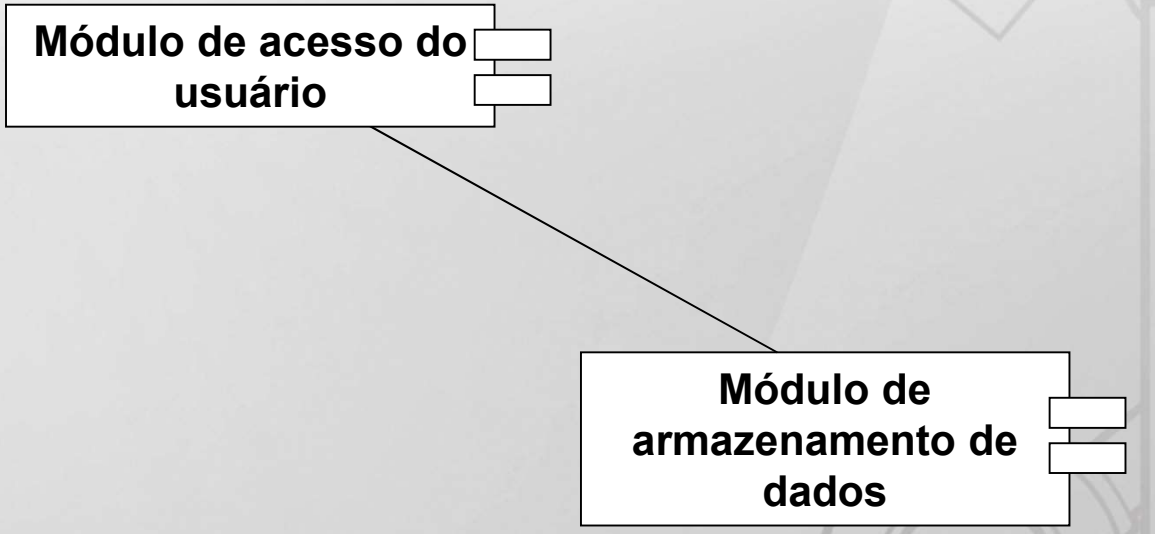
Fonte:

Carnegie Mellon Software Engineering Institute

Seleção de estilos

Exemplo

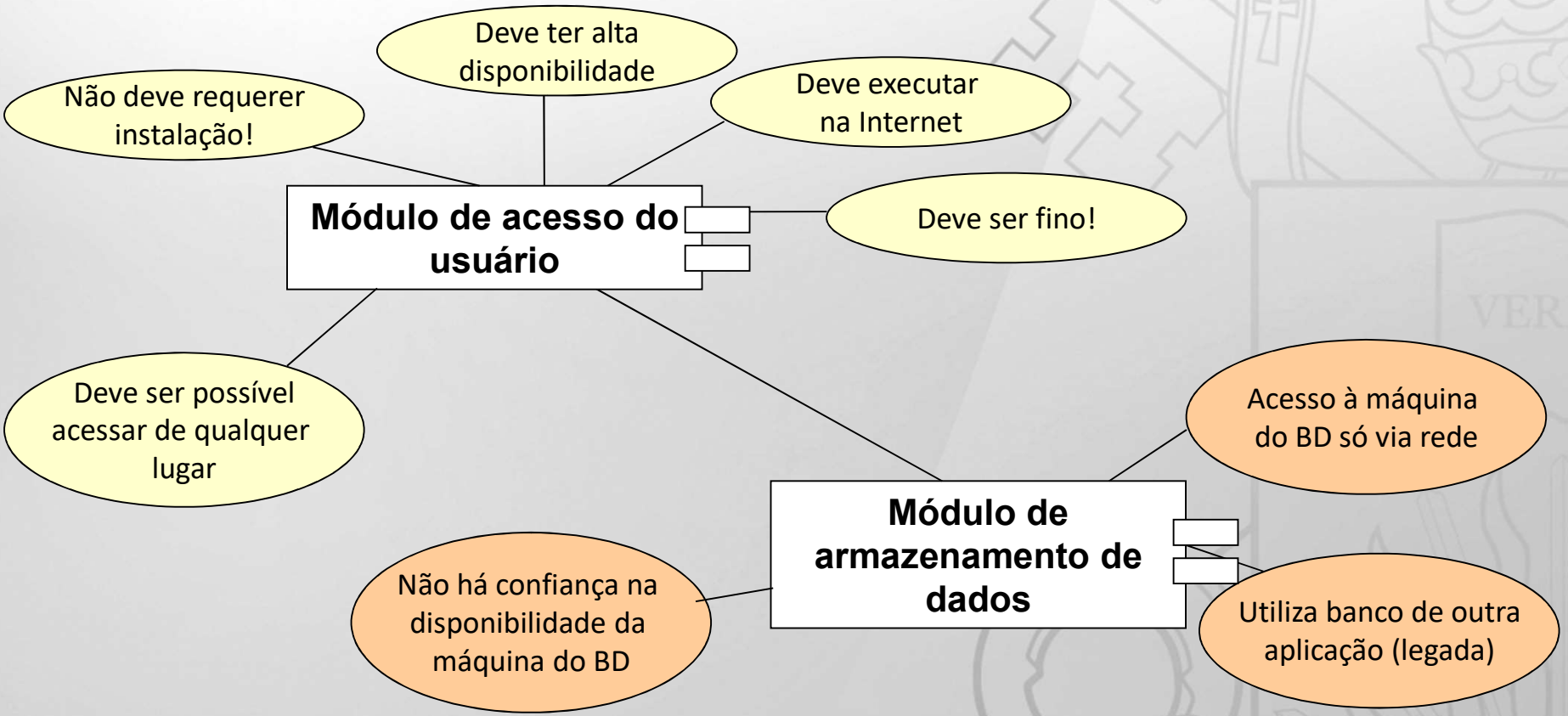
- 1. Identificar os principais elementos da arquitetura
  - Sistema: acadêmico



Seleção de estilos

Exemplo

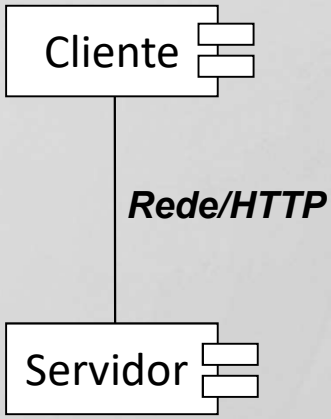
- 2. Identificar o estilo arquitetural dominante



Seleção de estilos

Exemplo

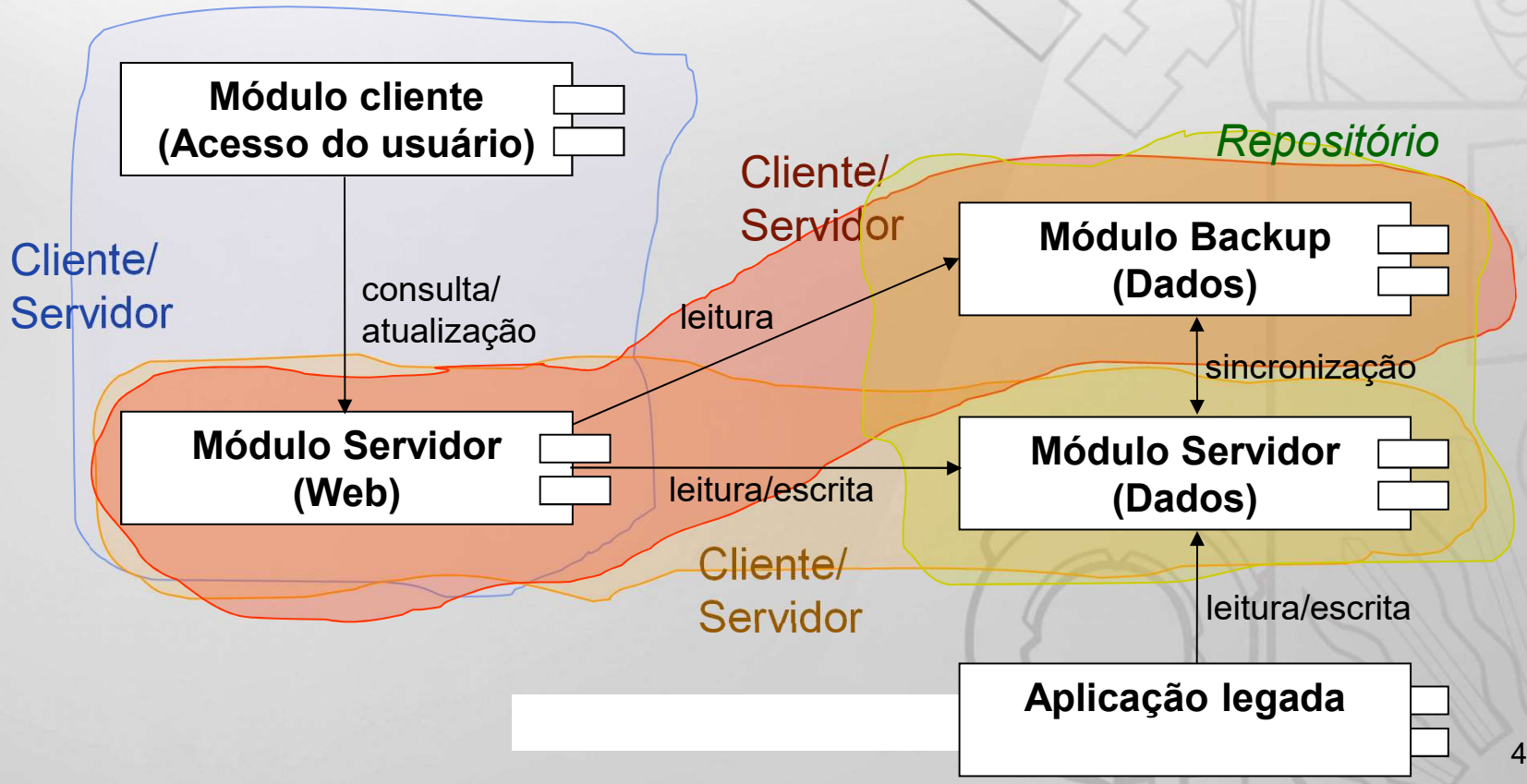
- 3. Considerar responsabilidades adicionais associadas com a escolha do estilo
  - Estilo dominante escolhido: Cliente-servidor
  - Estilo secundário: Repositório
  - Responsabilidades: considerar protocolo de comunicação



Seleção de estilos

Exemplo

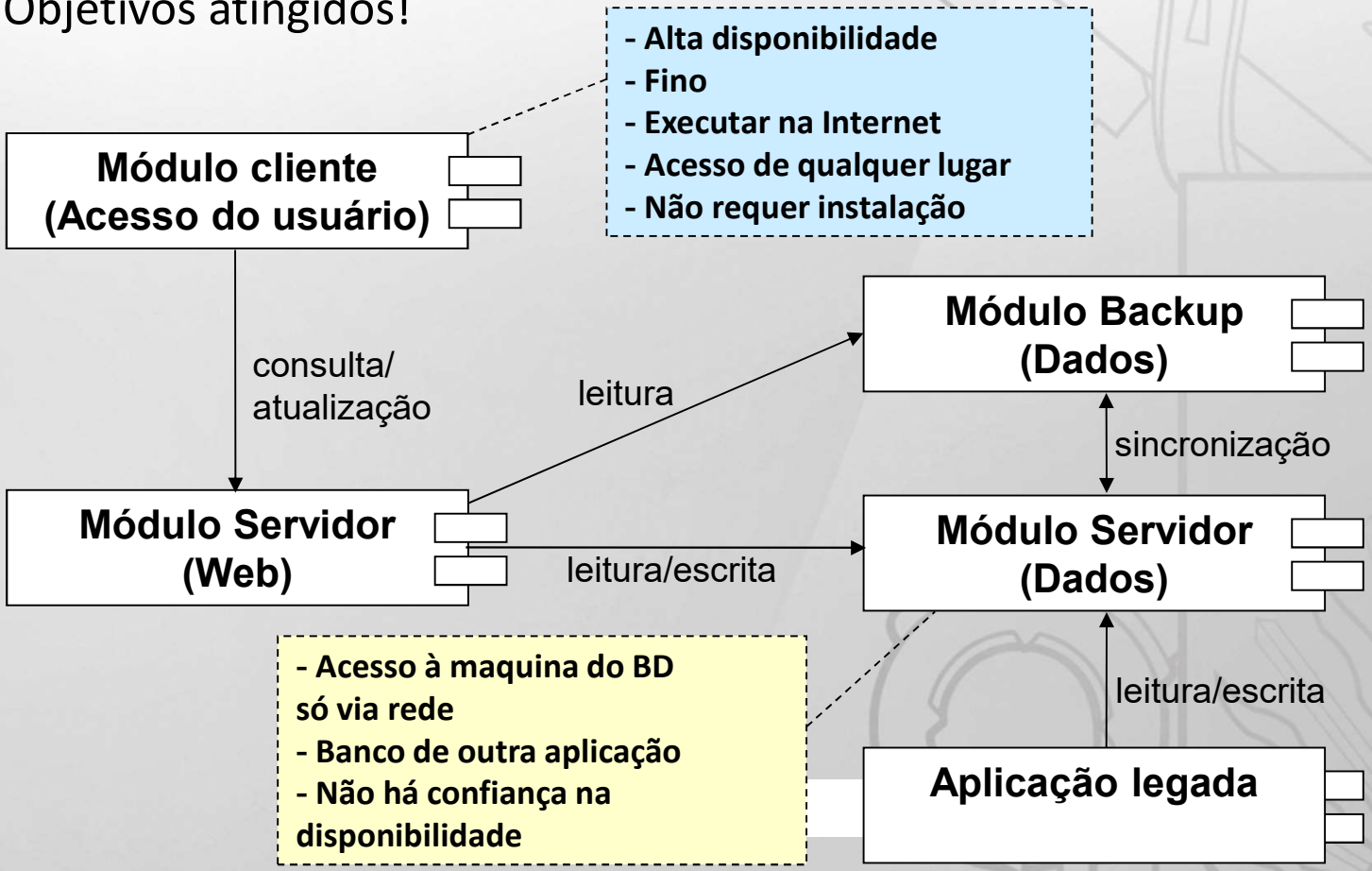
- 4. Modificar o estilo para atingir objetivos adicionais
  - Cliente servidor de 2 camadas e repositório com *backup*



Seleção de estilos

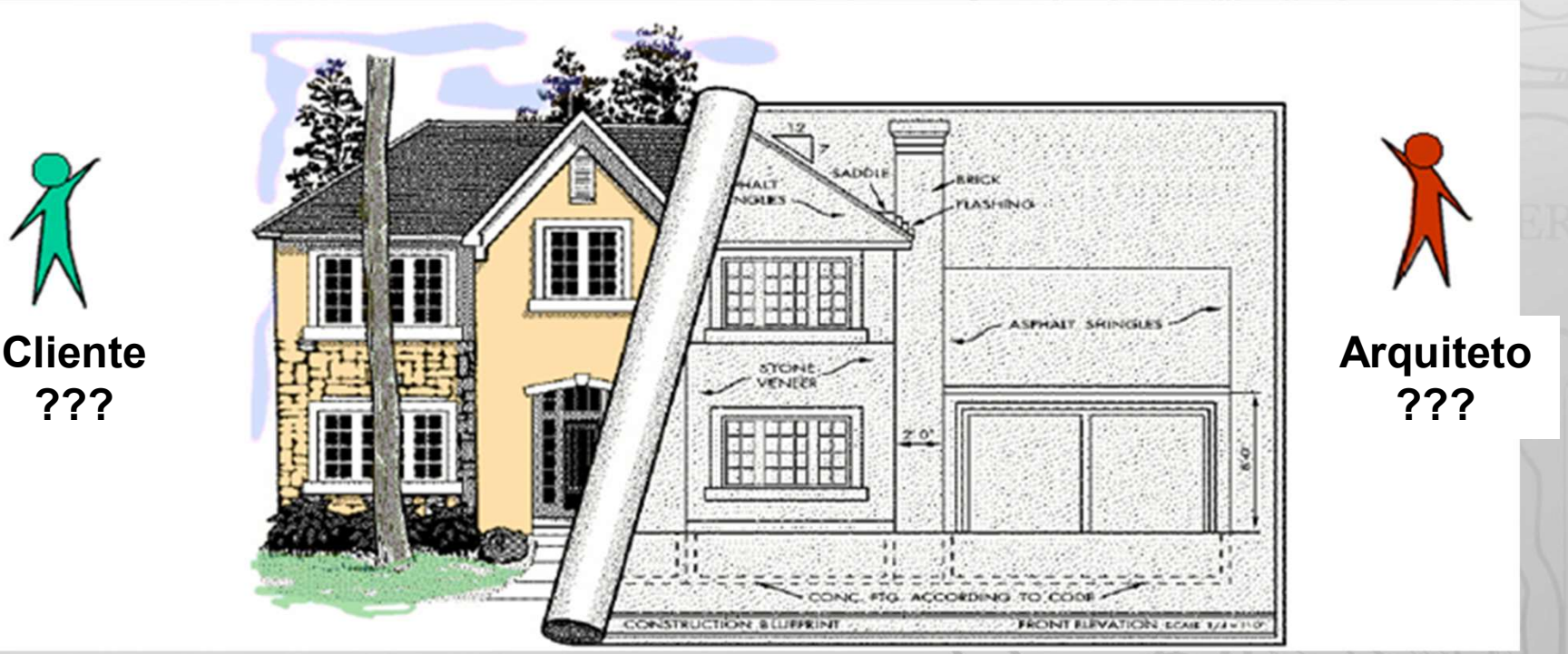
Exemplo

- 4. Modificar o estilo para atingir objetivos adicionais
  - Objetivos atingidos!



Seleção de visões

- Quais são as visões arquiteturais relevantes para o sistema sendo desenvolvido?



Seleção de visões

- Três passos:

1. Produza uma tabela de visões
2. Combine visões
3. Priorize visões

Seleção de visões

1. Produza uma tabela de visões
   - De acordo com as características do sistema

Stakeholder	Lógica	Processo	Desenvolvimento	Física
Gerente	d	v	v	d
Desenvolvedor	v	v	d	v
Testador	v	v	d	
Cliente	v			
Usuário final	v			
Analista	d			
Arquiteto	d	d	d	d

Legenda:

d = informação bem detalhada
a = alguns detalhes
v = visão geral

Seleção de visões

- 2. Combine visões
  - Considerando que para cada visão apresentada anteriormente, tem-se um ou mais modelos...
  - ... ficaria impraticável criar um modelo na perspectiva de cada uma das visões definidas anteriormente
  - Procure visões na tabela que requeiram apenas uma "visão geral" e com poucos *stakeholders* envolvidos
    - Lógica: 4 v
    - Física: 3 *stakeholders*
  - Os modelos gerados para esta visão podem ser simplificados

Seleção de visões

- 2. Combine visões
  - A visão de processo poderia ser combinada à de desenvolvimento (*Estereótipos*)

Stakeholder	Lógica	Processo	Desenvolvimento	Física
Gerente	d	v	v	d
Desenvolvedor	v	v	d	v
Testador	v	v	d	
Cliente	v			
Usuário final	v			
Analista	d			
Arquiteto	d	d	d	d

Legenda:

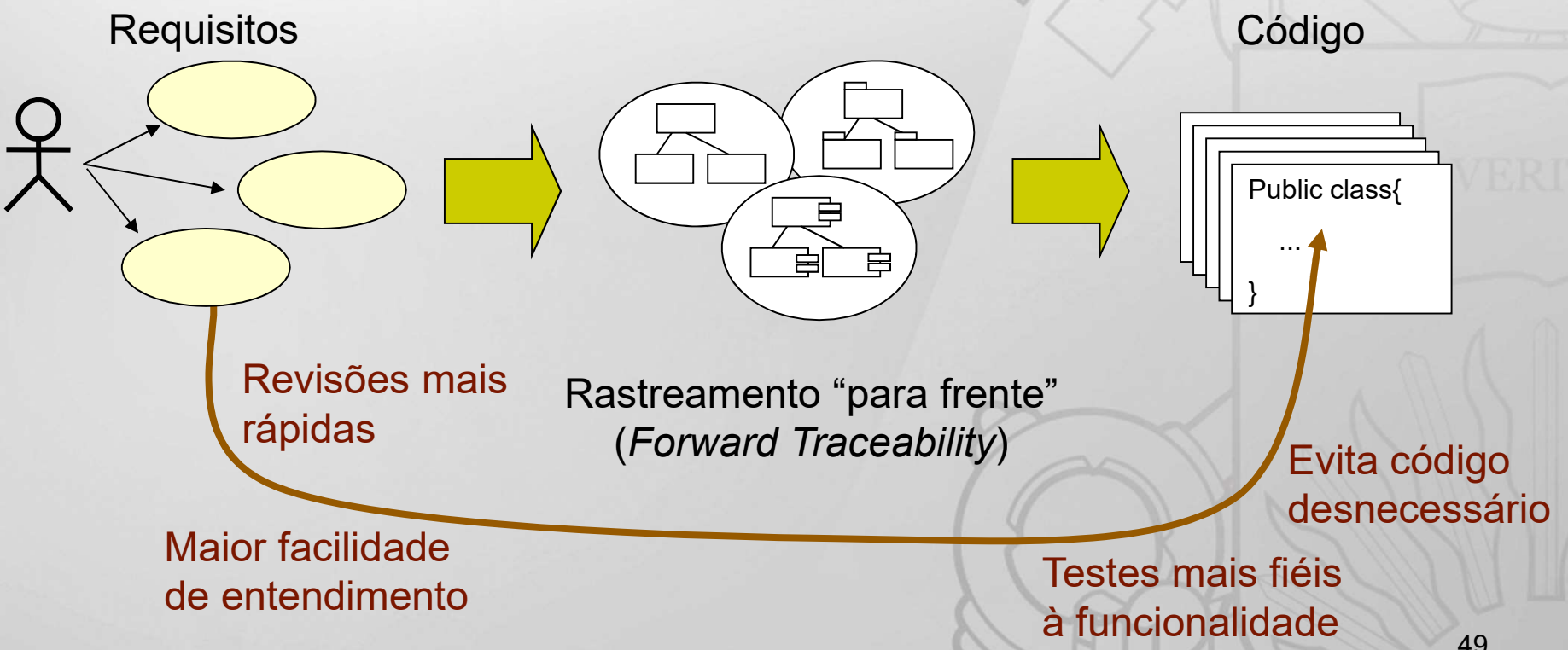
d = informação bem detalhada
a = alguns detalhes
v = visão geral

Seleção de visões

- 3. Priorize visões
  - Uma vez definidas as visões, deve-se estabelecer uma ordem de prioridade
  - Sempre começar uma nova visão após terminar outra
  - Exemplo de ordem:
    - Visão lógica
    - Visão física
    - Visão de desenvolvimento

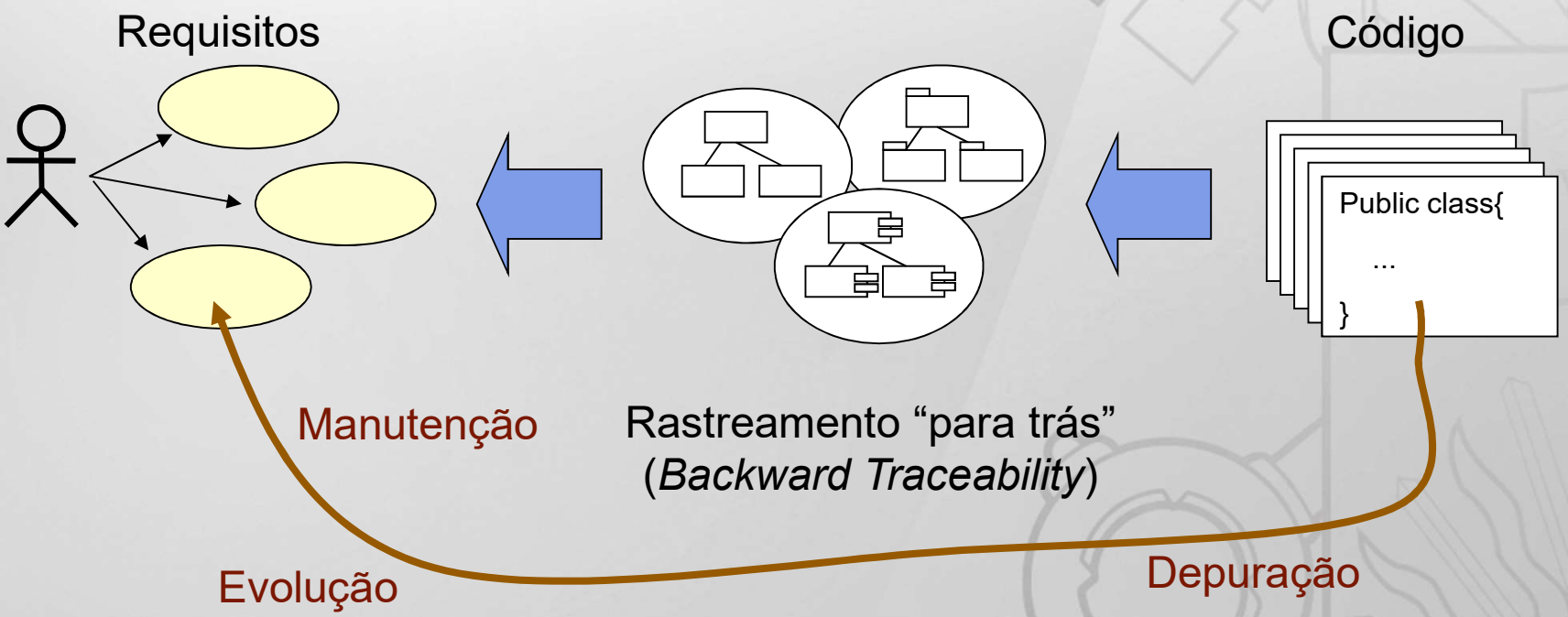
Rastreabilidade bidirecional

- Rastreamento: requisitos ⇨ código e vice-versa durante o ciclo de desenvolvimento do software



Rastreabilidade bidirecional

- Rastreamento: requisitos ⇨ código e vice-versa durante o ciclo de desenvolvimento do software



Rastreabilidade bidirecional

- Apontada como grande “arma” para o desenvolvimento...
  - CMMi (*Capability Maturity Model Integration*)
    - *Software Engineering Institute* – SEI
  - Manter rastreabilidade bidirecional do desenvolvimento é essencial para um processo de software de sucesso

Até a próxima Aula!!