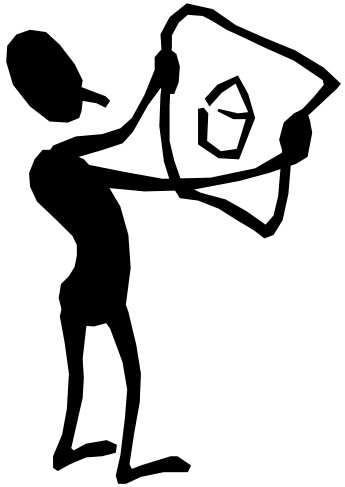
Projeto de Software

Engenharia de Software

Simone do Rocio Senger de Souza

srocio@icmc.usp.br



Engenharia de Software

1- Definição ("o que")
- **Análise do sistema**
- **Planejamento do projeto de software**
- **Análise de requisitos**

2- Desenvolvimento ("como")
- ***Projeto de software***
- **Codificação**
- **Testes**

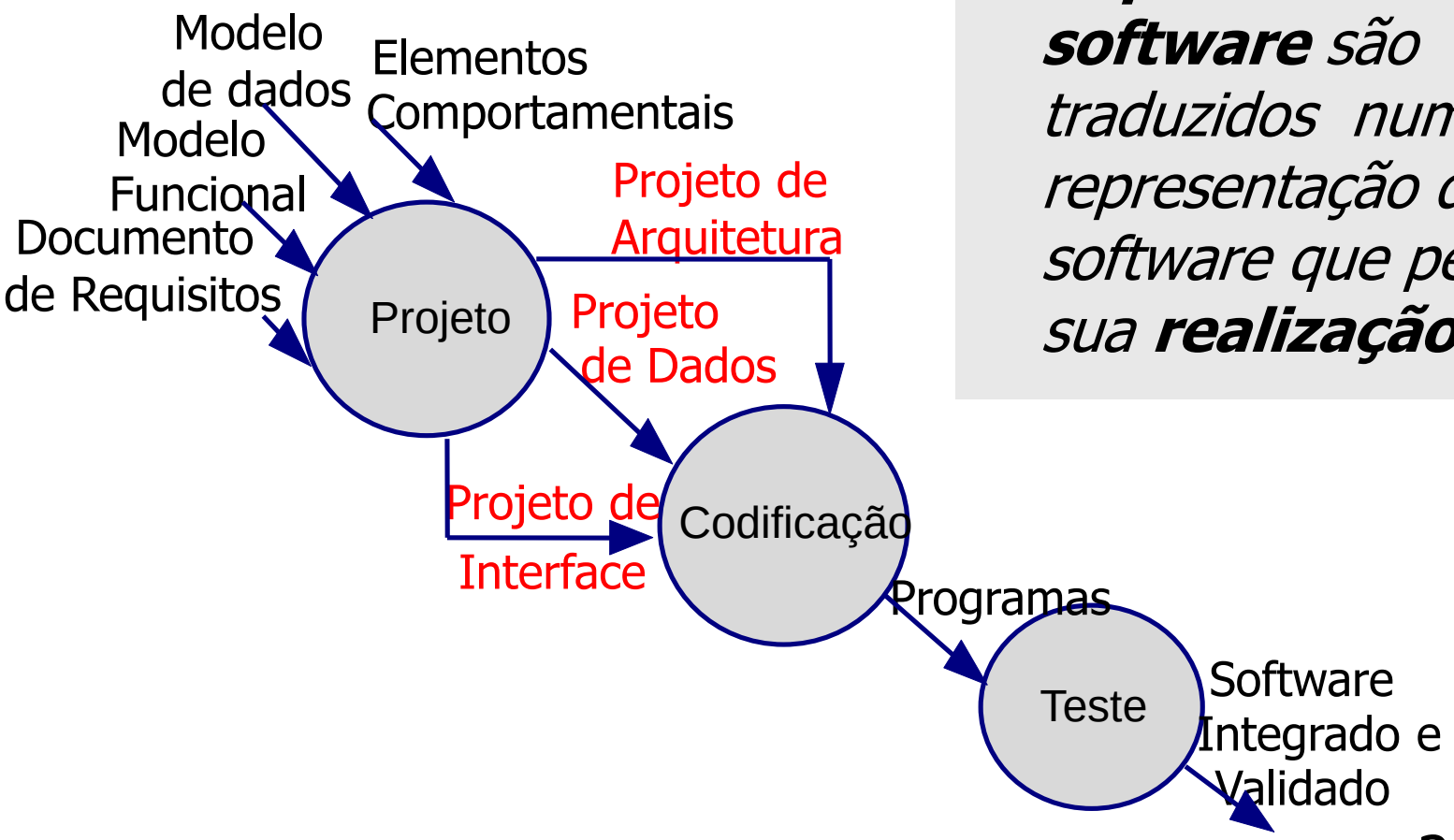
3- Manutenção ("alterações")

Atividades Guarda-Chuva

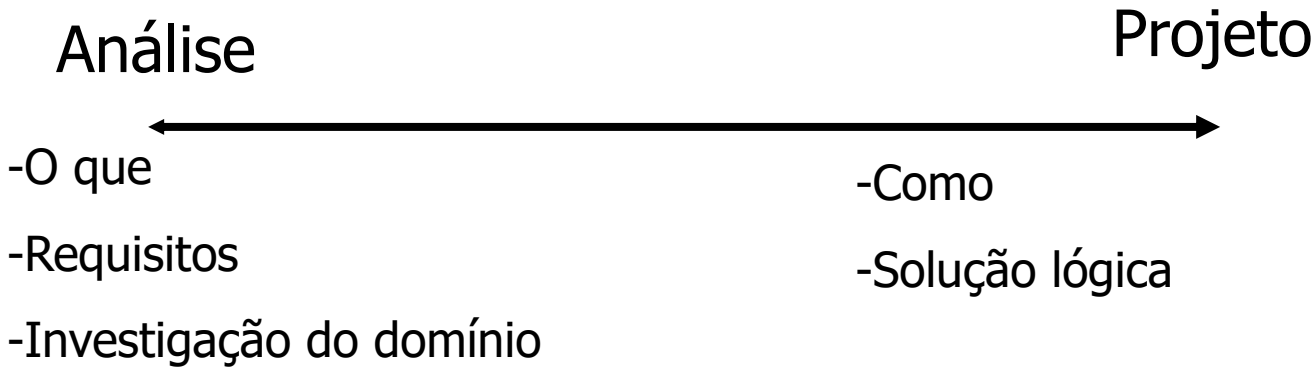
- Controle de Projeto
- Revisões Técnicas Formais
- Garantia de Qualidade
- Gerenciamento de Configuração
- Gestão de Reutilização
- Medição
- Gestão de Risco

Projeto de Software

Processo pelo qual os ***requisitos do software*** *são traduzidos numa representação do software que permite sua* ***realização física***



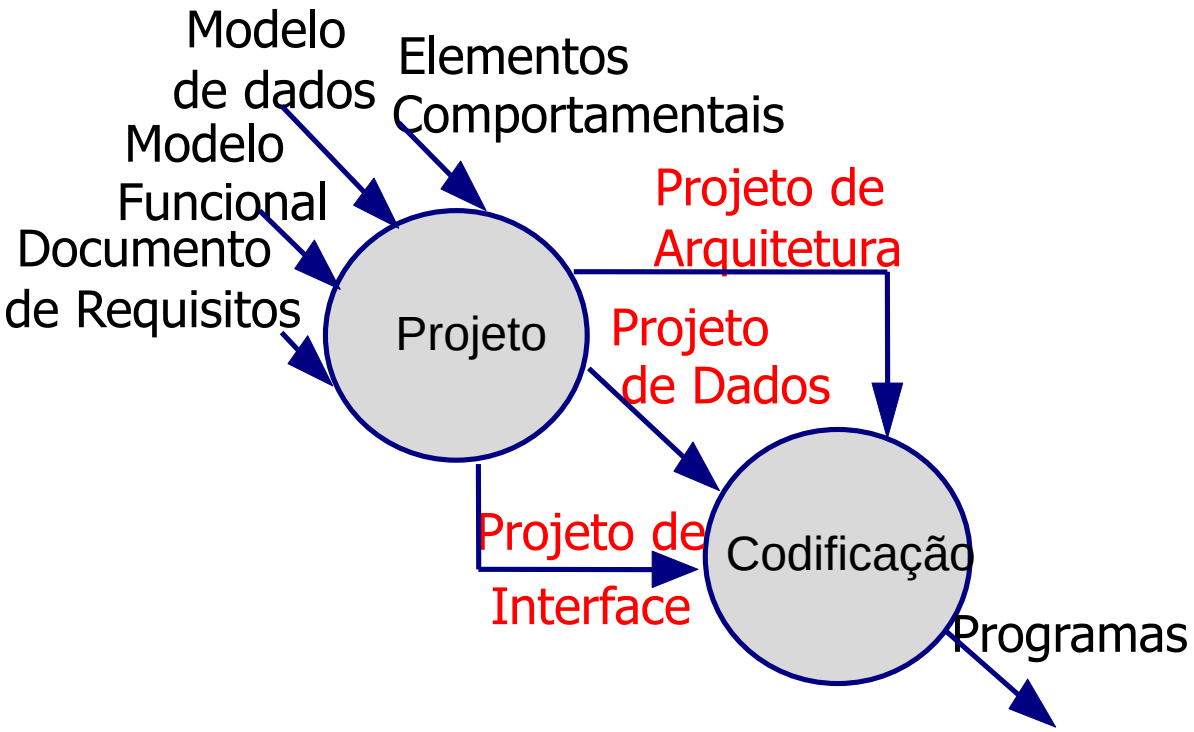
Projeto de Software



Projeto de Software

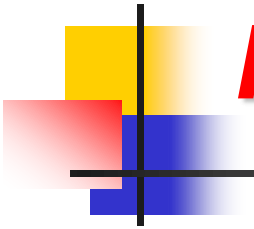
- Especificação de como será desenvolvida uma solução levantada na fase de análise
- Detalhes de implementação:
  - Ambientes de implementação adotado
    - linguagem, sistema operacional, banco de dados...
  - Arquitetura do software
    - módulos, comunicação entre módulos, ordem de execução
  - Detalhamento algorítmico
  - Aspectos de eficiência, desempenho e confiabilidade
    - Fundamental para sistema de tempo real, processamento paralelo, sistemas distribuídos...

Projeto de Software



...

Arquitetura do Software

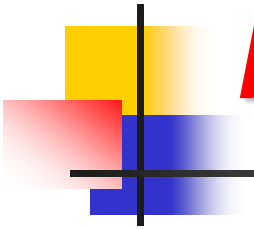


Projeto de Arquitetura

- Envolve a identificação dos componentes principais do sistema e suas comunicações
- A escolha da arquitetura do sistema pode afetar o desempenho, facilidade de distribuição e manutenção do sistema
  - A escolha pode ser influenciada pelos requisitos não funcionais

Exemplos de características de arquitetura

- Desempenho
  - Localizar operações críticas e minimizar comunicações. Usar componentes de alta granularidade – menos comunicação.
- Proteção
  - Usar uma arquitetura em camadas com itens críticos nas camadas mais internas.
- Segurança
  - Definir sub-sistemas exclusivos para tratar de aspectos de segurança do sistema
- Disponibilidade
  - Incluir componentes redundantes e mecanismos para tolerância a falhas.
- Facilidade de manutenção
  - Usar componentes substituíveis e de baixa granularidade.



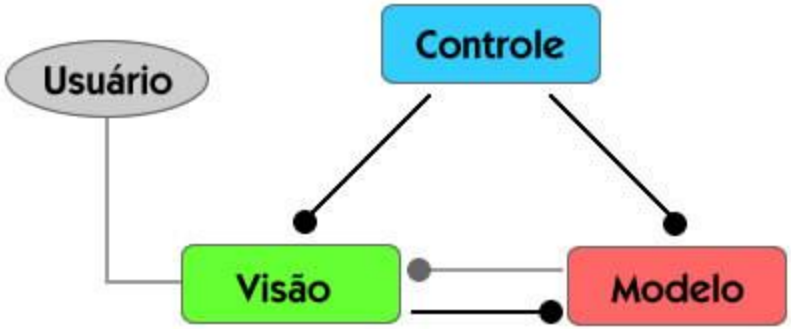
Modelos de Arquitetura

- Reflete a estratégia básica que é usada para estruturar um sistema.
- Padrões amplamente usados:
  - Arquitetura em camadas
  - Arquitetura de repositório de dados
  - Arquitetura cliente-servidor

Arquitetura em Camadas

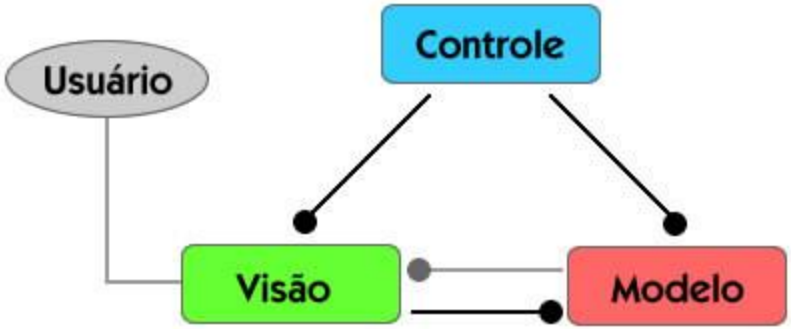
- Usado para modelar o interfaceamento dos subsistemas.
- Organiza o sistema em um conjunto de camadas (ou máquinas abstratas), cada uma das quais fornecendo um conjunto de serviços.
- Apóia o desenvolvimento incremental dos subsistemas em camadas diferentes. Quando uma camada de interface muda, somente a camada adjacente é afetada.

Exemplo: Padrão MVC



Quando usar?
Quando existirem várias maneiras de visualizar e interagir com os dados

Exemplo: Padrão MVC



Vantagem: apoia a apresentação dos dados de maneira diferentes

Não apropriado para dados e interações simples

Modelo de Arquitetura

- Reflete a estratégia básica que é usada para estruturar um sistema.
- Padrões amplamente usados:
  - Arquitetura em camadas
  - Arquitetura de repositório de dados
  - Arquitetura cliente-servidor

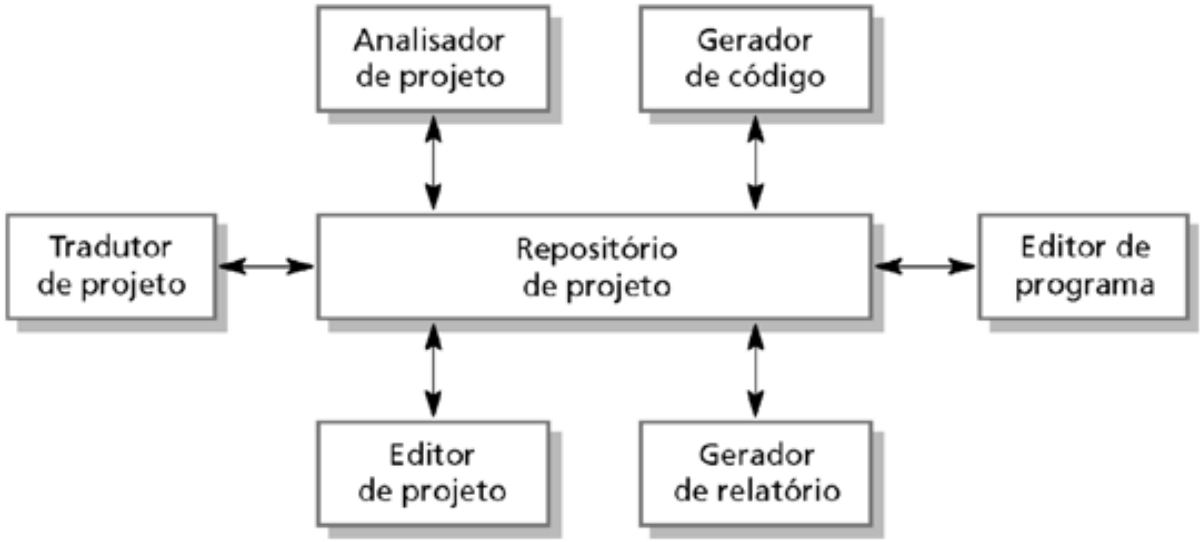
Arquietura de repositório de dados

- Os dados de um Sistema são gerenciados em um repositório central
  - Os componentes do Sistema não interage entre si, somente por meio do repositório

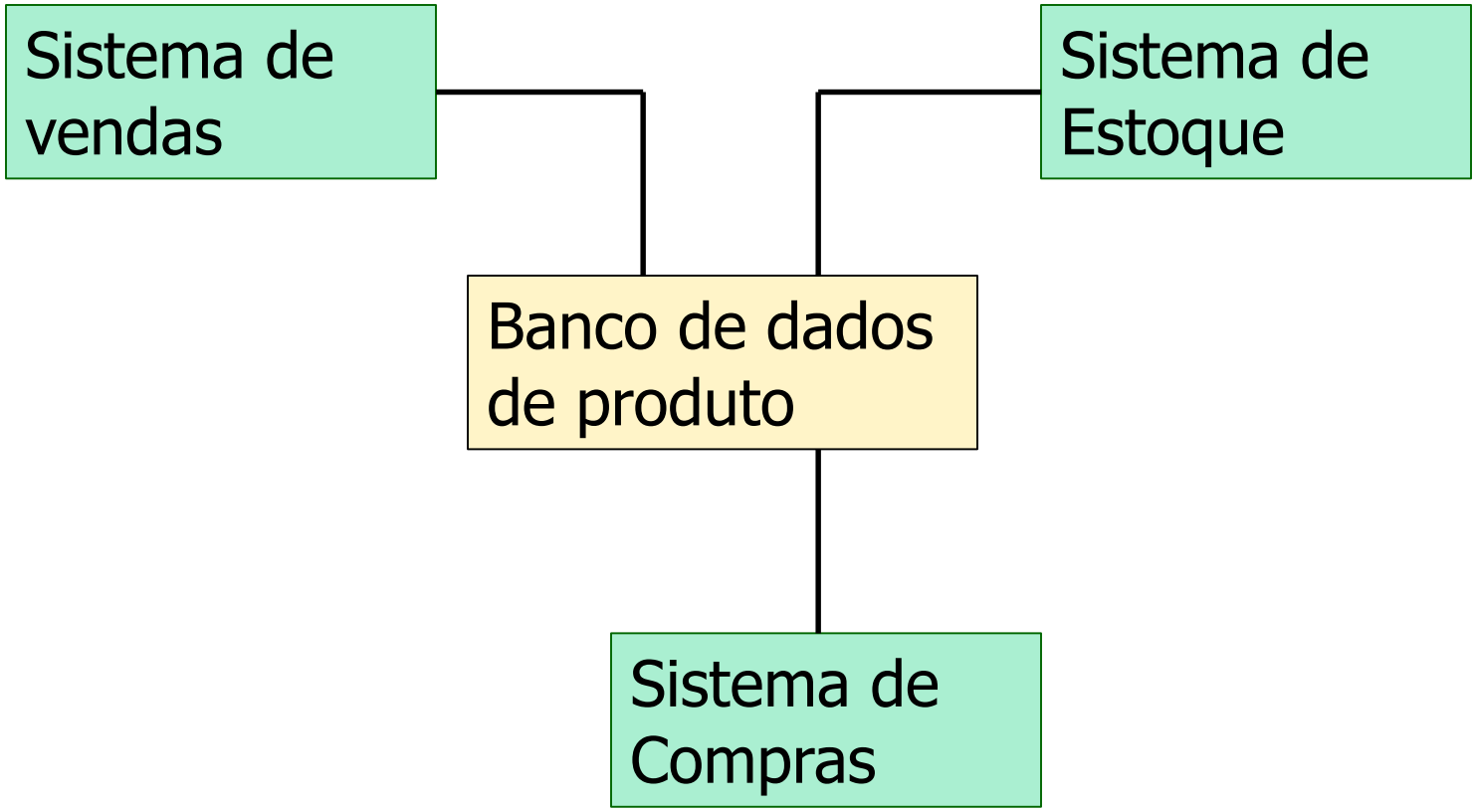
Exemplo: Arquitetura de repositório de dados

Figura 11.2

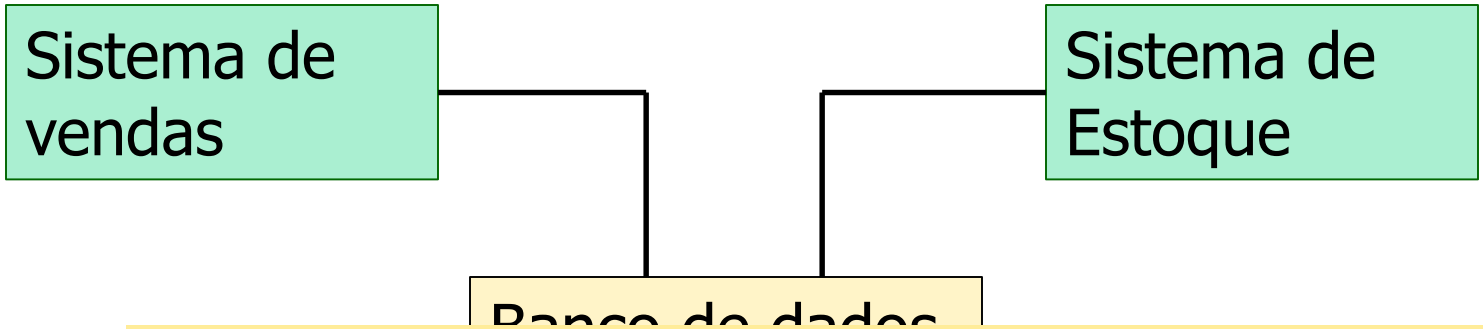
Arquitetura de um conjunto de ferramentas CASE integradas.



Exemplo: Arquitetura de repositório de dados



Exemplo: Arquitetura de repositório de dados



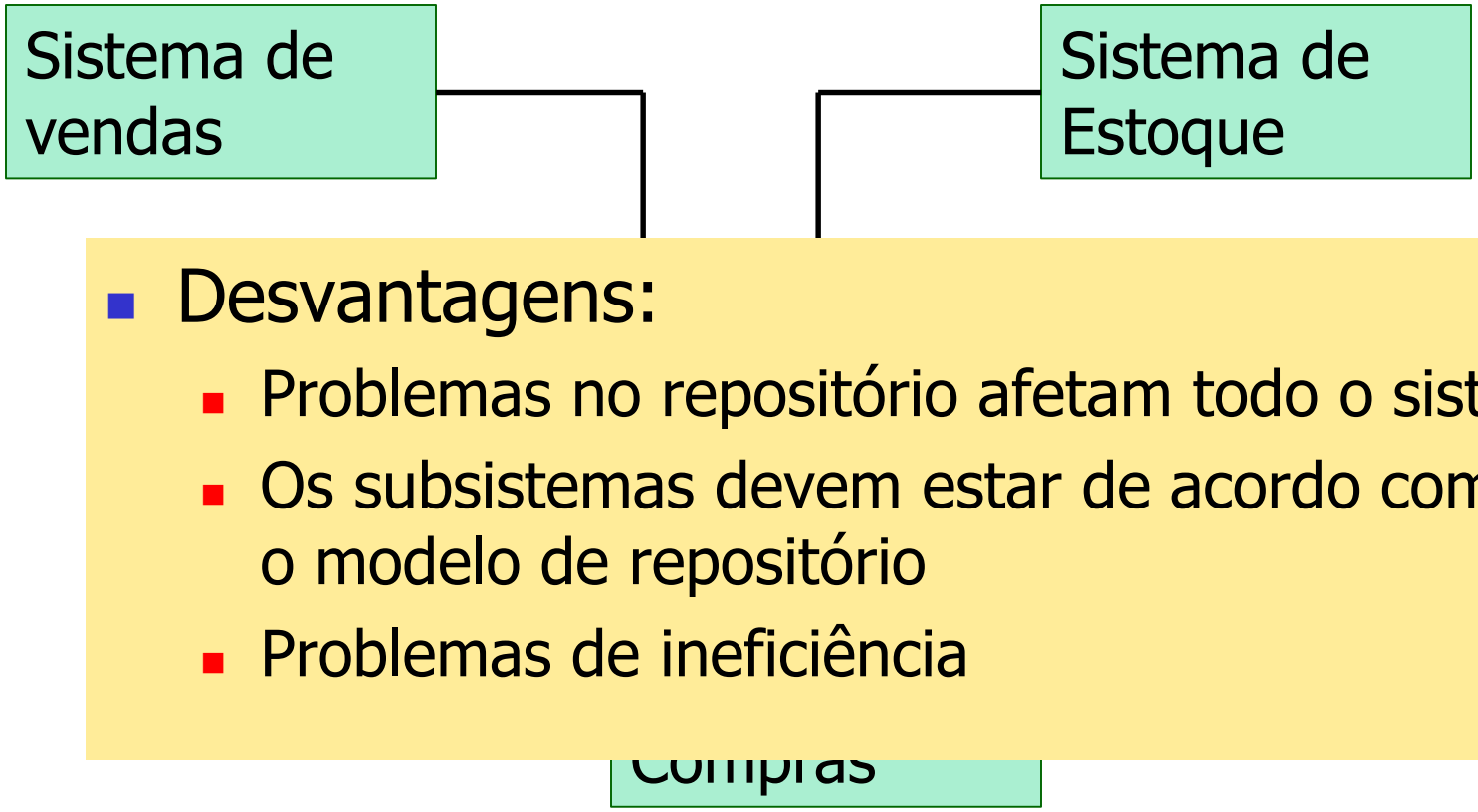
Quando usar?

Sistemas em que grande volume de informações são gerados

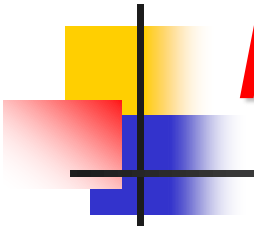
e precisam ser armazenados por longos períodos

Em sistemas dirigidos a dados (dados disparam ações no sistema)

Exemplo: Arquitetura de repositório de dados



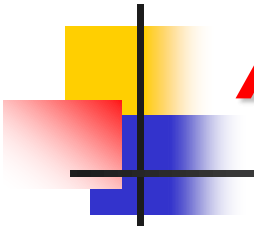
- Desvantagens:
  - Problemas no repositório afetam todo o sistema
  - Os subsistemas devem estar de acordo com o modelo de repositório
  - Problemas de ineficiência



Modelo de Arquitetura

- Reflete a estratégia básica que é usada para estruturar um sistema.
- Padrões amplamente usados:
  - Arquitetura em camadas
  - Arquitetura de repositório de dados
  - Arquitetura cliente-servidor

Arquitetura cliente-servidor

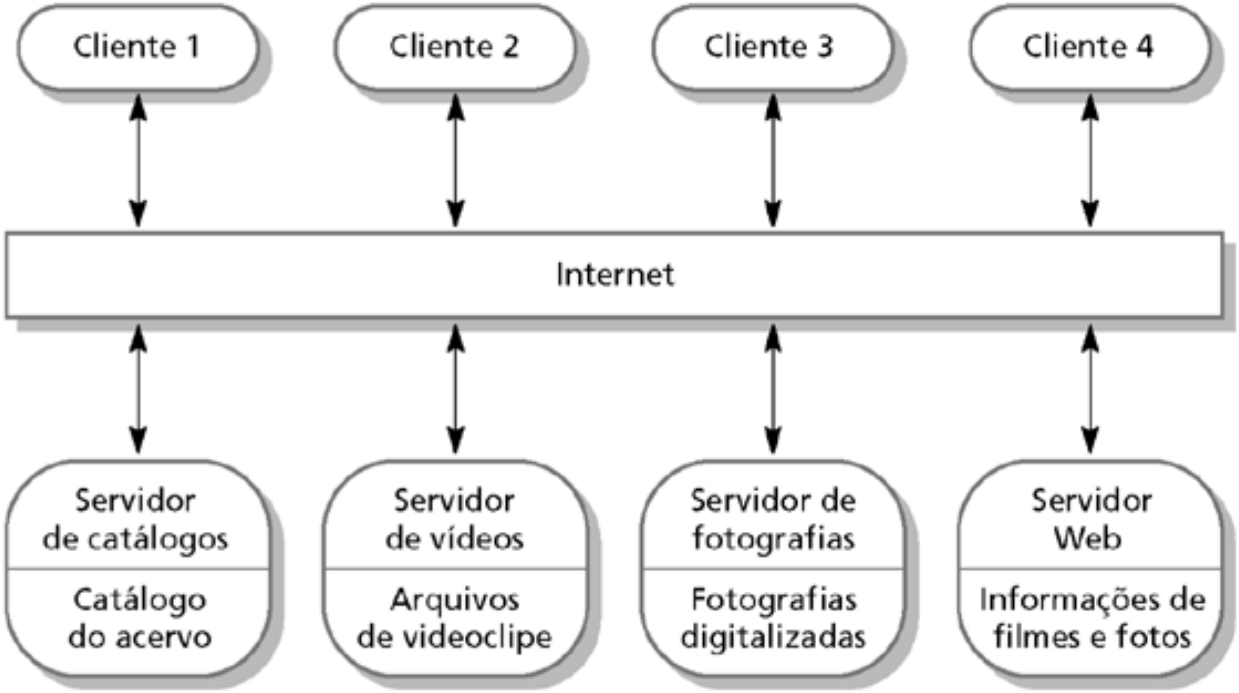


- A funcionalidade do Sistema está organizada em serviços
  - Cada serviço prestado por um servidor
- Ideal para sistemas distribuídos, pois mostra como dado e processamento são distribuídos por uma variedade de componentes.

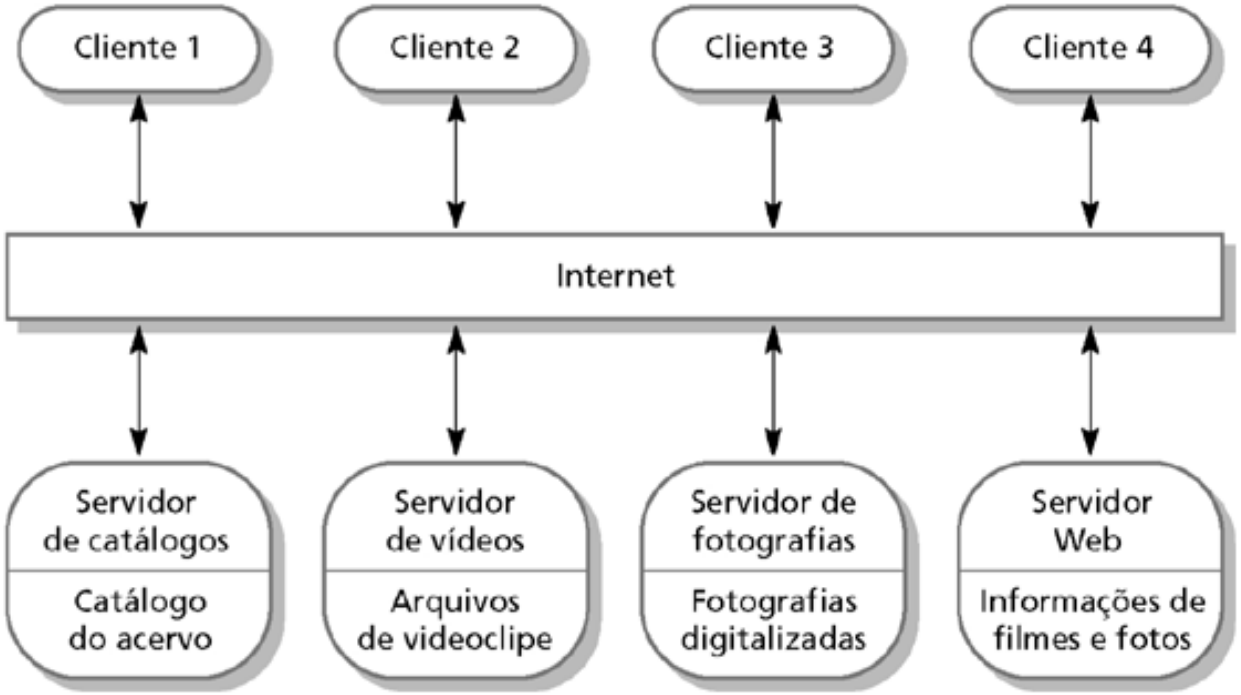
Modelo cliente-servidor

Figura 11.3

Arquitetura de um sistema de acervo de filmes e fotografias.



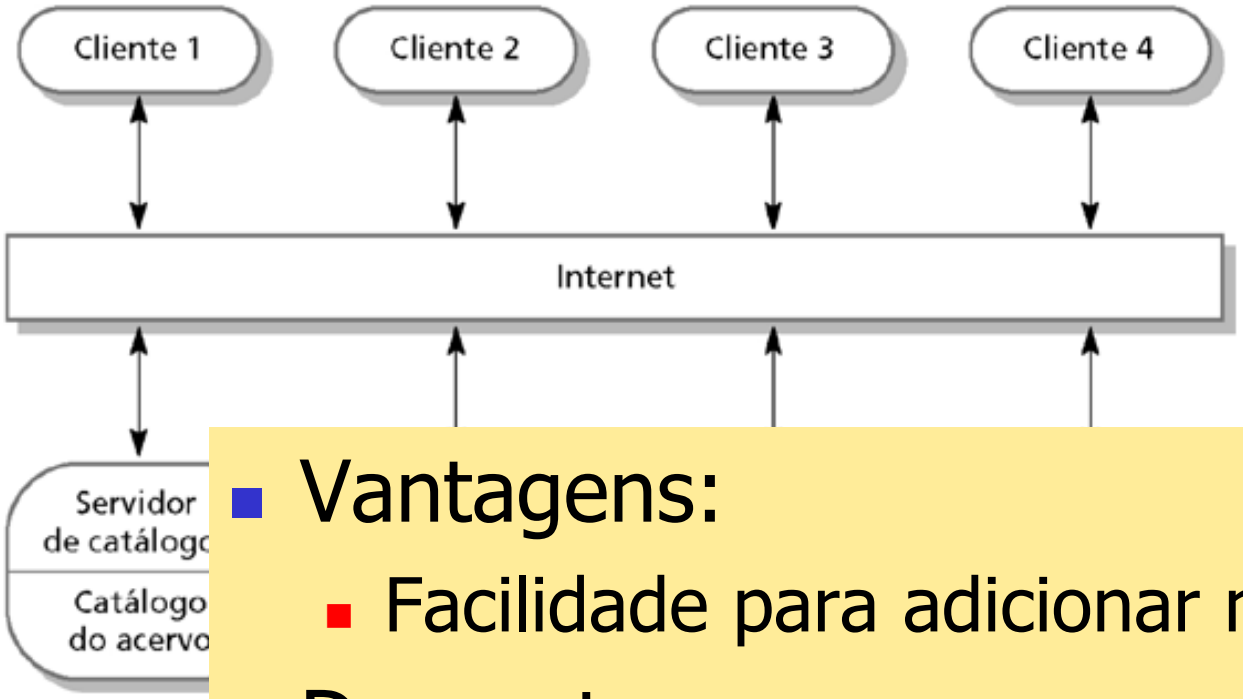
Modelo cliente-servidor



Quando usar?

Quando os dados em um BD compartilhado precisam ser acessados de vários locais

Modelo cliente-servidor

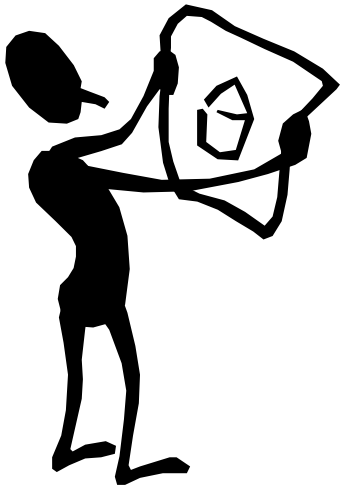


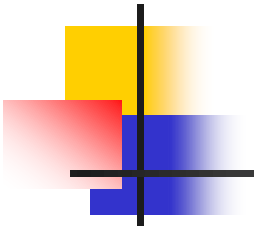
- Vantagens:
  - Facilidade para adicionar novos serviços.
- Desvantagem:
  - Problemas com desempenho da rede.

Qual arquitetura utilizar?

1. Sistema de emissão de bilhetes usado por passageiros em uma estação ferroviária
2. Sistema de videoconferência que permite que áudio, vídeo e dados sejam visíveis a todos os participantes ao mesmo tempo
3. Um robô limpador de chão, que se destina a limpar espaços livres. O robô deve perceber obstáculos e paredes.

Projeto de Dados

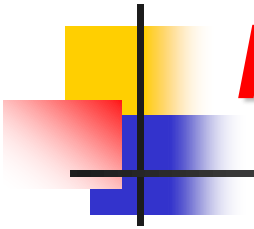




Projeto de Dados

- Retrata as estruturas de dados gerais e sucessivamente acrescentam-se detalhes sobre os elementos de dados e seus relacionamentos

- A partir de modelos da análise (ou DR), o projeto concentra-se nos elementos que devem ser manipulados pelo sistema.



Projeto de Dados

- Pode empregar modelos de BD (DER)
  - A partir dos dados extraídos, estabelece-se as funções que irão manipular esses dados.

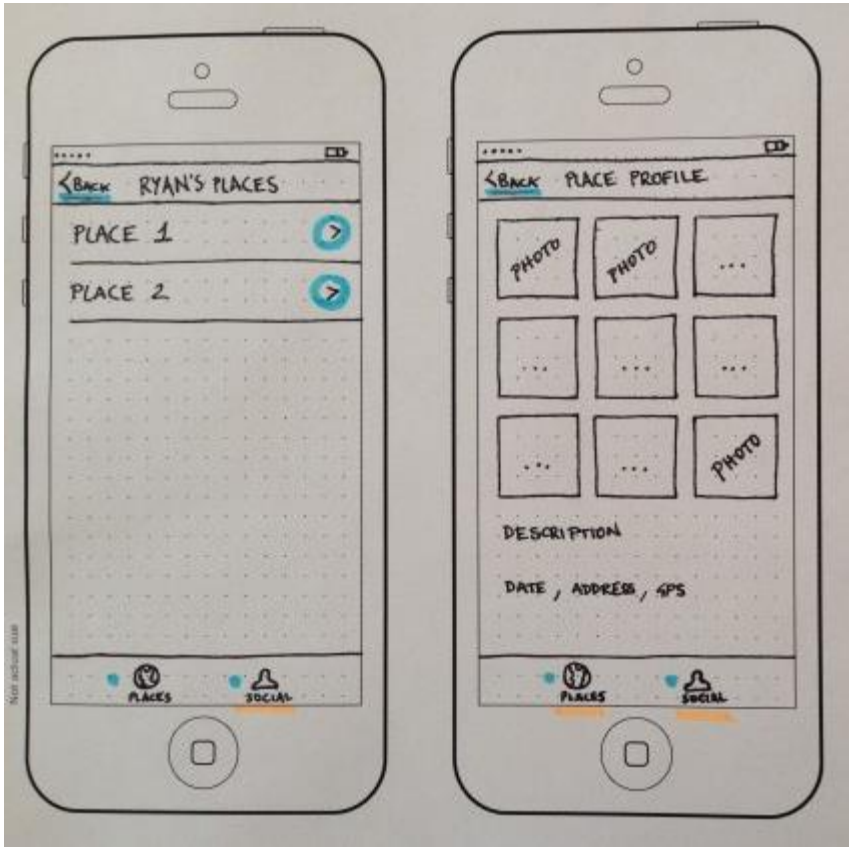
- Pode empregar modelos OO (Diagrama de Classes)
  - Identifica classes de objetos e seus relacionamentos
  - Apresenta os atributos de cada objeto, ações e relacionamentos

Projeto de Dados

- Gera o Diagrama de Classes com informações relevantes para implementação
  - Definição dos atributos
  - Identificação de métodos
  - Identificação de novos relacionamentos
  - Eventualmente, pode destacar relacionamentos e conceitos do modelo conceitual

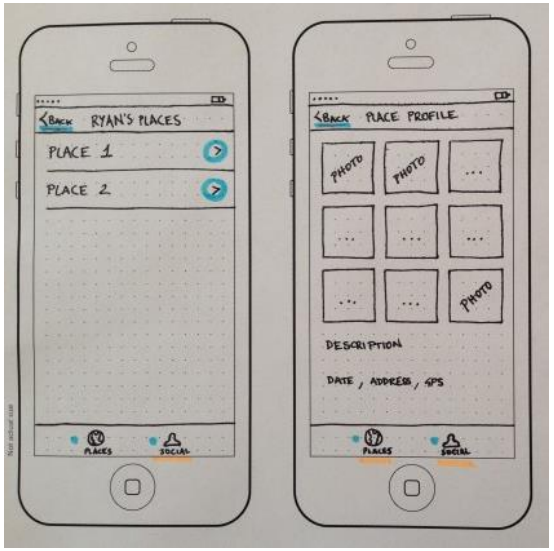
- O projeto OO engloba outros modelos:
  - diagramas de interações (seqüência e colaboração), modelagem do comportamento, pacotes, modelo de implementação ...

PROJETO DE INTERFACE



Interface com o usuário

- Projetadas para atender às habilidades, experiências e expectativas dos usuários previstos.
- Os usuários de sistemas frequentemente julgam um sistema pela sua interface ao invés de sua funcionalidade.
- Uma interface ruim pode levar um usuário a cometer erros e o software nunca ser usado.

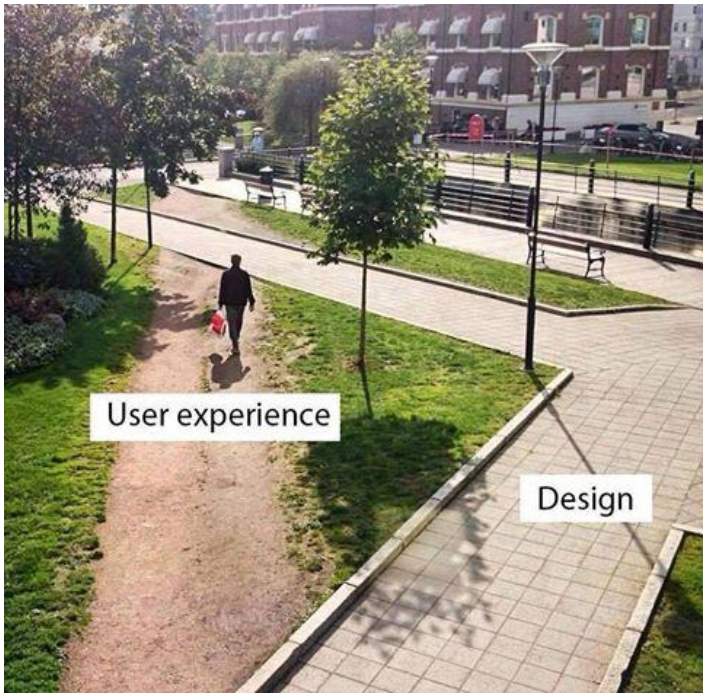


Fatores Humanos

- Memória limitada de curto prazo
  - Podemos lembrar instantaneamente de sete itens de informação.
- As pessoas cometem erros
  - alarmes e mensagens inapropriados podem aumentar o estresse e a probabilidade de mais erros.
- As pessoas são diferentes
  - capacidades físicas variadas: os projetos não podem ser desenvolvidos para as capacidades do desenvolvedor.
- As pessoas têm preferências diferentes de interação
  - Alguns gostam de figuras, outros, de textos

Princípios de Projeto de Interface de Usuário

1. Familiaridade de usuário
   - a interface deve usar termos e conceitos obtidos da experiência de pessoas que farão mais uso do sistema
2. Consistência
   - operações comparáveis devem ser ativadas da mesma maneira
3. Surpresa mínima
   - nunca surpreender o usuário com o funcionamento do sistema



Princípios de Projeto de Interface de Usuário

4. Facilidade de recuperação

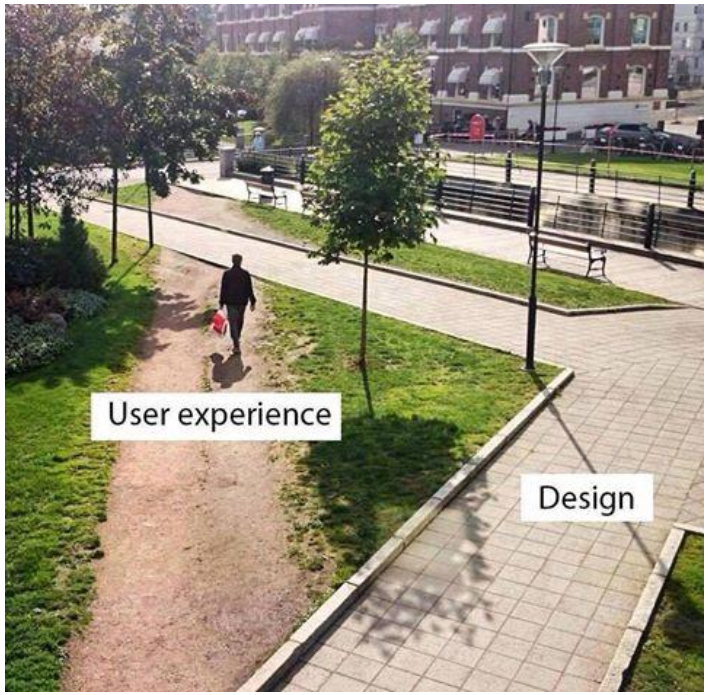
- mecanismos para recuperação de erros

5. Guia de usuário

- fornecer *feedback* significativo quando ocorrerem erros
- fornecer recursos sensíveis ao contexto para ajudar o usuário.

6. Diversidade de usuário

- interação adequada para tipos diferentes de usuários



Projeto de Interface de Usuário

- Pontos importantes:
  - Conhecer o usuário **(para quem?)**
    - **Uso de Personas**
  - Conhecer o domínio – tarefas **(o que? quando?)**
    - **Metas e ações das personas**
  - Dispositivo de interação **(Como?)**
    - **Diferentes tipos de dispositivos**

Projeto de Interface de Usuário

- Definir e validar com o cliente e usuários

Projeto de Software

Engenharia de Software

Simone do Rocio Senger de Souza

srocio@icmc.usp.br