DCC / ICEx / UFMG

Padrões Arquiteturais:

Sistemas Distribuídos, Interativos ou Adaptáveis

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Padrões de Arquitetura

- Sistemas Distribuídos
  - **Client-Server (Cliente-Servidor)**
  - Broker
- Sistemas Interativos
  - **Model-View-Controller (MVC)**
  - Presentation-Abstraction-Control
- Sistemas Adaptáveis
  - **Microkernel**
  - Reflection

Arquitetura Cliente-Servidor

Arquitetura Cliente-Servidor

- Organizada como um conjunto de serviços
  - Servidores dos serviços
  - Clientes dos serviços
- Exemplos de servidores
  - Servidor de impressão
  - Servidor de arquivos, etc.

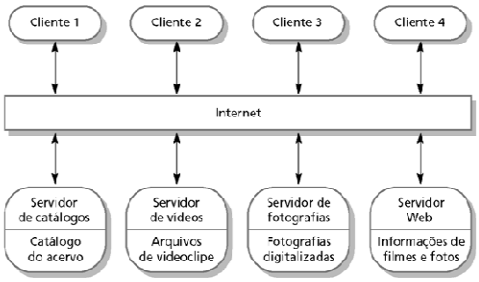
Arquitetura Cliente-Servidor

- Requer uma estrutura de rede para clientes acessarem os serviços
- Clientes sabem quais os serviços e servidores estão disponíveis
- Servidores não sabem quem são os clientes

Arquitetura Cliente-Servidor

- Essencialmente, é usado o protocolo *request-reply*
  1. Um cliente faz um pedido ao servidor e espera pela resposta
  2. O servidor executa o serviço e responde ao cliente

Exemplo de Cliente-Servidor



Vantagens

- A distribuição de dados é fácil e direta
- Faz uso efetivo dos recursos em rede
  - Possibilita hardware mais barato
- É fácil adicionar novos servidores ou atualizar servidores existentes

Desvantagens

- Não prevê modelo de dados compartilhado
  - Subsistemas usam diferentes organizações de dados
- Pode haver redundância de serviços em diferentes servidores
- Não prevê registro central de serviços
  - Difícil descobrir quais serviços estão disponíveis

Padrão MVC

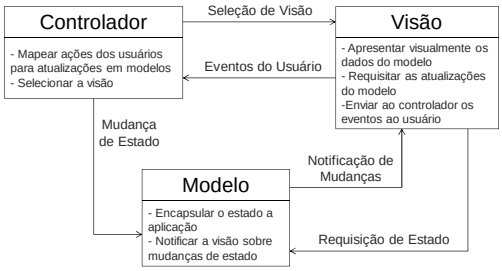
Padrão MVC

- Largamente utilizado em aplicações interativas Web
- Organiza o sistema em três componentes
  - **Modelo**: contém as funcionalidades e dados principais
  - **Visão**: responsável por apresentar os dados ao usuário
  - **Controlador**: trata os eventos de entrada

Descrição do MVC

- Separa a apresentação e a interação dos dados do sistema
  - Os três componentes tem responsabilidades distintas mas interagem entre si
- Quando é recomendado?
  - Quando existem várias maneiras de visualizar e interagir com os dados
  - São desconhecidos (ou são voláteis) os requisitos de interação com os dados

Representação dos Papéis



Vantagens

- Permite que os dados sejam alterados de forma independente de sua representação (e vice versa)
  - Apóia a apresentação dos mesmos dados de maneiras diferentes
- Facilita a distribuição do componente de visão
  - Os dados são mantidos centralizados e protegidos

Desvantagens

- Complexidade excessiva quando o modelo de dados e de interações é muito simples
  - A estrutura do padrão pode impor código adicional desnecessário

Padrão Microkernel

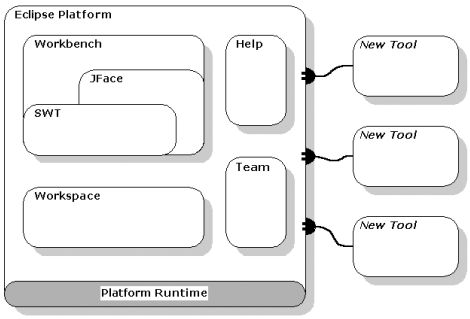
Microkernel

- Destinado a domínios de sistemas que possuem requisitos muito voláteis
  - Sistemas precisam ser capazes de se adaptar aos requisitos voláteis
- Este padrão de arquitetura separa a funcionalidade mínima em um núcleo
  - Novas funcionalidades são agregadas por extensões na forma de plug-in

Responsabilidades

- Microkernel
  - Provê as funcionalidades básicas
  - Oferece o meio de comunicação entre as extensões
  - Gerencia os recursos
- Extensões
  - Inclui novas funcionalidades
  - Fornece uma interface para interação com o microkernel e com outras extensões

Exemplo de Microkernel



Bibliografia

- Ian Sommerville. **Engenharia de Software**, 9a. Edição. 2011.
  - Cap. 6 (Seções 6.3 e 6.4)
- F. Buschmann et al. **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons, 1996.
  - Cap. 2 Architectural Patterns