DCC / ICEx / UFMG

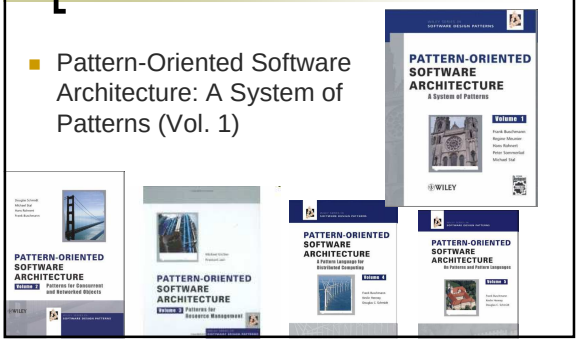
Examples of Architectural Patterns

Eduardo Figueiredo

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

Architectural Patterns Books

- Pattern-Oriented Software Architecture: A System of Patterns (Vol. 1)



Architectural Patterns

- From Mud to Structure
  - Layered Architecture
  - Blackboard
  - Pipes and Filters
- Distributed Systems
  - Client-Server
  - **Broker**
- Interactive Systems
  - Model-View-Controller (MVC)
  - **Presentation-Abstraction-Control**
- Adaptable Systems
  - **Microkernel**
  - Reflection

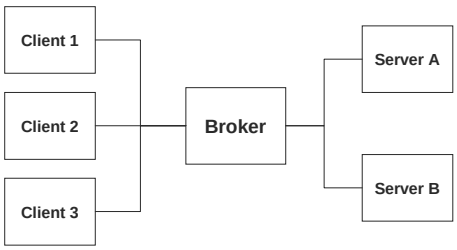
Broker

- Used to structure distributed systems with decoupled components
  - Components interact by remote service invocations
- The broker component is responsible for coordinating communication
  - Forwarding requests and transmitting results

Broker Roles

- **Servers** register themselves to the broker
  - They make their services available
- **Clients** do not know servers
  - They access the servers functionalities by sending requests to the broker
- **Broker** finds the appropriate server, forwarding the request to the server
  - And transmitting the results to the client

Broker Structure



Benefits

- Location Transparency
  - Clients do not need to know where servers are located
- Changeability and Extensibility
  - If a server changes but keeps its interface, it can be replaced by an equivalent server
- Interoperability between different Broker systems

Liabilities

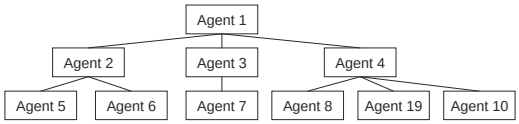
- Restricted efficiency
  - Applications using a broker implementation are usually slower
- Lower fault tolerance
  - If the broker fails during the program execution, clients are unable to access the servers

Presentation-Abstraction-Control

- PAC defines a structure for interactive systems as a hierarchy of agents
- Every agent is responsible for a specific aspect of the application functionality
- An agent is a unit (component) that handles events, updates its state and may produces new events
  - It can be as simple as a object or as complex as a complete system

PAC Structure

- The system should be organized in three layers
  - One top level agent
  - Some intermediate agents
  - Several bottom level agents



PAC Agent Roles

- Top Level Agents
  - Provides the core functionality of the system
  - Most other agents depend on this core
- Bottom Level Agents
  - Represent concepts or a group of functionality that users can act
- Intermediate Agents
  - Link top level agents to bottom level ones

PAC Agents

- Every agent consists of three parts: Presentation, Abstraction and Control
- Presentation
  - Interface of an agent (visible behavior)
- Abstraction
  - Data model of an agent
- Control
  - Connects presentation and abstraction
  - Allows agents communicate to each other

Benefits

- Separation of concerns
  - Different semantic concepts are handled by separate agents
- Support for change and extension
  - Changes inside an agent do not impact other agents
- Support for multi-tasking
  - Agents can be working in different tasks concurrently

Liabilities

- Increased system complexity
  - The implementation of every concern as an agent may result in a complex system
- Complex communication flow
  - Collaboration between agents may became complex in a hierarchical structure

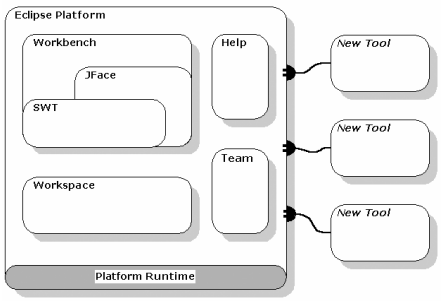
Microkernel

- 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



Benefits

- Portability
  - A Microkernel system offers a high degree of portability
- Flexibility and extensibility
  - It can easily includes and removes functionalities (plug-ins)
- Scalability
  - Each new functionality tends to be simple and self-contained

Liabilities

- Performance
  - Overhead of communication in a microkernel system tends to be high
- Complex design and implementation
  - Developing a microkernel system is not trivial
  - Implement plug-ins also requires knowledge about the system structure

Bibliography

- F. Buschmann et al. **Pattern-Oriented Software Architecture: A System of Patterns**. John Wiley & Sons, 1996.
  - Chapter 2 Architectural Patterns