Clasificación de patrones arquitectónicos según Wiley

From Mud to Structure

Patterns in this category help you to avoid a 'sea' of components or objects. In particular, they support a controlled decomposition of an overall system task into cooperating subtasks. The category includes the Layers pattern, the Pipes and Filters pattern and the Blackboard pattern.

- The Layers pattern helps to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction (for more information, see page 31).
- The Pipes and Filters pattern provides a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component. Data is passed through pipes between adjacent filters. Recombining filters allows you to build families of related systems (for more information, see page 53).
- The Blackboard pattern pattern is useful for problems for which no deterministic solution strategies are known. In Blackboard several specialized subsystems assemble their knowledge to build a possibly partial or approximate solution (for more information, see page 71).

Distributed Systems

This category includes one pattern: Broker. The Broker architectural pattern can be used to structure distributed software systems with decoupled components that interact by remote service invocations. A broker component Is responsible for coordinating communication, such as forwarding requests, as well as for transmitting results and exceptions (for more information, see page 99).

Interactive Systems

This category comprises two patterns, the Model-View-Controller pattern and the Presentation-Abstraction-Control pattern. Both patterns support the structuring of software systems that feature human-computer interaction.

The Model-View-Controller architectural pattern (MVC) divides an interactive application into three components. The model contains the core functionality and data. Views display information to the user. Controllers handle user input. Views and controllers together comprise the user interface. A change-propagation mechanism ensures consistency between the user interface and the model (for more information, see page 125).

The Presentation-Abstraction-Control architectural pattern (PAC) defines a structure for interactive software systems in the form of a hierarchy of cooperating agents. Every agent is responsible for a specific aspect of the application's functionality and consists of three components: presentation, abstraction, and control. This subdivision separates the human-computer interaction aspects of the agent from its functional core and its communication with other agents (for more information, see page 145).

Adaptable Systems

The Reflection pattern and the Microkernel pattern strongly support extension of applications and their adaptation to evolving technology and changing functional requirements.

The Microkernel architectural pattern applies to software systems that must be able to adapt to changing system requirements. It separates a minimal functional core from extended functionality and customer-specific parts. The microkernel also serves as a socket for plugging in these extensions and coordinating their collaboration (for more information, see page 171).

The Reflection architectural pattern provlde‌s a mechanism for changing structure and behavior of software systems dynamically. It supports the modification of fundamental aspects, such as type structures and function call mechanisms. In this pattern, an application is split into two parts. A meta level provides information about selected system properties and makes the software self-aware. A base level includes the application logic. Its implementation builds on the meta level. Changes to information kept in the meta level affect subsequent base-level behavior (for more information, see page 193).