Lecture 23: Software Architectures

→ Architectural Styles
  - Pipe and filter
  - Object oriented:
    - Client-Server, Object Broker
    - Event based
  - Layered:
    - Designing Layered Architectures
    - Blackboard, MVC
  - Process control

→ Examples:
  - UNIX shell commands
    - Filters: Lexical Analysis -> parsing -> semantic analysis -> code generation
  - Signal Processing

→ Interesting properties:
  - Filters don't need to know anything about what they are connected to
  - Filters can be implemented in parallel
  - Behaviour of the system is the composition of behaviour of the filters
  - Specialized analysis such as throughput and deadlock analysis is possible

→ Object Oriented Architectures

→ Examples:
  - Abstract data types

→ Interesting properties
  - Data hiding (internal data representations are not visible to clients)
  - Can decompose problems into sets of interacting agents
  - Can be multi-threaded or single thread

→ Disadvantages
  - Objects must know the identity of objects they wish to interact with

→ Variant 1: Client Server

→ Interesting properties
  - Is a special case of the previous pattern object oriented architecture
  - Clients do not need to know about one another

→ Disadvantages
  - Client objects must know the identity of the server
Variant 2: Object Brokers

→ Interesting properties
  % Adds a broker between the clients and servers
  % Clients no longer need to know which server they are using
  % Can have many brokers, many servers.

→ Disadvantages
  % Broker can become a bottleneck
  % Degraded performance

Broker Architecture Example

→ Examples
  % Database management systems (for data integrity checking)
  % Graphical user interfaces

→ Interesting properties
  % Announcers of events don't need to know who will handle the event
  % Supports re-use, and evolution of systems (add new agents easily)

→ Disadvantages
  % Components have no control over ordering of computations

Event based (implicit invocation)

→ Examples
  % Debugging systems (listen for particular breakpoints)
  % Database management systems (for data integrity checking)
  % Graphical user interfaces

→ Interesting properties
  % Announcers of events don't need to know who will handle the event
  % Supports re-use, and evolution of systems (add new agents easily)

→ Disadvantages
  % Components have no control over ordering of computations

Layered Systems

→ Examples
  % Operating Systems
  % Communication protocols

→ Interesting properties
  % Support increasing levels of abstraction during design
  % Support enhancement (add functionality) and re-use
  % Can define standard layer interfaces

→ Disadvantages
  % May not be able to identify (clean) layers
**Variant: 3-layer data access**

- **Presentation layer**
  - Application (client)
  - Database (server)
- **Application Logic layer**
  - Presentation layer (user interface)
  - Business Logic
  - Database
- **Storage layer**
  - Presentation layer (user interface)
  - Applications
  - Domain Entities
  - Database

**Open vs. Closed Layered Architecture**

- **Closed architecture**
  - Each layer only uses services of the layer immediately below.
  - Minimizes dependencies between layers and reduces the impact of a change.

- **Open architecture**
  - A layer can use services from any lower layer.
  - More compact code, as the services of lower layers can be accessed directly.
  - Breaks the encapsulation of layers, so increases dependencies between layers.

**How many layers?**

- **2-layers:**
  - Application layer
  - Database layer
  - E.g. simple client-server model
- **3-layers:**
  - Separate out the business logic
  - Helps to make both user interface and database layers modifiable
- **4-layers:**
  - Separates applications from the domain entities that they use:
  - Boundary classes in presentation layer
  - Control classes in application layer
  - Entity classes in domain layer
- **Partitioned 4-layers**
  - Identify separate applications

**Repositories**

- **Examples**
  - Databases
  - Blackboard expert systems
  - Programming environments
- **Interesting properties**
  - Can choose where the locus of control is (agents, blackboard, both)
  - Reduce the need to duplicate complex data
- **Disadvantages**
  - Blackboard becomes a bottleneck
Variant: Model-View-Controller

→ Properties
- One central model, many views (viewers)
- Each view has an associated controller
- The controller handles updates from the user of the view
- Changes to the model are propagated to all the views

Model View Controller Example

MVC Component Interaction

Process Control

→ Examples
- aircraft/spacecraft flight control systems
- controllers for industrial production lines, power stations, etc.
- chemical engineering

→ Interesting properties
- separates control policy from the controlled process
- handles real-time, reactive computations

→ Disadvantages
- Difficult to specify the timing characteristics and response to disturbances