**Lecture 23: Software Architectures**

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

- **Pipe-and-filter**

  ![Pipe-and-filter Diagram]

  - **Examples:**
    - UNIX shell commands
    - Compilers:
      - 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

*Source: Adapted from Shaw & Garlan 1996, p21-2. See also van Vliet, 1999 Pp266-7 and p279*
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

Source: Adapted from Shaw & Garlan 1996, p22-3.
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
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**
  - Java
  - AWT

- **Application Logic layer**
  - Objects
  - Control
  - Business logic

- **Storage layer**
  - Query
  - Engine
  - DBMS
  - File

---

**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 increase 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

Navigateability arrows show the directions in which messages will be sent.
### MVC Component Interaction

![Diagram of MVC Component Interaction]

### Process Control

*Source: Adapted from Shaw & Garlan 1996, p27-31.*

#### 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