**XVIII.1 Software Architectures**

- **Software Architectures**
  - Subsystems, Modules and Connectors
  - Pipes and Filters, Object-Oriented, Layered, Event-Driven, Repository-Based Architectures
  - Client Server Architectures
  - Web-Based Software Architectures
  - Examples

---

**Architectural Styles**

- It is useful to classify software architectures into classes of *architectural styles*.
- For example, the client-server architecture discussed earlier is an architectural style.
- The styles we’ll discuss below are as follows:
  - Pipes and filters;
  - Object-Orientation;
  - Event-Based
  - Layered;
  - Repository-Based;
  - Client-Server;
  - Three-Tier;
  - …more…
Pipes and Filters

- Each component has inputs and outputs. A component reads streams of data on its inputs and produces data on its outputs, continuously as data are coming in.
- Components compute by performing local transformations on their inputs to produce their outputs and are termed filters. The connectors of components transmit the outputs of one component to the inputs of another and are termed pipes.
- Unix supports a linear pipe and filter architecture called pipeline.

Pipes and Filters: Strengths and Weaknesses

**Strengths**

- Makes it easy to understand overall function of the system as a composition of filter functions
- Encourages reuse of filters
- Facilitates maintenance
- Facilitates deadlock and throughput analysis

**Weaknesses**

- Often leads to batch-type processing
- Not good for interactive applications where you often want to do incremental computations, e.g., incremental display updates
- Can’t coordinate stream inputs
- Data transmission critical for system performance
**Data Abstraction and Object-Orientation**

- Data structures and their associated operations are **encapsulated** in an abstract data type (ADT) or **object**. The components of a system are instances of an ADT and they interact through procedure (or **method**) calls.
- An object is responsible for preserving the integrity of its data structures and also these data structures are hidden from other objects.
- Objects may operate concurrently or not.

**Data Abstraction: Strengths and Weaknesses**

**Strengths**

- Possible to change implementation of an object without affecting its clients.
- Encourages decomposition of a problem into a number of interacting components/agents.
- Encourages software reuse.

**Weaknesses**

- For an object to interact with another, it must know its identity (not so for pipe&filter architectures).
- When the methods of an object change, so must all other objects that use this object.

*Client-Server Architecture a special case of the Data Abstraction Architecture*
**Event-Based Architectures**

Instead of invoking a procedure directly, a component can announce one or more events (such as arrival of data or execution of an operation)

- **On** `<event>` *if* `<condition>` *then* `<action>`
- **On** `arrive(D)` *if* `D < a` or `D ≥ b` *then* `print("out of bounds")`

Such procedures are also called triggers, actors or event-condition-action (ECA) rules.

An advantage of event-based invocation is that it encourages reuse; a component can be introduced in a system simply by registering it for the events of that system.

A drawback is that sometimes event-based systems become quite unpredictable and hard to control.

---

**Event-Based Architecture for Data Integration**

[Gal97]
**Layered Systems**

- A layered system is organized hierarchically, each layer serving the layer above. In some systems, inner layers are hidden in all but the adjacent outer layer.
- Best examples of layered software systems are layered communication protocols.
- Layered systems support design based on increasing levels of abstraction. However, not all systems can be structured in a layered fashion.

![Layered System Diagram](image)

**Repository-Based Architectures**

![Repository-Based Architecture Diagram](image)
Repository-Based Architectures

- A repository architecture consists of a central data structure (often a database) and a collection of independent components which operate on the central data structure.
- Examples of repository architectures include blackboard architectures, where a blackboard serves as communication centre for a collection of knowledge sources, and database systems serving several applications.
- Repositories are very important for data integration, are being introduced in a variety of applications, including software development, CAD etc.

Other Architectural Styles

- Table-Driven Interpreters -- each interpreter offers a "virtual machine" to high layers of interpreters; special case of the layered architecture.
- Distributed Processes -- program consists of distributed components organized into a static or dynamic configuration; this is a special case of the object-oriented architecture.
- Main Program/Subroutine -- FORTRAN-style architecture.
- State-Transition Architecture -- system structured in terms of states, state transitions; useful architecture for real-time systems.
Additional Reading