Software Engineering I CSC-382

Lecture 15

Software Engineering I CS-382

- Chapter 10
- What we will cover: (Architectural Design)
  - Chapter 10 in Pressman (Section …)
  - Goal of this chapter is to understand the tools and approaches we will follow in the top-level design phase, also called Architectural Design

Why Architecture?

- The architecture is not the operational software
- Rather, it is a representation that enables a software engineer to:
  - analyze the effectiveness of the design in meeting its stated requirements
  - consider architectural alternatives at a stage when making design changes is still relatively easy, and
  - reduce the risks associated with the construction of the software

What is Architecture?

- Large systems are always decomposed into sub-systems
- Architectural design is the initial top-level design
- We define:
  - The sub-systems
  - The control framework for executing the sub-systems
  - The communications framework between the sub-systems
- To date we have focused primarily on functional requirements
- Defining the best architecture for a system strongly influences and is strongly influenced by **non-functional** requirements

Non-functional Requirements Driving Architecture

- Performance
  - If the performance is most critical then the architecture should localize critical operations and use a coarse-grain architecture (few larger components) to reduce communication
- Security
  - If security is critical then a layered architecture should be considered
- Safety
  - Design system to place safety related items together to simplify validation

Non-functional Requirements Driving Architecture II

- Availability
  - If system availability is most critical then consider a redundant architecture
- Maintainability
  - If long-term maintainability is critical then consider a fine-grain architecture of many self-contained components

Questions That The System Architect Must Ask

1. Is there a generic application architecture that can act as a template?
   - Recall the lecture on Patterns and Frameworks
2. What architectural **style** is most appropriate?
3. What will be the fundamental approach used to structure the system?
4. How will the structural units in the system be **decomposed into modules**?

From: Sommerville

Questions That The System Architect Must Ask II

5. If it is a distributed architecture, then how should the system be distributed across a number of processors?
6. What will the **control strategy** for the system be?
7. How will we evaluate candidate architectures?
8. How should the architecture be documented?

From: Sommerville

Why is Architecture Important?

- The architecture selected implies a certain view of the problem
  - This will naturally lead to some criteria in the design being more heavily weighted in the subsequent design of the system.
  - The architecture highlights early design decisions that will have a profound impact on all software engineering work that follows.

Why is Architecture Important? II

- Representations of software architecture are an enabler for communication between all parties (stakeholders) interested in the development of a computer-based system.
  - Architecture “constitutes a relatively small, intellectually graspable model of how the system is structured and how its components work together” [BAS03].

Information Underlying Each Architectural Style

- Each style describes a category of system that depends on:
  1. a set of components that perform a function required by a system, (e.g., a database, computational modules)
  2. a set of connectors that enable "communication, coordination and cooperation" among components,
  3. constraints that define how components can be integrated to form the system, and
  4. semantic models that enable a designer to understand the overall properties of a system by analyzing the known properties of its constituent parts.

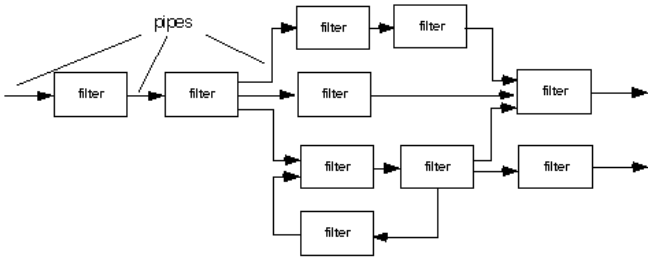
Common Architectural Styles

Data Flow Systems	Virtual Machines
Call and Return Systems	Data-centered Systems (repositories)
Independent Components	Process Control Systems
Layered Hierarchy (Abstract machine)	
	From: Shaw & Garlan

Elements of the Organizational Space

- Data Dimension:
  1. Central Data (repository or database approach)
  2. Distributed Data (data exchanged via message passing)
- Object vs Functional Dimension
  1. Object-based Decomposition (can be indep or call-return styles)
  2. Functional Decomposition (leads to pipeline styles)
- Control Dimension
  1. Centralized Control (top-down style or single controller in system)
  2. Distributed Control (message and event driven systems)

Data Flow Architectural Style



(a) pipes and filters

(b) batch sequential

In batch sequential, data is only passed onto the next Filter when it is completely processed by the previous filter

Call-return Architectural Style

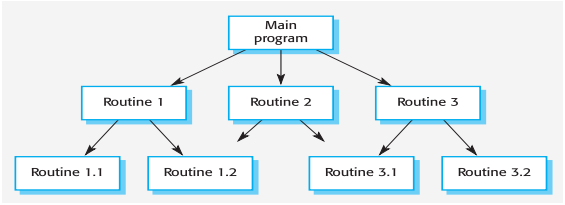
- Two common sub-architectures for this style:
  1. **Main program-sub-program**
     - This architecture is derived from the classic program structure where we divide a main program into its subroutines and functions
     - This system is divided the same way with a main controller and its subordinate components

Call-return Architectural Style II

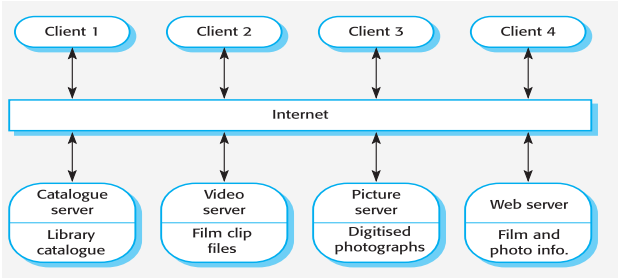
- Two common sub-architectures for this style:
  2. Remote Procedure Call
     - Client server fits this model
     - Set of stand-alone servers which provide specific services such as printing, data management, etc.
     - Set of clients which call on these services via request-reply protocols.
     - Client sits and waits until Server returns with a reply

Call-return Architectural Style III

1. Main program-sub-program



2. Remote Procedure Call (client server)

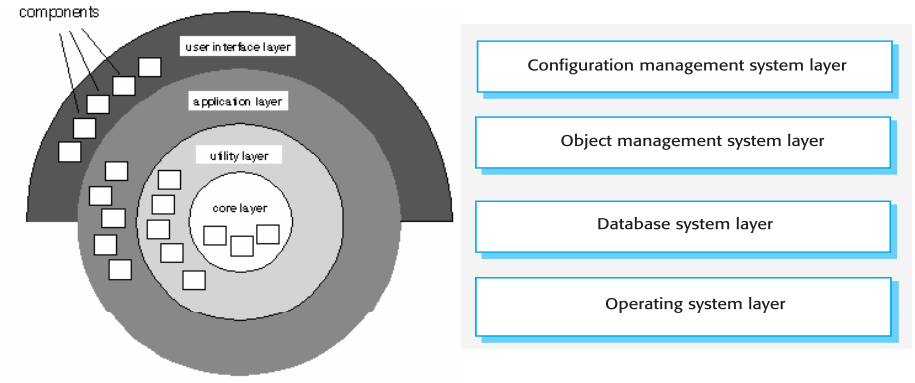


From: Sommerville

Layered Architectural Style

- Organises the system into a set of layers (or abstract machines) each of which provide a set of services.
- Layers only communicate with nearest neighbor layers
- However, often artificial to structure systems in this way, but systems like protocol stacks and Operating Systems fit it nicely.

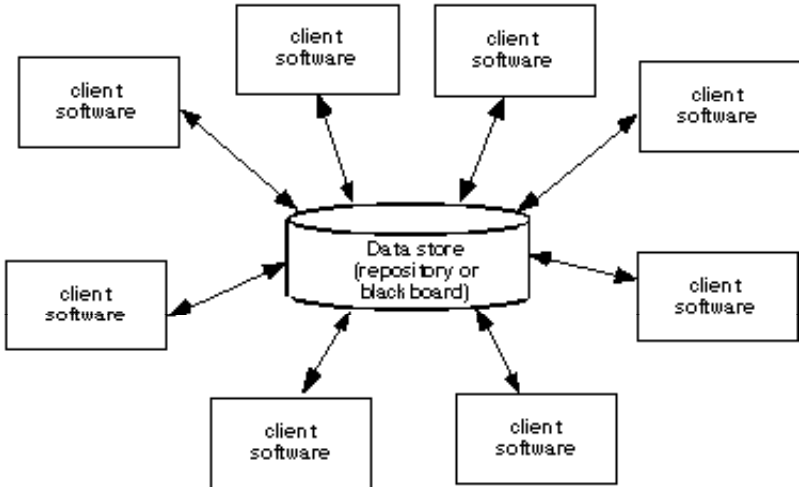
Layered Architectural Style



Data-Centered (Repositories) Architecture

- Two types of control mechanisms define two types of systems
  - In **Database** systems, the type of transaction entered into the data store triggers which process to execute
  - In **Blackboard** systems, the current state of the central data store triggers the process to execute.
    - Can be useful in Robotics where multiple processes are working together on a problem and the specific task changes as the problem progresses

Data-Centered (Repositories) Architecture

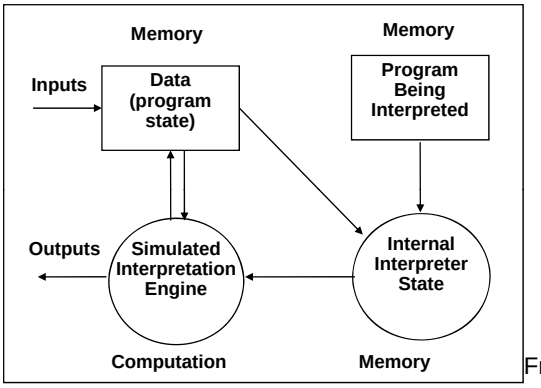


Repository Model Characteristics

- Advantages
  - Efficient way to share large amounts of data;
  - Sub-systems need not be concerned with how data is produced Centralised management e.g. backup, security, etc.
  - Sharing model is published as the repository schema.
- Disadvantages
  - Sub-systems must agree on a repository data model. Inevitably a compromise;
  - Data evolution is difficult and expensive;
  - No scope for specific management policies;
  - Difficult to distribute efficiently.

Interpreter Architectural Style

- Interpreters are Virtual Machines that exist in software
  - The interpreter includes the program to interpret and the interpretation engine



From: Shaw & Garlan

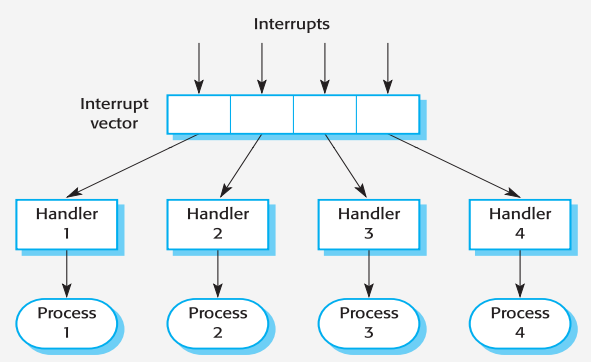
Event-driven Architectural Style

- Driven by externally generated events where the timing of the event is outwith the control of the sub-systems which process the event.

- Two principal event-driven models
  - **Broadcast models**.
  - **Interrupt-driven models**.

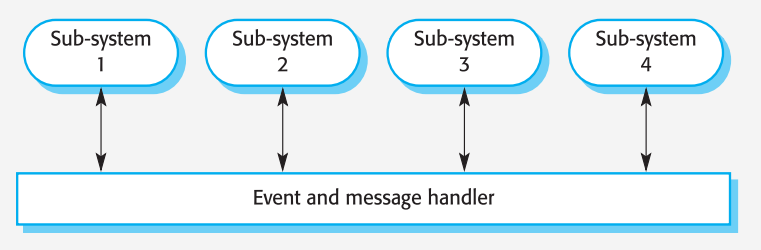
Event-driven Architectural Styles

- Interrupt:
  - Used in real-time systems where interrupts are detected by an interrupt handler and passed to some other component for processing.



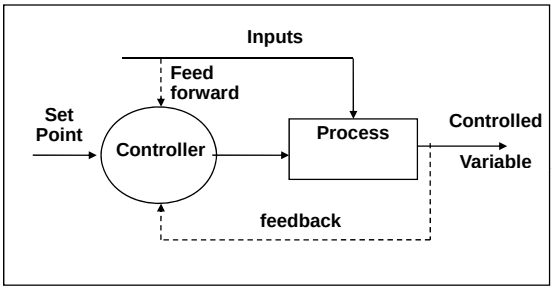
Event-driven Architectural Styles

- Broadcast: An event is broadcast to all sub-systems. Any sub-system which can handle the event may do so;
  - Sub-systems register an interest in specific events. When these occur, control is transferred to the sub-system that handles the event.
  - E.g. Object systems using Object Request Brokers (ORBs)



Process Control Architectural Style

- Similar to Data Flow Architecture
  - Difference is that control architectures have cyclic topologies



From: Shaw & Garlan

Heterogeneous Architectural Styles

- Note these have been ‘pure’ architectural styles
  - It is unlikely you would use just one of these.
  - Complex systems can require mixes of the styles

Mixing Architectural Styles

1. Hierarchical Combination
   - Recall a system is composed of sub-systems
   - Each sub-system can likewise be decomposed using its own style
2. Use a mixture at any level (including top level)
   - Example, data may be extracted from a data repository, but then processed using a pipeline architecture
   - Common in real-time, where a process is started from interrupt and this process then performs pipeline processing
3. Different levels in a hierarchy uses a particular model

In-class Architecture Definition

For Next Class

- Finish Chapter 10 – Architectural Design