[CS3704] Software Engineering

Shawal Khalid

Virginia Tech

3/18/2024

Announcements

- HW3 due March 22nd at 11:59pm
- Discussion Presentation on March 27th is moved to March 25th!

High-Level Design

High-Level Design
Software Architecture
Architecture Patterns

Learning Outcomes

By the end of the course, students should be able to:

- **Understand software engineering processes, methods, and tools used in the software development life cycle (SDLC)**
- **Use techniques and processes to create and analyze requirements for an application**
- **Use techniques and processes to design a software system**
- **Identify processes, methods, and tools related to phases of the SDLC**
- **Explain the differences between software engineering processes**
- **Discuss research questions and current topics related to software engineering**
- **Create and communicate about the requirements and design of a software application**

Design

Goal: decide the structure of the software and the hardware configurations that support it.

- The *how* of the project

- How individual classes and software components work together in the software system.
  - Programs can have 1000s of classes/methods
- *Software Artifacts:* design documents, class diagrams (i.e. UML)

Design Engineering

- The process of making decisions about HOW to implement software solutions to meet requirements.
- Encompasses the set of concepts, principles, and practices that lead to the development of high-quality systems.

High-Level Design

- Explains the architecture used to develop a system.
    - Also known as architectural design…
    - But there is debate on which term is most appropriate
- Provides a technical representation of functional (and some non-functional) requirements and the flow of information across assets or components in the system.

What is Software Architecture?

“Software architecture is, simply, the organization of a system. This organization includes all components, how they interact with each other, the environment in which they operate, and the principles used to design the software. In many cases, it can also include the evolution of the software into the future.”

[CAST Software]

Software Architecture

- Design the overall shape and structure of a system
  - The components
  - Their externally visible properties
  - Their relationships
- **Goal:** Construct the program to reduce risks in software construction and meet requirements

Software Architecture (cont.)

Software architecture is composed of:

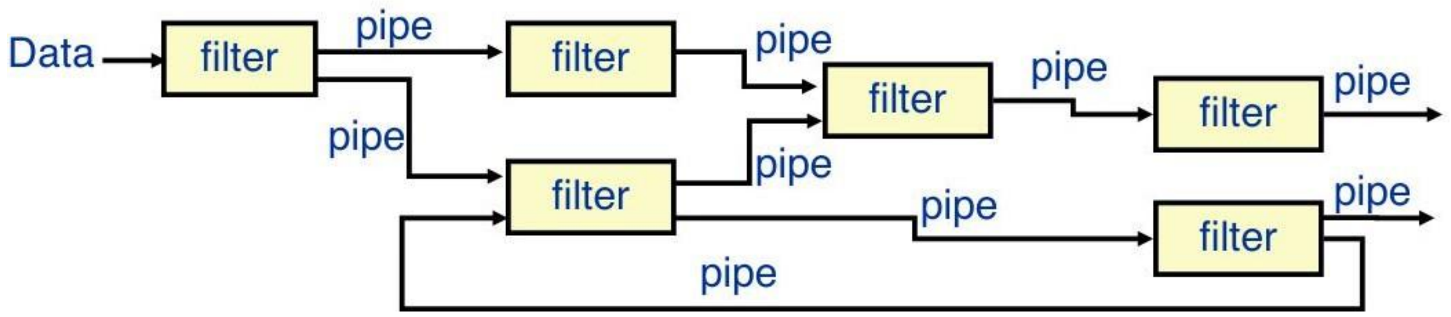
- Set of components
- Set of connectors between them
  – Communication, coordination, co-operation
- Constraints
  – How can components be integrated?
- Semantic models
  – What are the overall properties based on understanding of individual component properties?

Architecture Patterns

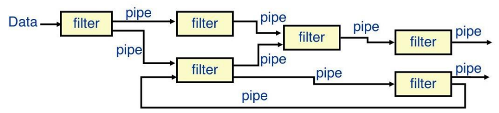
- Common program structures:
  1. Pipe and Filter
  2. Event-based
  3. Layered

Pipe and Filter

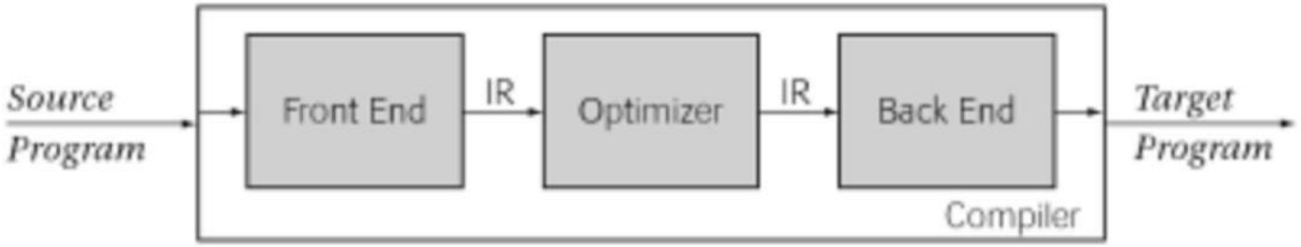
- A pipeline contains a chain of data processing elements
  - –The output of each element is the input of the next element (usually with some buffering in between)



Pipe and Filter (cont.)



Example: *Compiler Optimization* [Cooper]



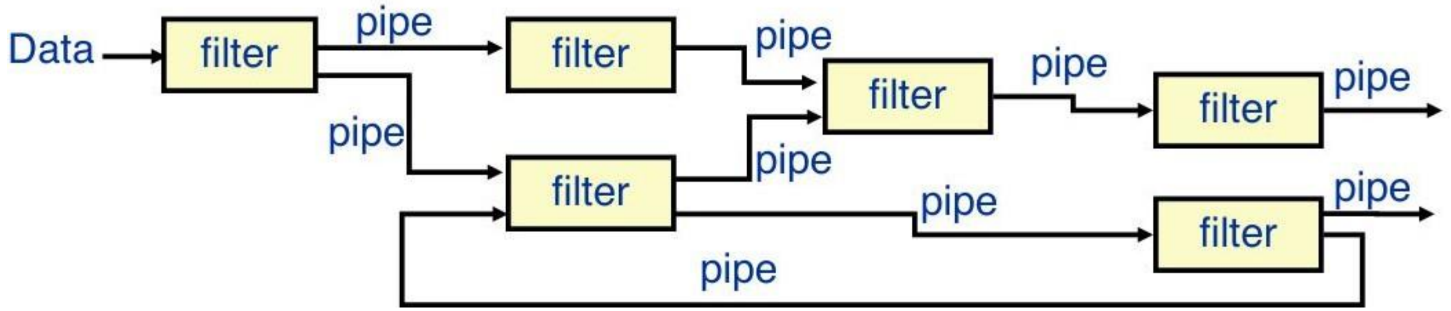
Other examples:

- **UNIX pipes**
- **Signal processors**
- **etc…**

Pipe and Filter: Pros and Cons

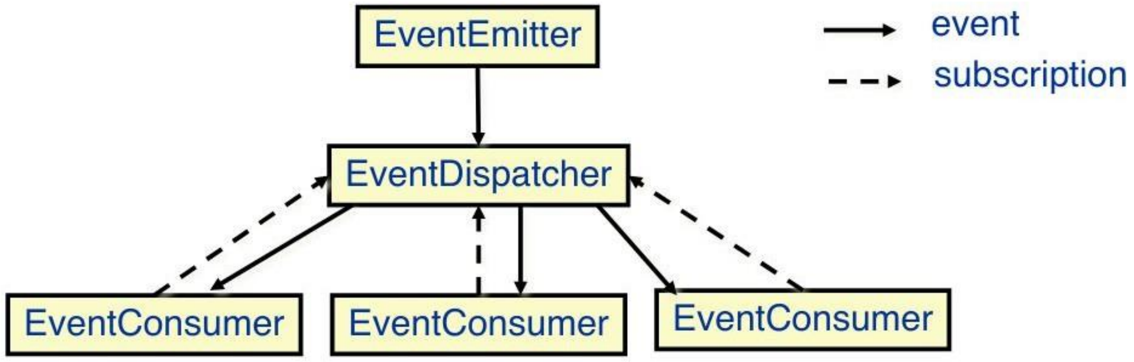
Pros:
- Easy to add or remove filters
- Filter pipelines perform multiple operations concurrently

Cons:
- Difficult to handle errors
- May need encoding/decoding of input/output

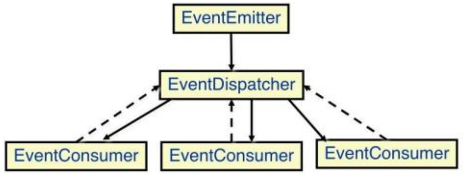


Event-Based Architecture

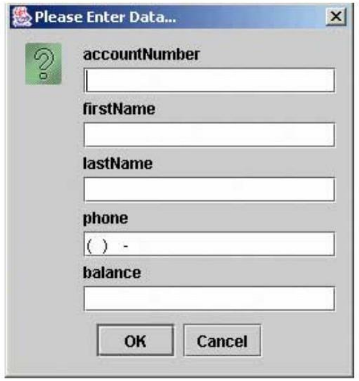
- Promotes the production, detection, consumption of, and reaction to events
- *Event-driven programming*



Event-Based (cont.)



Examples: *Graphical User Inerfaces*



Another reason why UI design (last lecture) is important!

Other examples:

- **Mobile apps**
- **Robotics**
- **Video games**
- **AR/VR**
- **etc.**

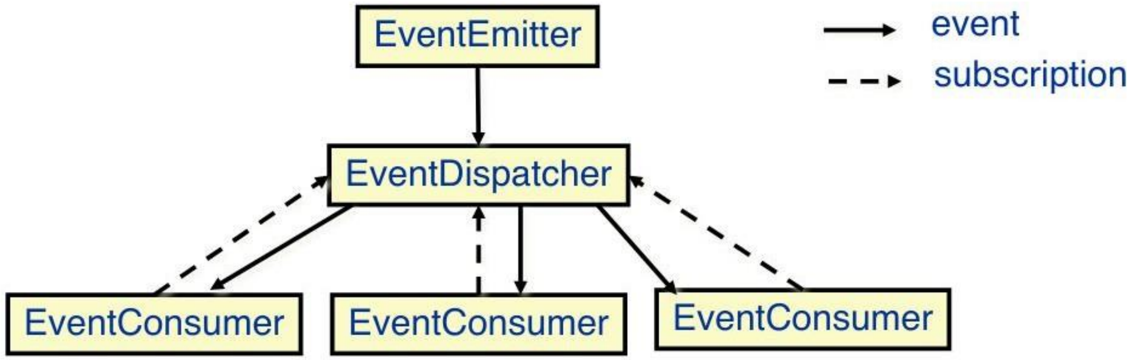
Event-Based: Pros and Cons

Pros:

- Anonymous handlers of events
- Support reuse and evolution, new features easy to add

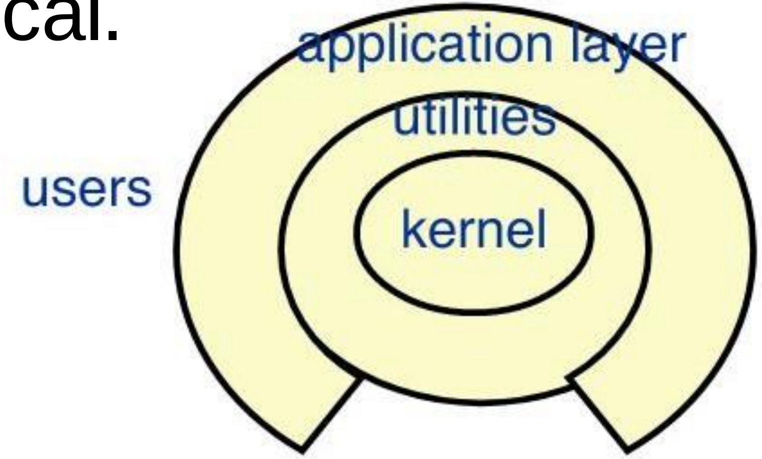
Cons:

- Components have no control over order of execution

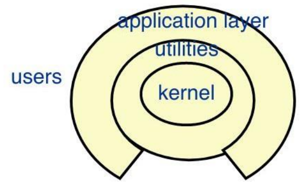


Layered/Tiered Architecture

- Multiple layers are defined to allocate responsibilities of a software product
- The communication between layers is hierarchical.



Layered/Tiered Arch. Variants

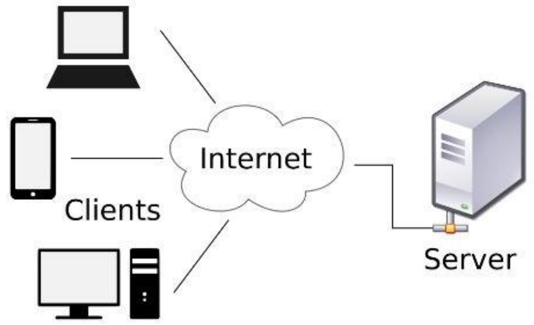


- **2-layer architecture**
  - Client-Server Architecture
  - Data-Centric Architecture

- **3-layer architecture**
  - Model-View-Controller

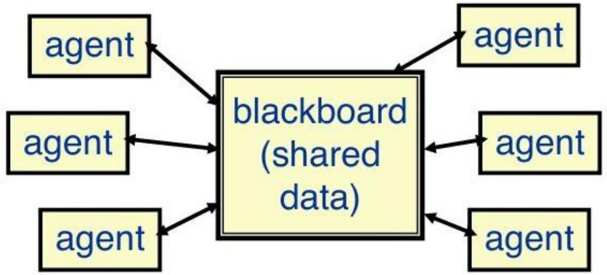
Client-Server Architecture

- Partition tasks or workloads between the providers and consumers of service or data (multiple hardware)
- Same system, different hardware, network communication



Data-Centric Architecture

- A data store resides at the center to be accessed frequently by agents
- Blackboard sends notification to subscribers when data of interest changes

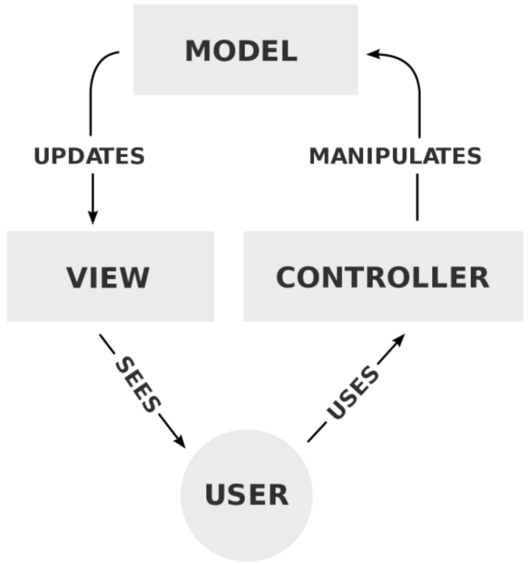


2-Layer: Examples, Pros, and Cons

- Examples
  - Distributed file systems, version control systems,...
- Pros
  - Low requirements for agents
  - Easy to add/change agents
- Cons
  - Blackboard/Server can be a bottleneck
  - Data integrity

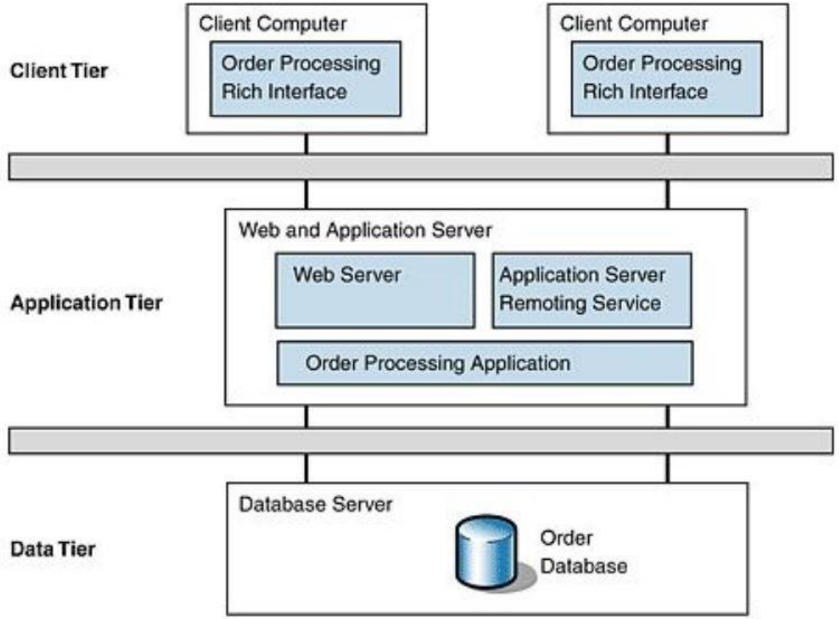
Model-View-Controller Architecture

- Model-View-Controller
  - Includes UI (view) to interact with users
  - Store and retrieve information as needed



MVC: Example

- Most modern web applications


Client Tier
Client Computer
Order Processing Rich Interface
Client Computer
Order Processing Rich Interface
Application Tier
Web and Application Server
Web Server
Application Server Remoting Service
Order Processing Application
Data Tier
Database Server
Order Database

MVC: Pros and Cons

Pros:

- Support increasing levels of abstraction during design
- Support reuse and enhancement

Cons:

- The performance may degrade
- Hard to maintain

How to Do Architecture Design?

When decomposing a system into subsystems, take into consideration:

- how subsystems share data
  – data-centric or data-distributed
- how control flows between subsystems
  – as scheduled or event-driven
- how they interact with each other
  – via data or via method calls

Next Time…

Design Patterns Workshop on Friday (03/22)

HW3 due Monday (03/22 at 11:59pm)

References

- Dr. Chris Brown
- RS Pressman. *“Software engineering: a practitioner's approach”.*
- Cast Software *“What is Software Architecture?”*. <https://www.castsoftware.com/glossary/what-is-software-architecture-tools-design-definition-explanation-best>
- K.D. Cooper, L. Torczon, *“Engineering a Compiler”.* Theo Mandel. *“Golden Rules of User Interface Design”.* <https://theomandel.com/resources/golden-rules-of-user-interface-design/>
- <https://medium.com/swlh/ordering-food-and-the-mvc-architecture-d5cbf3859d60>
- Na Meng and Barbara Ryder, Chris Parnin, Sarah Heckman