Introduction to Software Architecture

Software Architecture

Software architecture is the process of designing the global organization of the system, specifically:

- Dividing the system into subsystems
- Determining how subsystems interact with one another (when and with whom)
- Identifying how the subsystems are deployed (e.g. into same/different processes or machines altogether!)

Software architecture is high-level design:

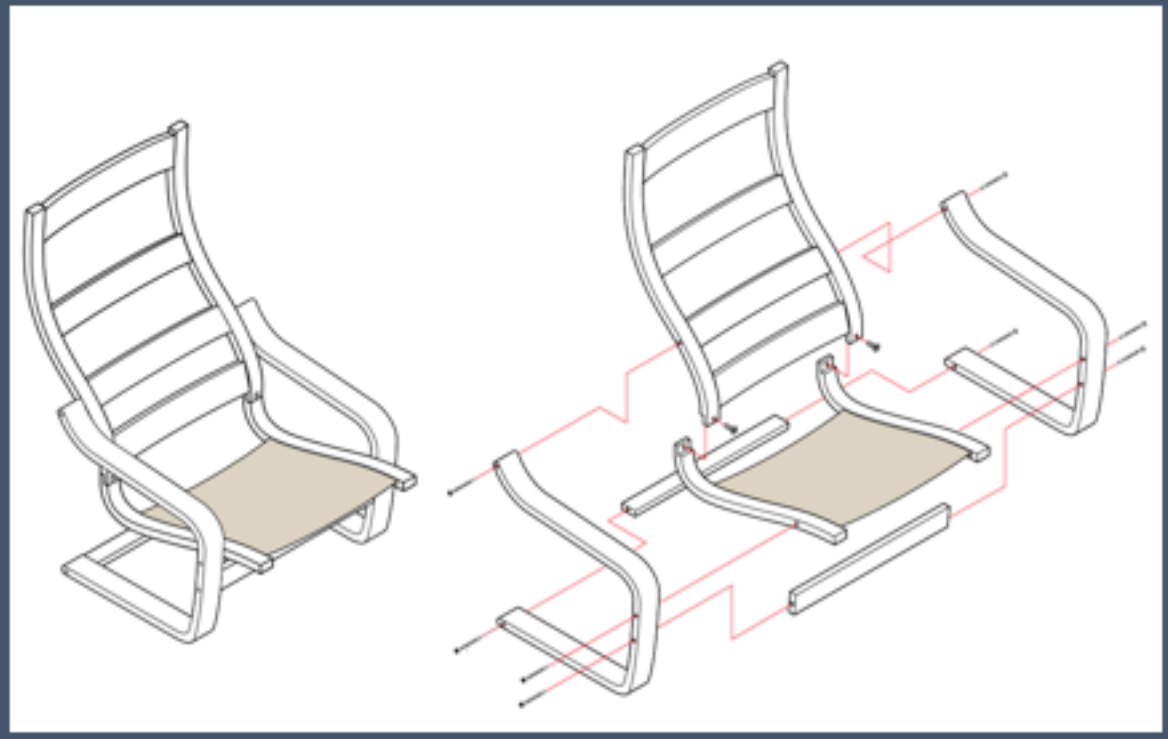
- Abstract away details of individual classes
- Subdividing system into manageable subsystems
- Considering how these pieces fit together

Why is an architecture important?

Supports understanding of the system

Allows sub-teams or individuals to work on subsystems in isolation

Enables re-use and re-usability



Partitioning Considerations

Hardware constraints (e.g. external database server; scalability)

Computational cost (parallelizing expensive operations)

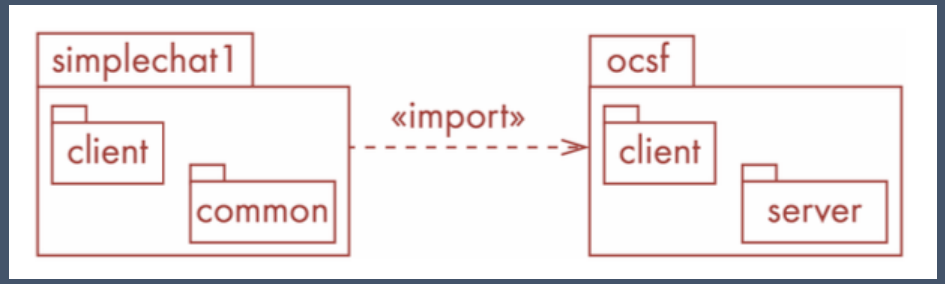
Logical divisions (e.g. UI, business logic, database, etc.)

Communication

- Intraprocess (same method, same thread, different threads)
- Interprocess (same node)
- Interprocess (different nodes)

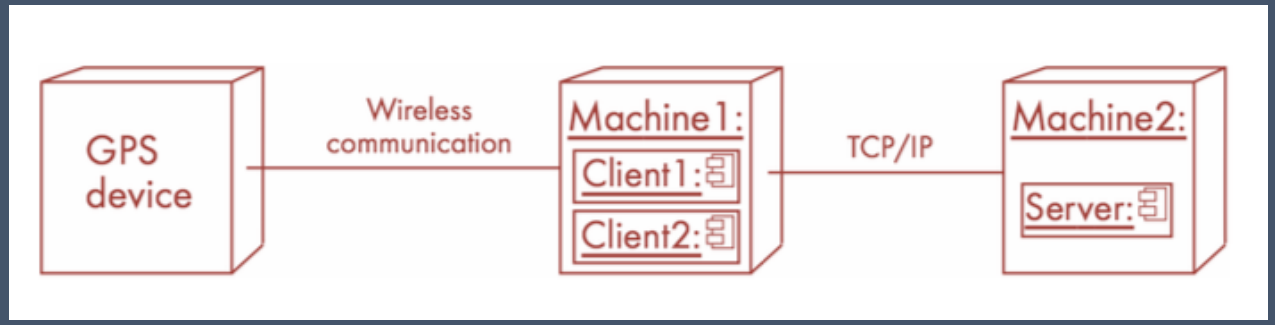
Some UML for describing architecture too!

Package diagram
(emphasizes logical structure)



Component diagram
(emphasizes interfaces/communication between components)

Deployment diagram
(emphasizes deployment model)



Architectural Models are judged on Stability

Stable: new features can be easily added with no or small changes to the architecture

So: an architectural model ought to be designed to be stable—this aids maintainability, extensibility, and reliability of the system in the long term

Common Architectural Styles

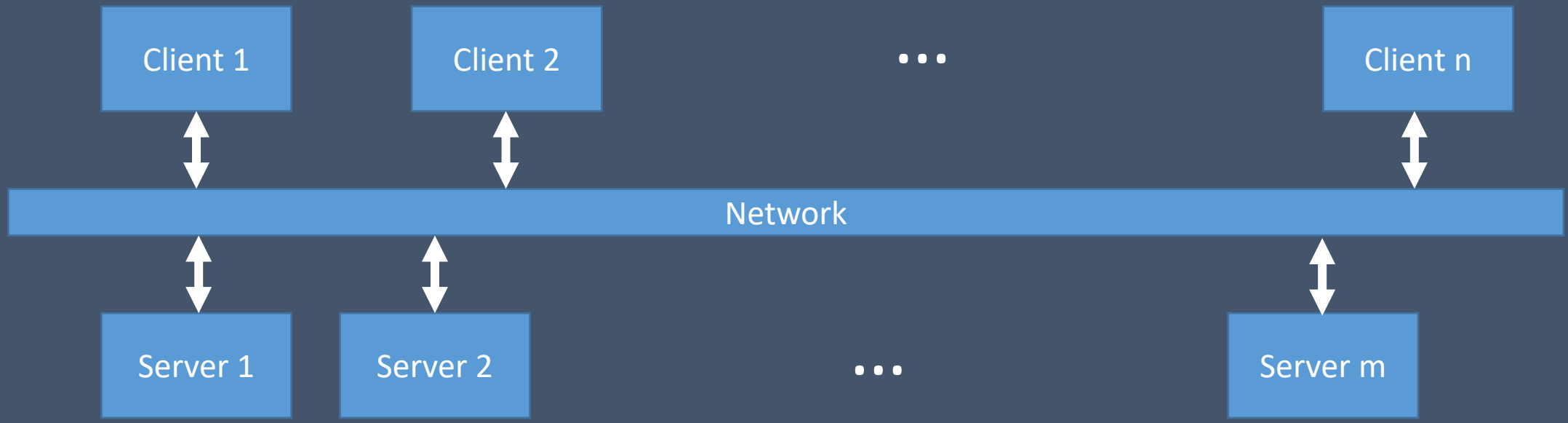
Client/server
Pipes-and-filters
Repository
Model-view-controller
Layered (three-tier, four-tier)
Peer-to-peer
Interpreter
Plugin
Component-based
Event-based
…

Client-Server Style

Stand-alone servers

Stand-alone clients

Network connects them



Examples

WWW: browser (clients) - servers

Email: IMAP, SMTP, email clients

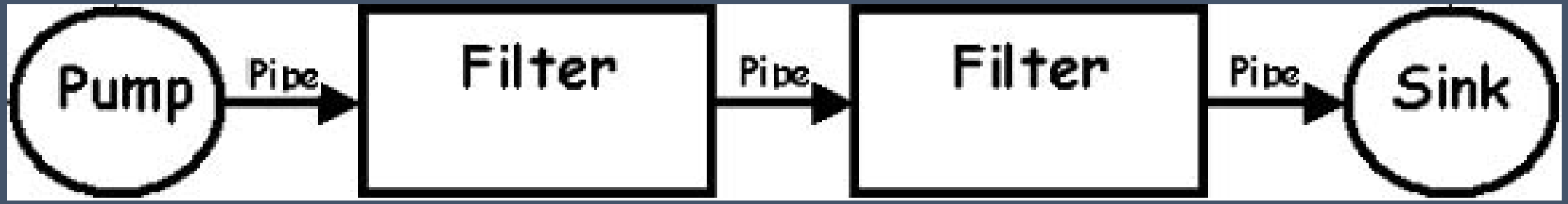
Games: Quake (and most derivatives)

Pipes and Filters

Pump = source

Filter = small, self-contained units that take input from a pipe, transform the input, and send it out to another pipe

Sink = something that



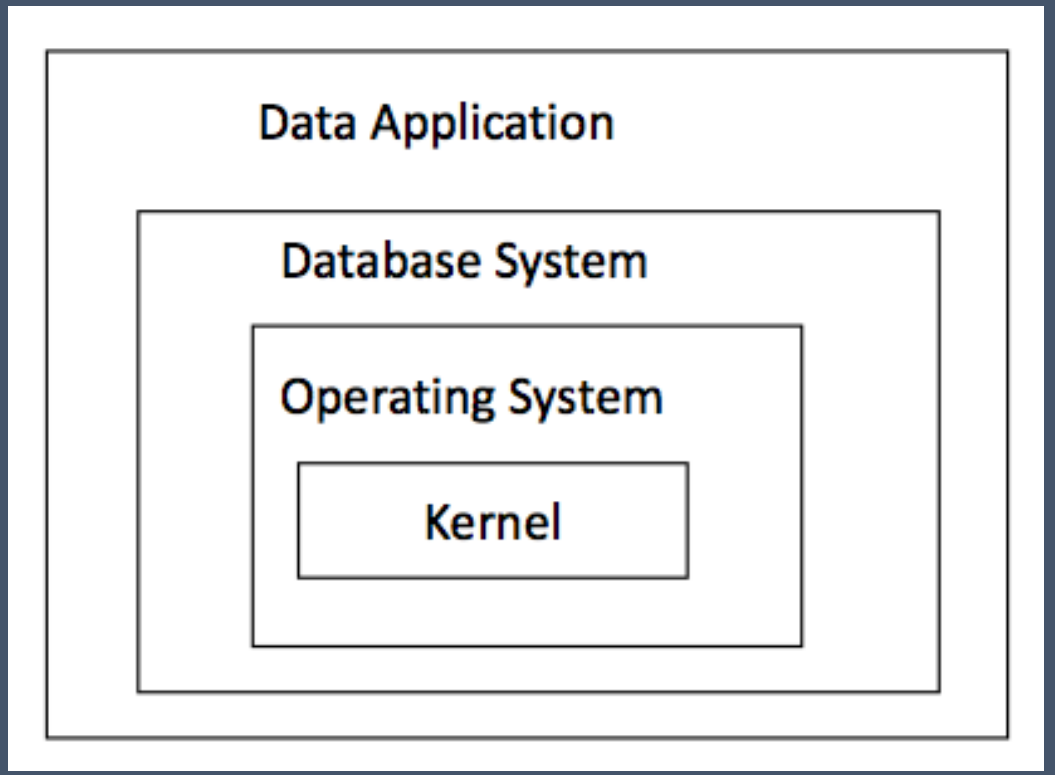
Layered Architecture

Sub-systems are organized into layers

Each layer:

- uses the services of the layer below
- provides services to layer above through well-defined interfaces

The terms “tier” and “layer” are interchangeable, for most people



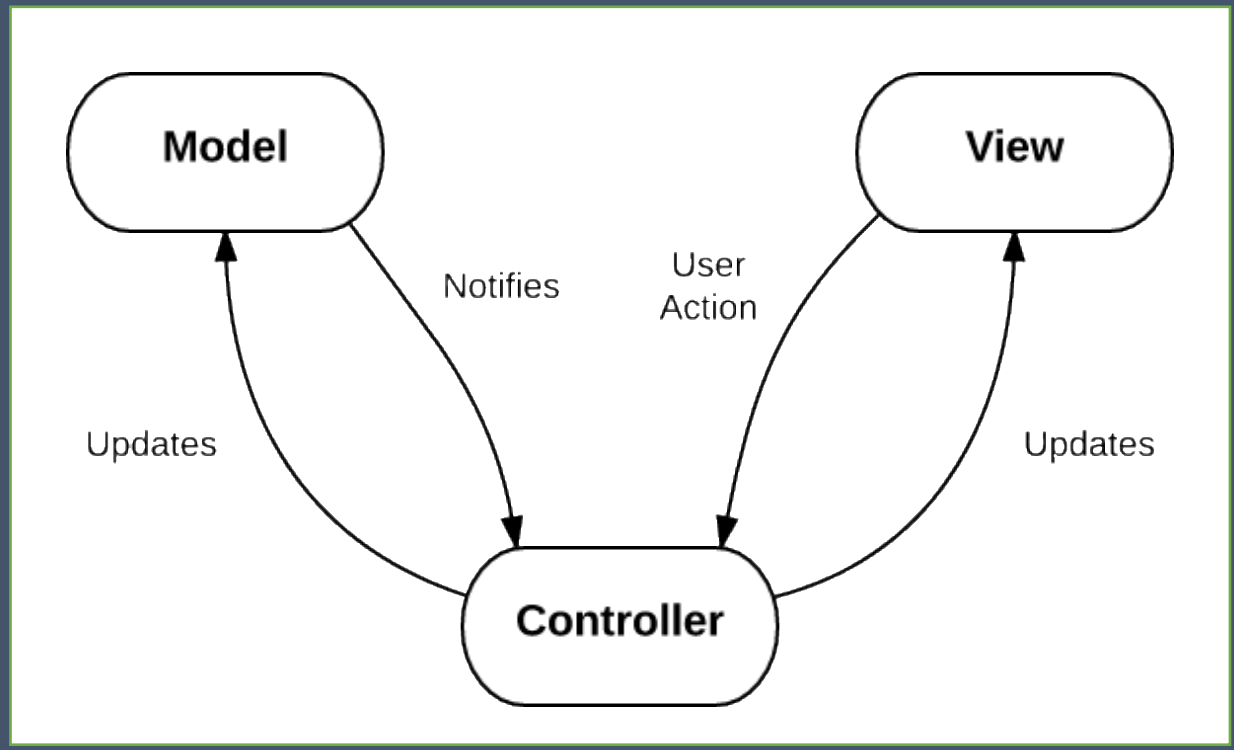
Model-View-Controller Architectural Style

Separation of M, V and C:

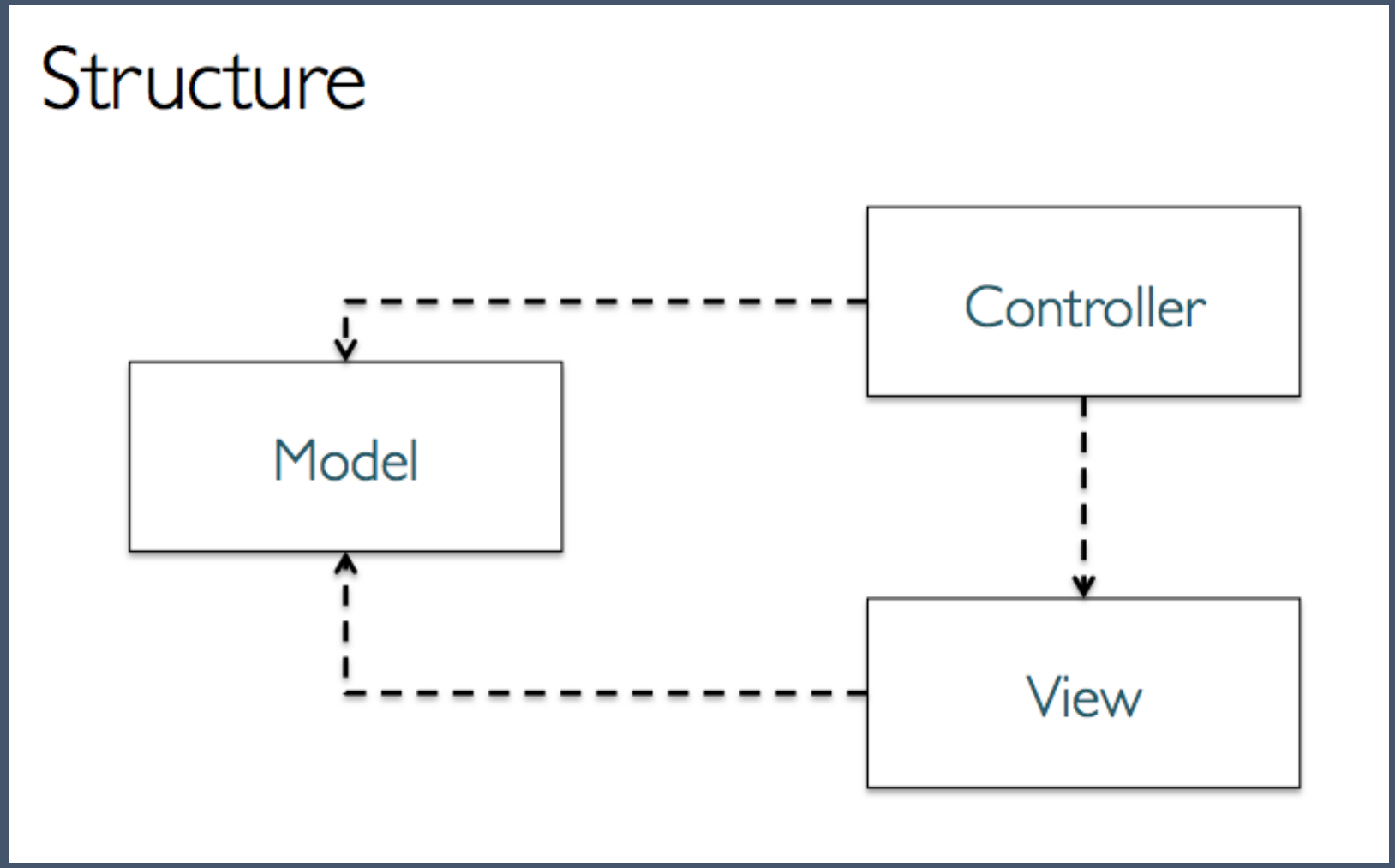
Model: manages behaviour of data; responds to requests about state (from View), responds to state change commands (Controller)

View: manages display of info

Controller: interprets user input, and updates model and view



MVC



Service Oriented Architectural Style

Loosely-coupled, autonomous, distributed services

Close adherence to a schema and contract, not class (i.e. it’s usually about data)

Applications are then mostly about composing services together