Announcements

- GCP Assignment
    - Check out the helpful resources on *Deploying .jar files on Google Cloud* posted under Additional Resources on class website
        - The key is setting up the *app.yaml* file
    - Enable the links to your API after the due date of the GCP assignment (October 11, 11:59 PM)
    - We will post an announcement as soon as your assignment has been graded so that you can disable it and conserve the GCP credits.

- Mid term Feedback Survey Released Today
    - Please provide express any concerns/comments you have about the course so far.
    - Your responses will be stored anonymously

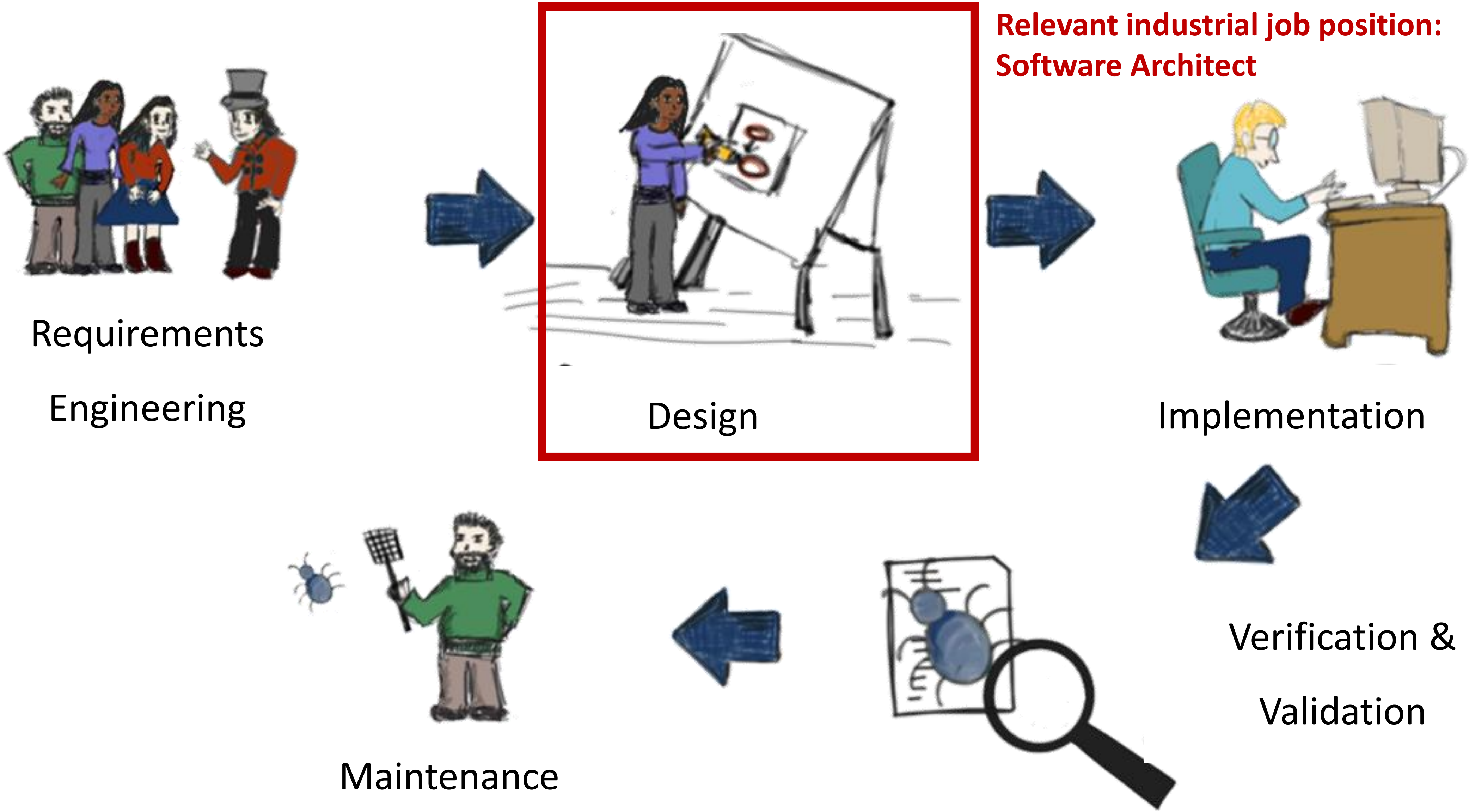
CS3300 Introduction to Software Engineering

Lecture 11: Software Architecture & Design

Nimisha Roy ▸ nroy9@gatech.edu

Slides adapted from Alessandro Orso

Traditional Software Development Phases



Requirements Engineering

Design

Relevant industrial job position: Software Architect

Implementation

Verification & Validation

Maintenance

What is Software Architecture?

Perry and Wolf
SWA = { Elements, Form, Rationale}

What (processes, data, connectors) ; How (properties, relationship between elements) ; Why (justification for elements and relationships)

Shaw and Garland
SWA = [is a level of design that] involves

- Description of elements from which systems are built
- Interactions among those elements
- Patterns that guide their composition
- Constraints on these patterns

A general definition of SWA

Set of principal design decisions about the system

Blueprint of a software system

- Structure
- Behavior
- Interaction
- Nonfunctional properties

Temporal Aspect

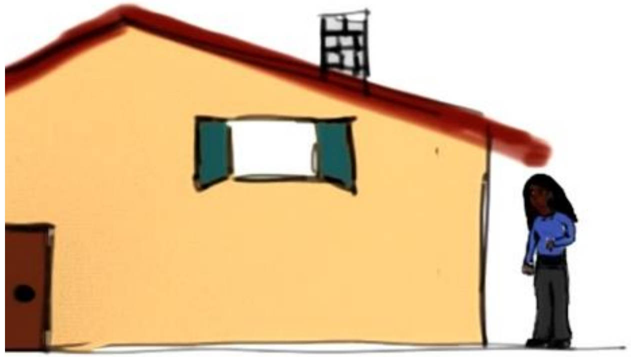
A SWA is not defined at once, but iteratively, over time

At any point in time, there is a SWA, but it will change over time

Design decisions are made, unmade, and changed over a system's lifetime.

Prescriptive vs. Descriptive Architecture

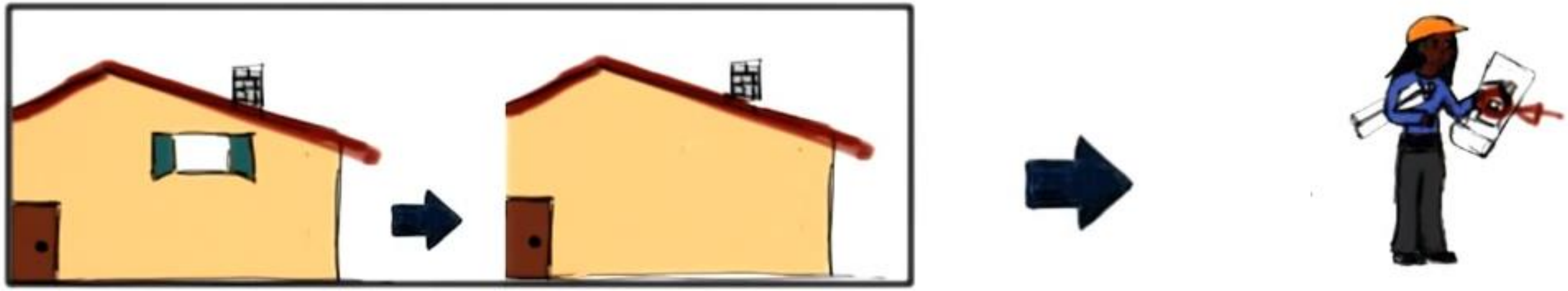
A prescriptive architecture captures the design decisions made prior to the system's construction
=> as- conceived SWA



A descriptive architecture describes how the system has actually been built
=> as- implemented SWA

Architectural Evolution

When a system evolves, ideally its prescriptive architecture should be modified first



In practice, this rarely happens

- Developer’s sloppiness
- Short deadlines
- Lack of documented prescriptive architectures

Architectural Degradation

Architectural drift : Introduction of architectural design decisions orthogonal to a system’s prescriptive architecture

Architectural erosion : Introduction of architectural design decisions that violate a system’s prescriptive architecture

Architectural Recovery

Drift and Erosion => Degraded architecture

Keep tweaking the code (typically disastrous)

Architectural recovery: determine SWA from implementation and fix it

Example Quiz

Which of the following statements is true.

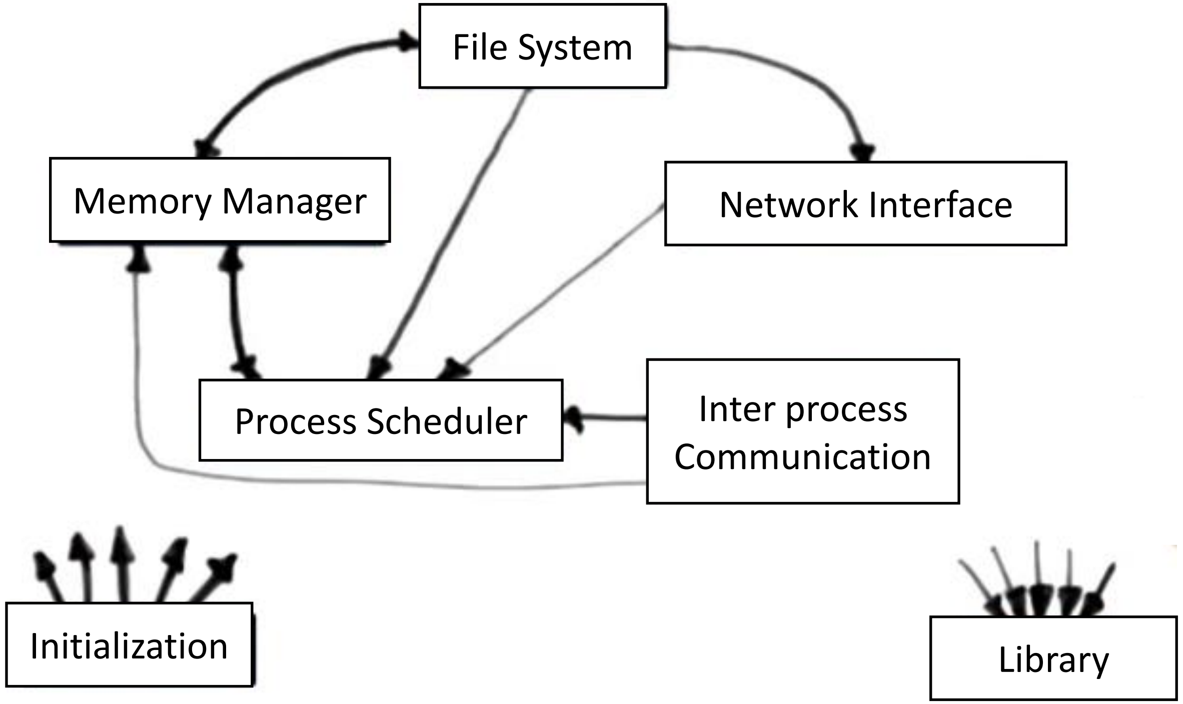
[   ] Prescriptive architecture and descriptive architecture are typically the same.

[✓] Architectural drift results in unnecessarily complex architectures.

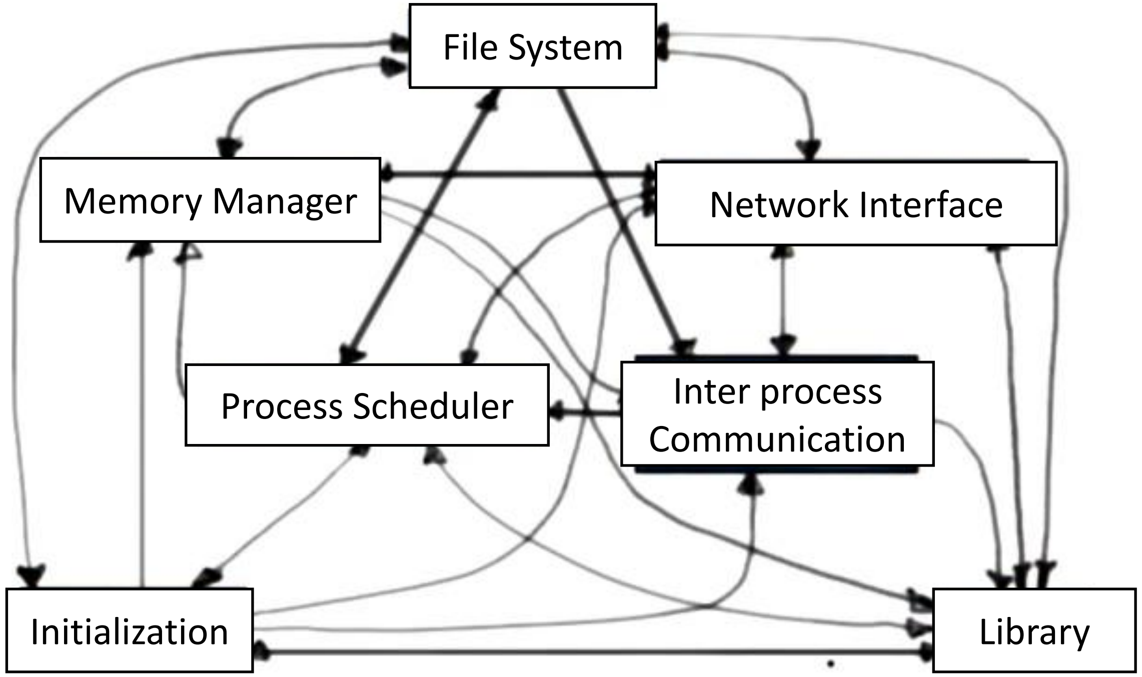
[   ] Architectural erosion is less problematic than architectural drift.

[   ] The best way to improve a degraded architecture, is to keep fixing the code until the system starts looking and behaving as expected

An example from the Linux Kernel



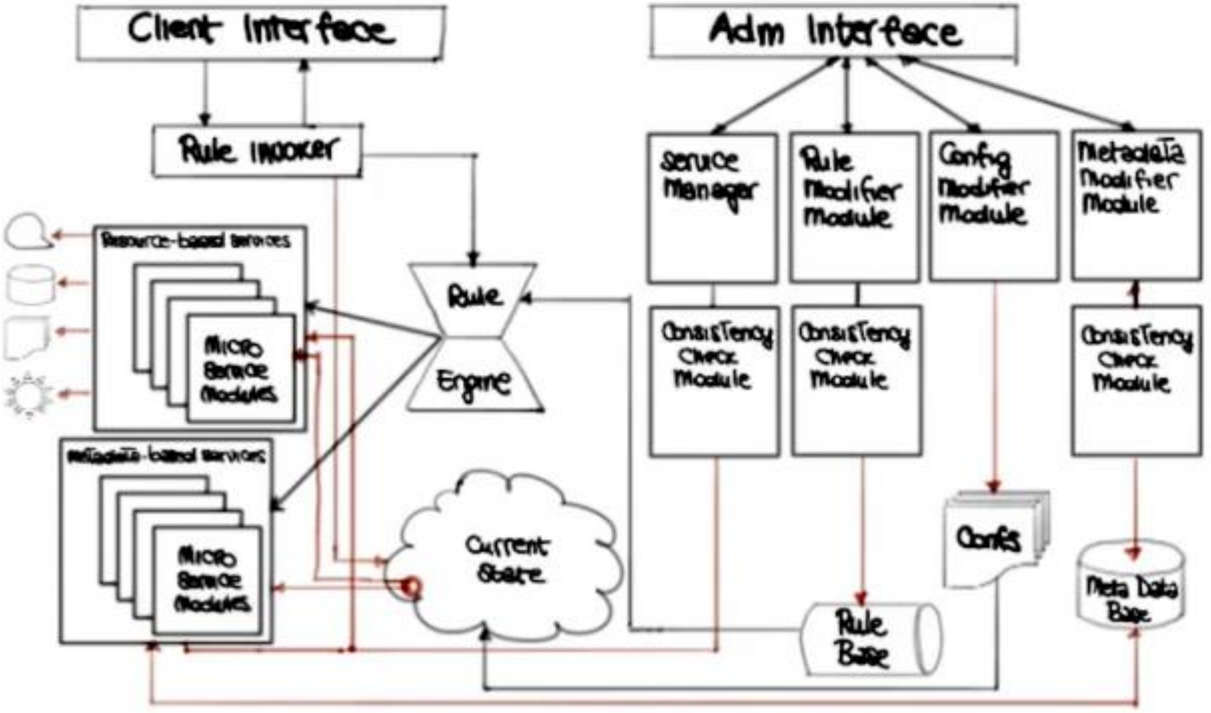
Prescriptive Architecture



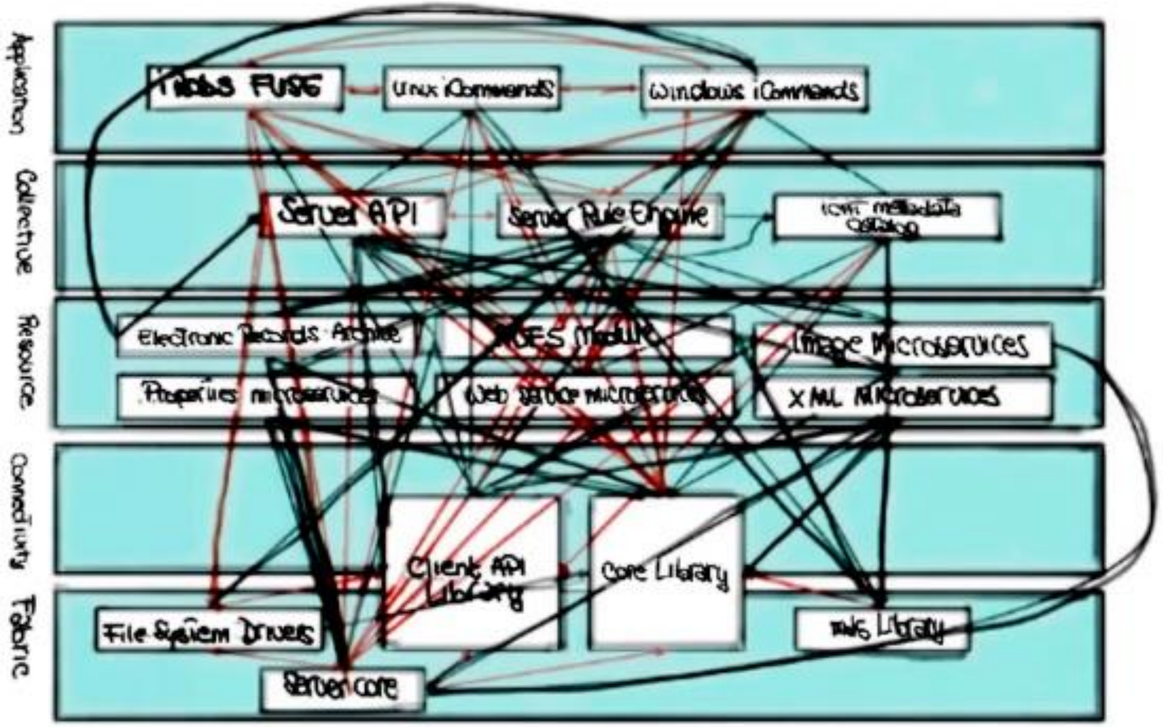
Descriptive Architecture

Another example: iRODS

Data grid system that was built by a biologist. It's a system for storing and accessing big data.



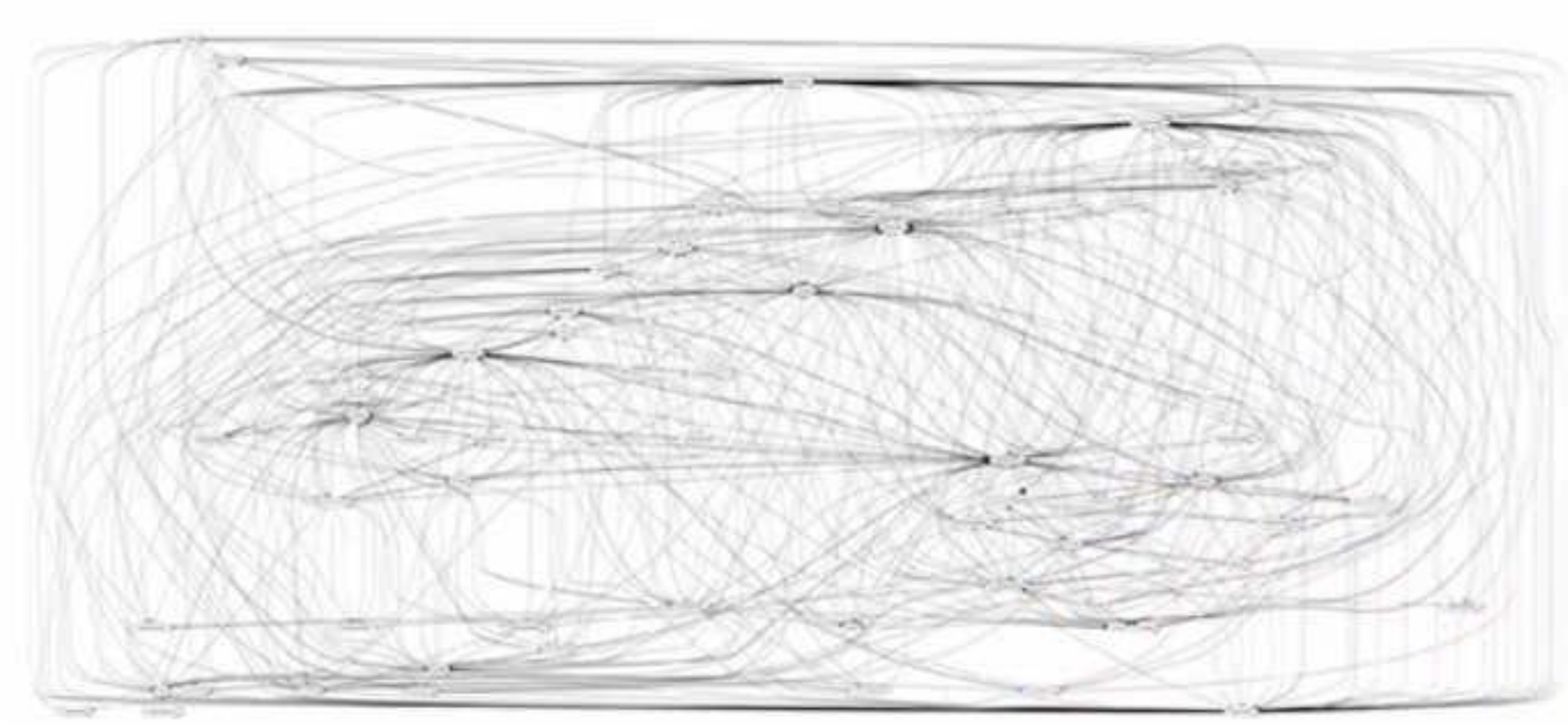
Prescriptive Architecture



Descriptive Architecture

More examples: Hadoop

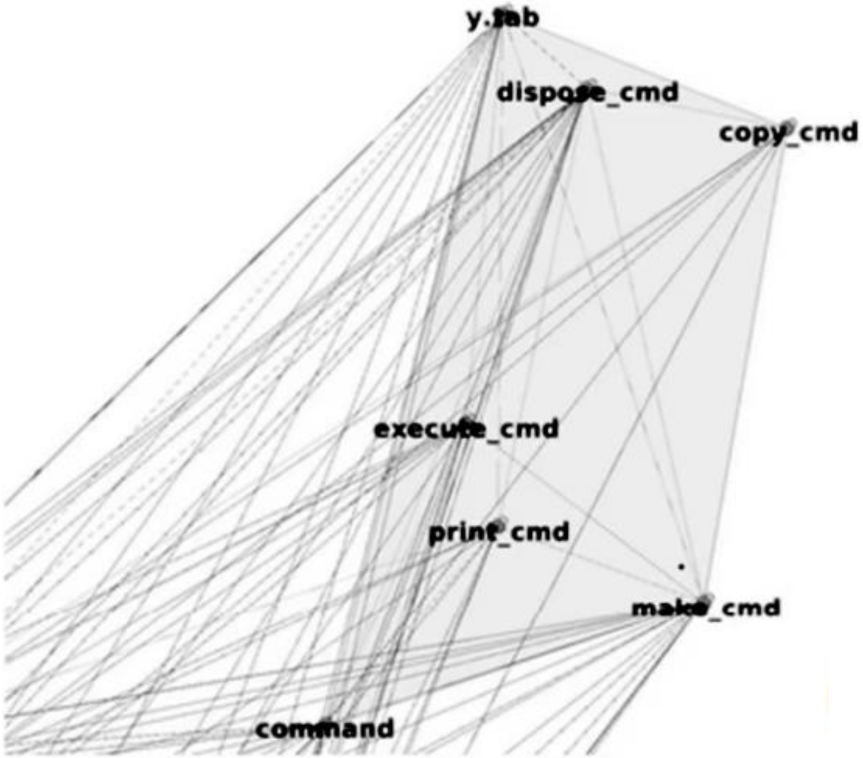
Open-source software framework for storage and large-scale processing of data sets



Descriptive Architecture

Final example: Bash

Unix shell written as a free software replacement for the traditional Bourne shell



Lack of cohesion in the component

High coupling among components

Descriptive Architecture of the command component of Bash.

Example Quiz

Which of the following are ideal characteristics of an architectural design

[✓] Scalability

[ ] Low cohesion

[✓] Low coupling

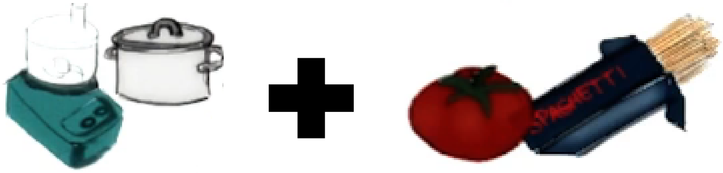
Software Architecture's Elements

A software architecture typically is not a monolith composition, but an interplay of different elements

Processing Elements

Data Elements

Interaction elements

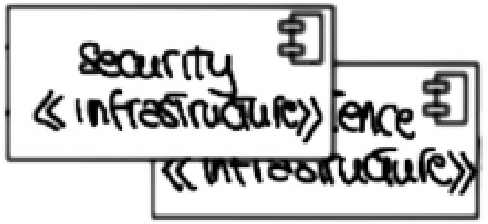


Components

Connectors

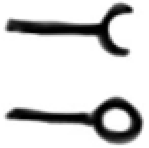
Configuration

Components, Connectors, and Configurations

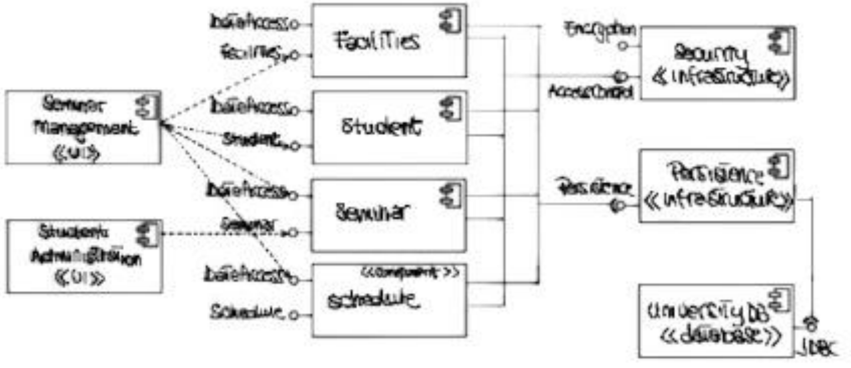


Software Component: Architectural Entity that

- encapsulates a subset of the system's functionality and/or data
- Restricts access to that subset via. an explicitly defined interface

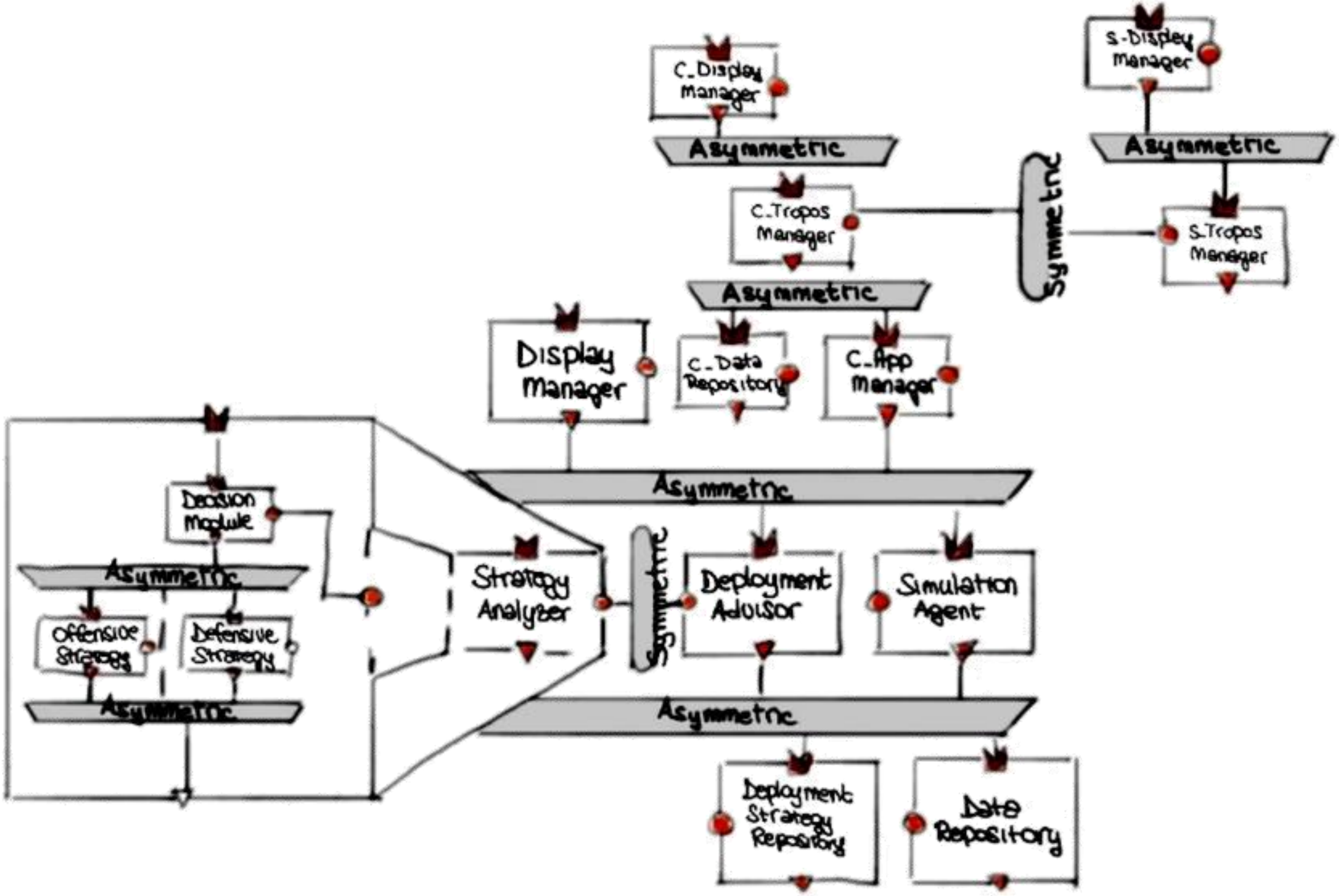


Software connector: Architectural entity effecting and regulating interaction among components

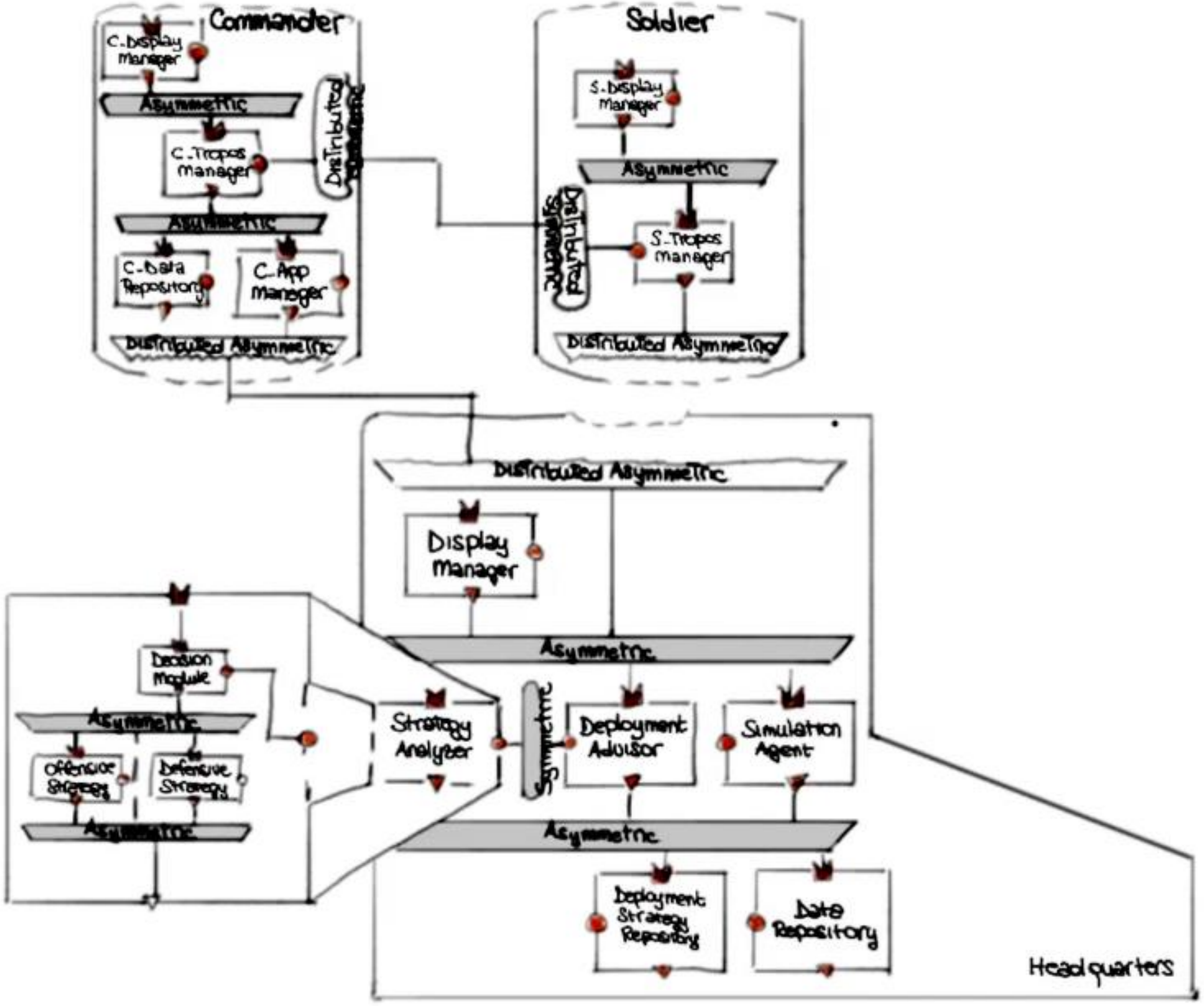


Architectural configuration: Association between components and connectors of a software architecture

An example configuration



Deployment Architectural Perspective



- A system cannot fulfill its purpose until it is deployed.
- Deploying a system involves physically placing the system's executable modules on the hardware devices on which they are supposed to run.
- Deployment view of an architecture can be critical in assessing whether the system will be able to satisfy its requirement.
- Enough memory available? Power consumption profile handled by hardware? Enough network bandwidth for interactions?

Architectural Styles

An architectural style defines “a family of systems in terms of a pattern of structural organization; a vocabulary of components and connectors, with constraints on how they can be combined”

M. Shaw and D. Garlan, 1996

Basically, named collection of architectural design decisions applicable in a given context.

Architectural Styles

Pipes and Filters
(Unix pipes)

Event – Driven
(GUI)

Publish- Subscribe
(Twitter)

Client- Server
(Email)

Peer - to – Peer
(Skype)

Representational State Transfer
(WWW)

Example Quiz

Consider the following architectural styles that we just saw:
pipes and filters (A),
event driven (B),
publish-subscribe (C),
client-server (D),
peer-to-peer (E),
REST (F). Mark which style(s) characterizes the following systems.

[F, D] World Wide Web

[D, E] Skype

[B, C] Android OS

[D] Dropbox

Peer-to-Peer (P2P) Architectures

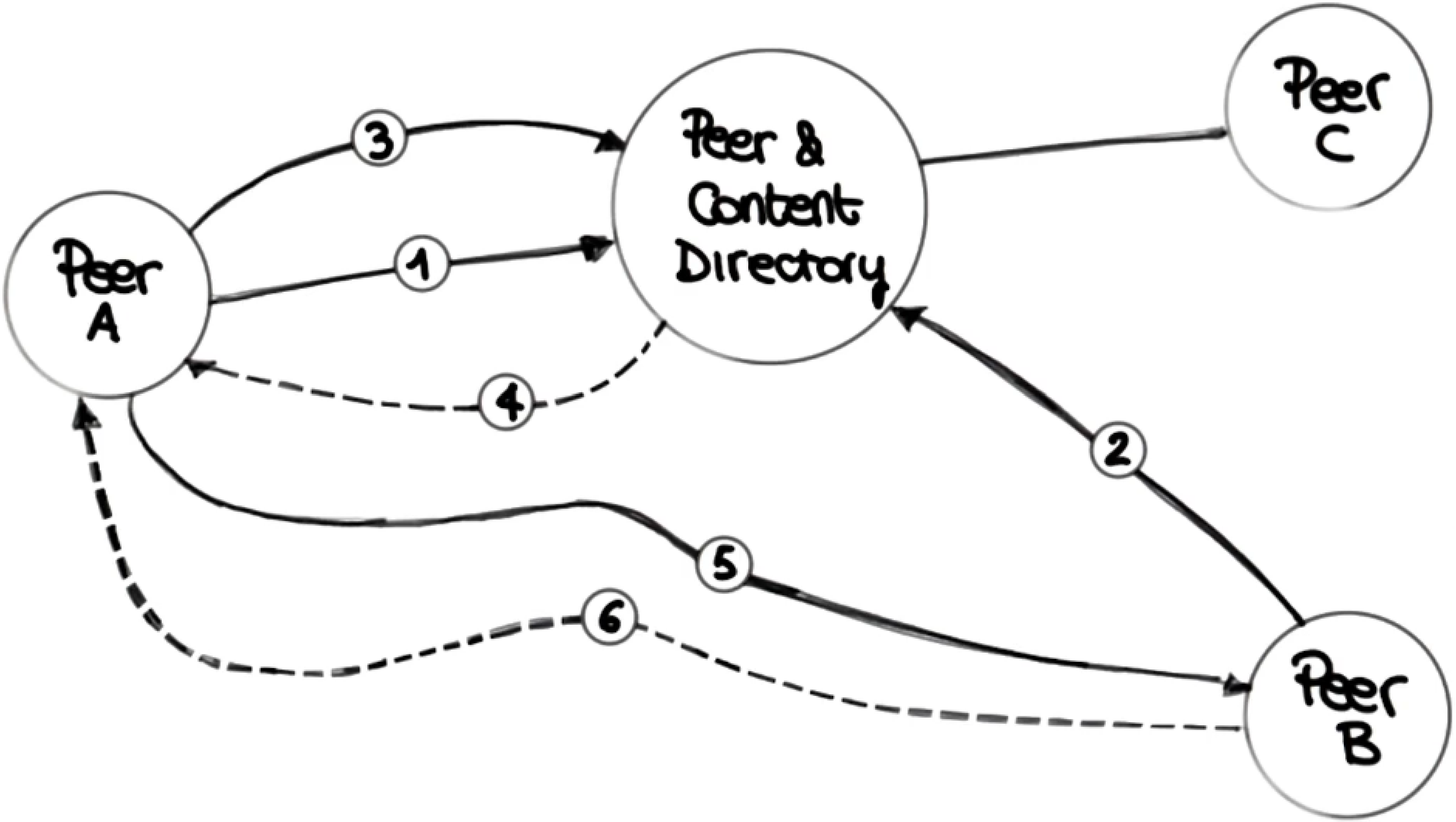
Decentralized resource sharing and discovery

Two representative examples:

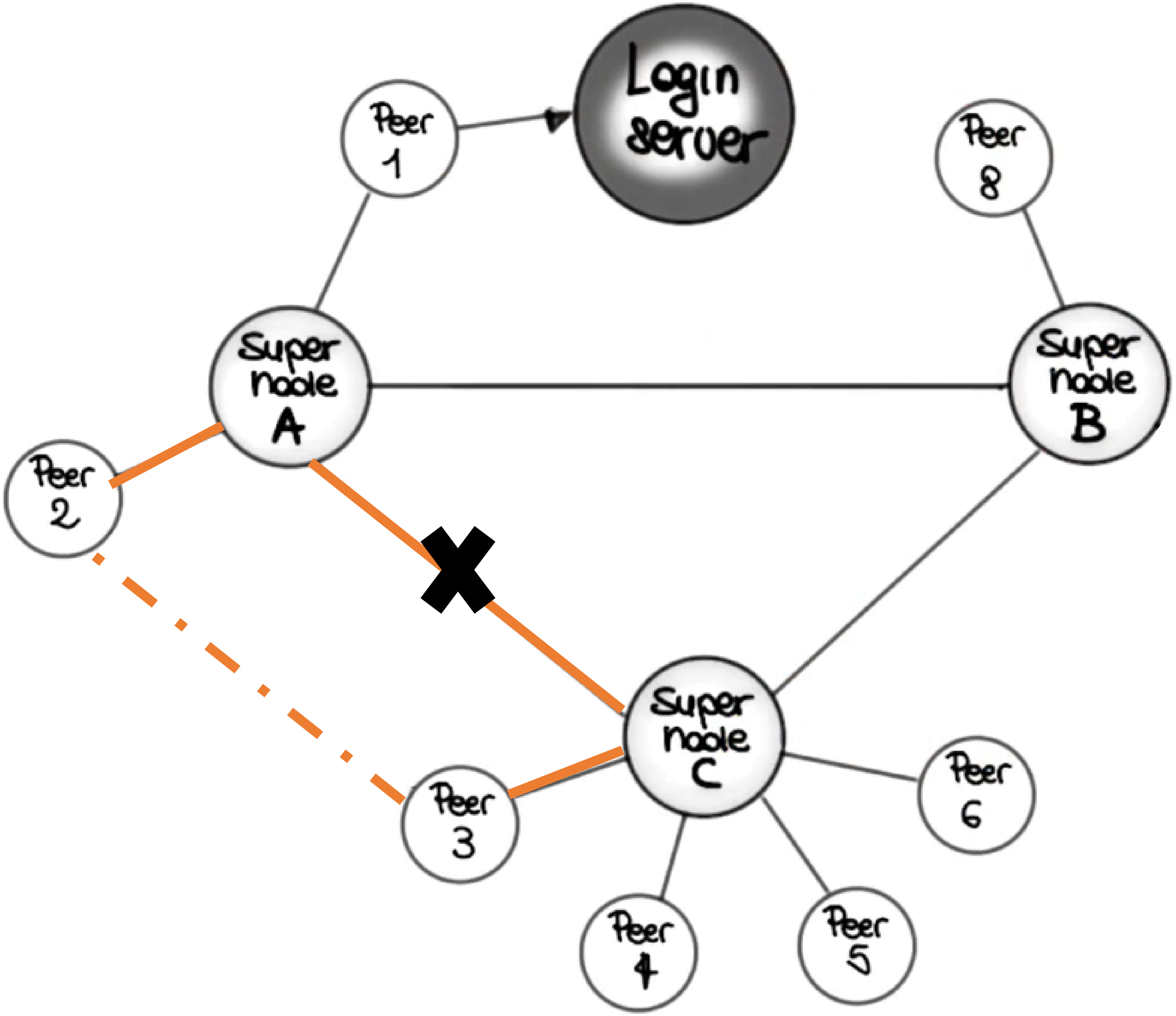
Napster

Skype

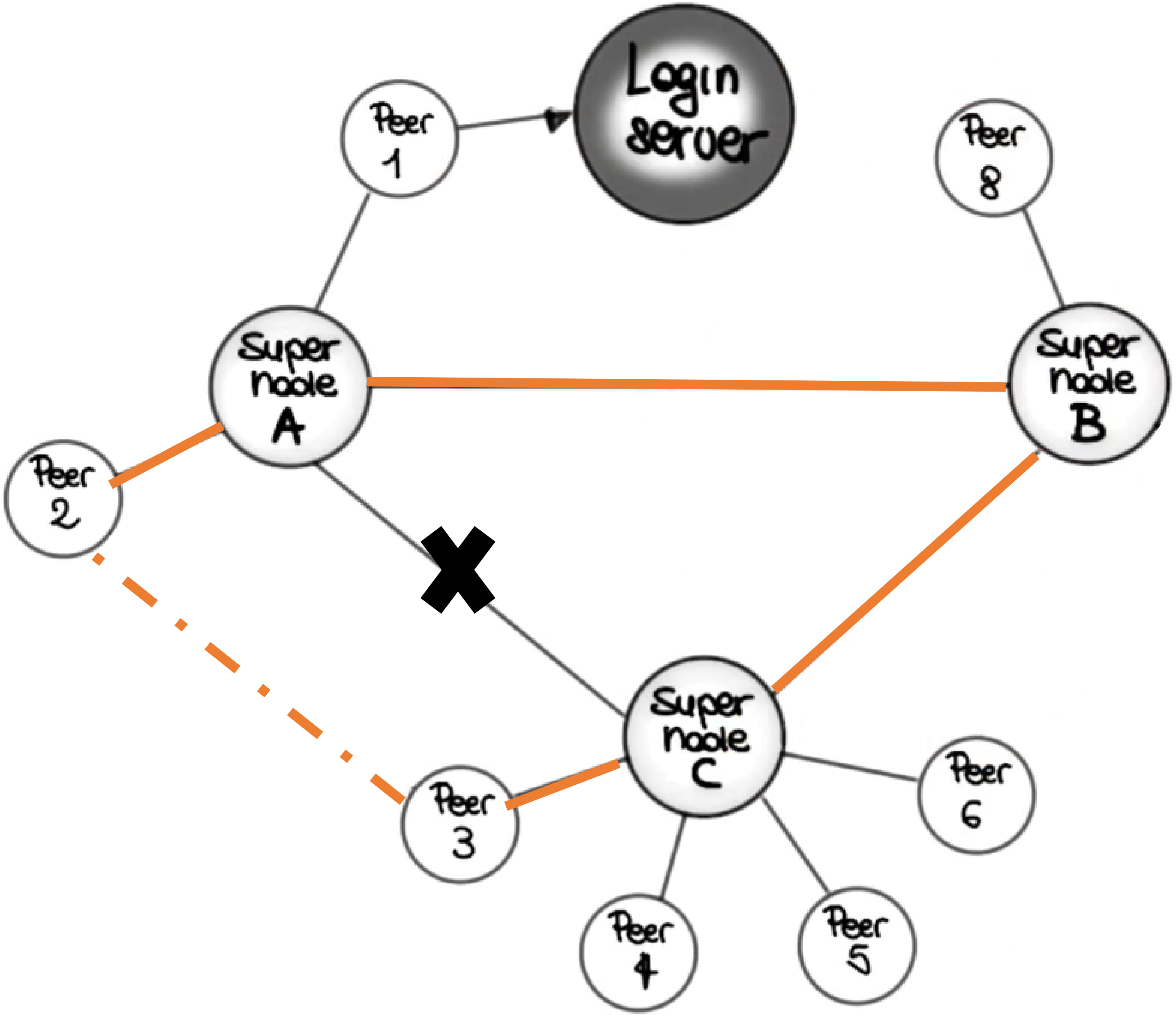
NAPSTER



SKYPE


Login server
Peer 1
Peer 8
Super node A
Super node B
Peer 2
Super node C
Peer 3
Peer 6
Peer 4
Peer 5

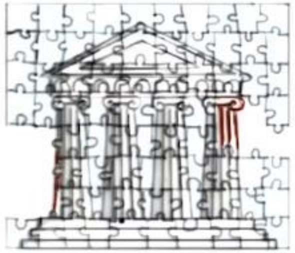
SKYPE


Peer 1
Login server
Peer 8
Super node A
Super node B
Peer 2
Super node C
Peer 3
Peer 6
Peer 4
Peer 5

Takeaways

A great architecture is a ticket to success

A great architecture reflects deep understanding of the problem domain



A great architecture normally combines aspects of several simpler architectures