Material and some slide content from:
- Emerson Murphy-Hill, Reid Holmes
- Software Architecture: Foundations, Theory, and Practice
- Essential Software Architecture
- Steve Easterbrook

Architectural Decomposition

Mei Nagappan

What is SW architecture?

- Definition:

> "The set of principal design decisions about the system"

- Blueprint for construction and evolution.
- Encompasses:
  - Structure
  - Behaviour
  - Non-functional properties

Components

- Elements that encapsulate processing and data at an architectural level.
- Definition:
  - Architectural entity that:
    - encapsulates a subset of functionality.
    - restricts access via explicit interface.
    - has explicit environmental dependencies.

Connectors

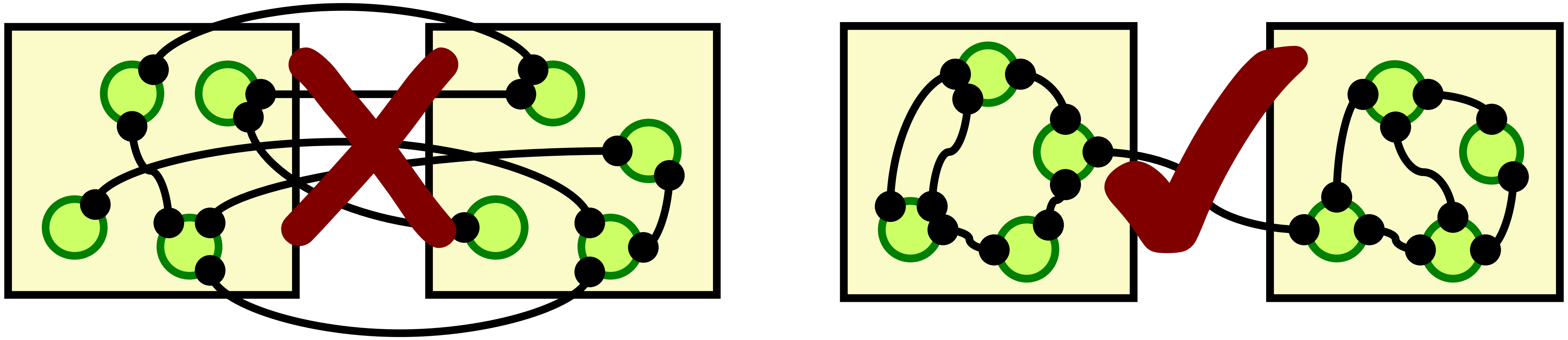
- Definition:
  - An architectural entity tasked with effecting and regulating interactions between components.
- Connectors are often more challenging than components in large heterogenous systems.
- Often consists of method calls, but be much more.
- Frequently provide application-independent interaction mechanisms.

Configurations

- Bind components and connectors together in a specific way.
- Definition:
  - An architectural configuration, or topology, is a set of specific associations between the components and the connectors of the system's architecture.
- Differentiates a bag of components and connectors from an implementable system.

Topological Goals

- Minimize coupling between components
  - The less components know about each other, the better (also known as information hiding).
- Maximize cohesion within each component
  - Components should be responsible for a logical service; extraneous functionality should not be present.



[Steve Easterbrook: http://www.cs.toronto.edu/~sme/CSC302/notes/04-package-diagrams.pdf]

Abstraction

- Complex problems can be approached by abstracting away unnecessary detail
- Focus on the key issues while eliding extraneous detail (some of these details will be pertinent during more detailed design activities)
- In software two classes of abstraction dominate:
  - Control abstraction
    - (e.g., structured programming)
  - Data abstraction
    - (e.g., abstract data types)

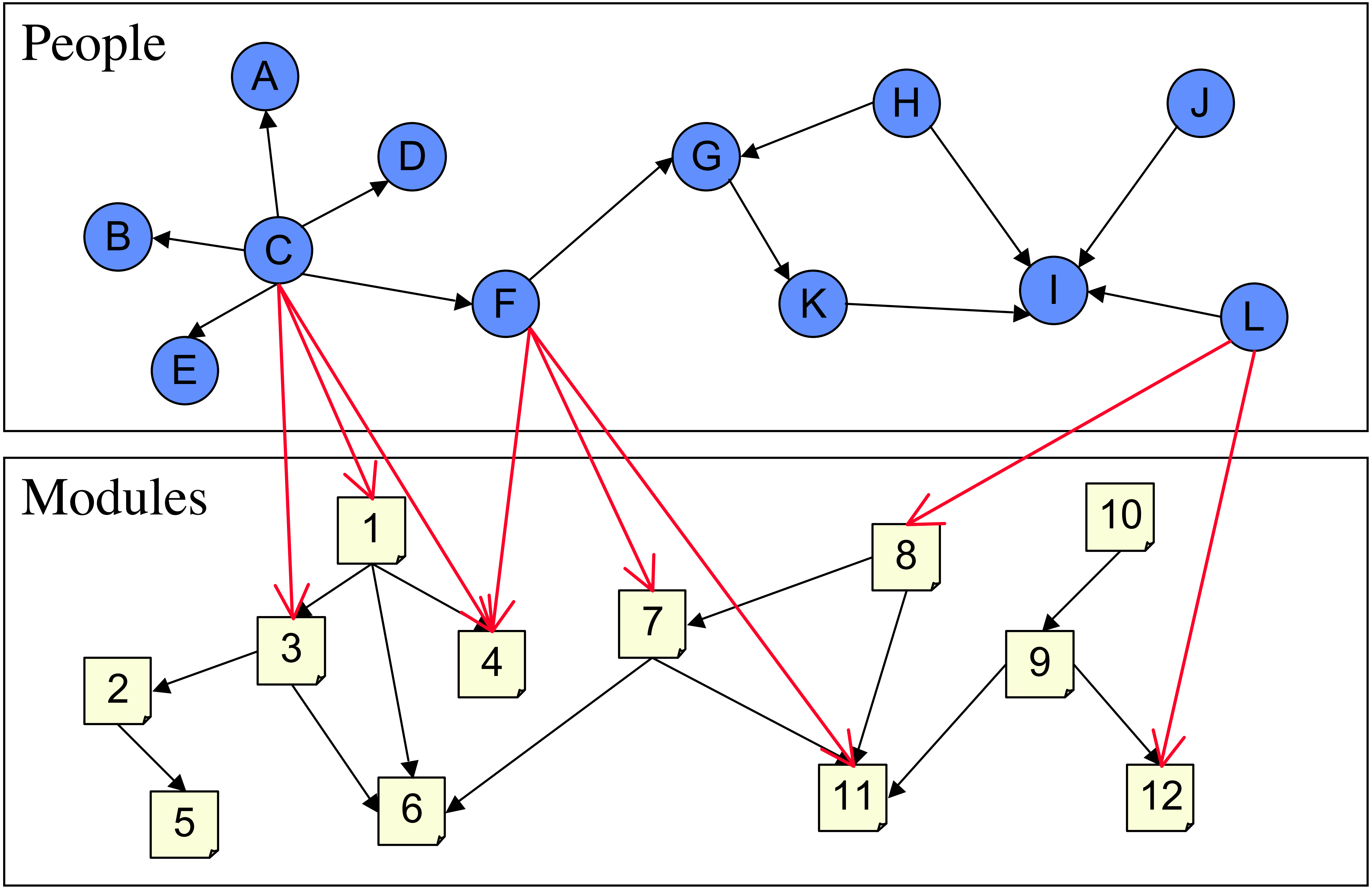
Decomposition

- Top-down abstraction is also called decomposition
    - Break problem into independent components
    - Describe each component
- Criteria for decomposition can include:
    - Implementing teams
    - Application domains (aka obvious partitions)
    - Parallelization
- Make typical cases simple, and exceptional cases possible

Conway’s Law

“The structure of a software system reflects the structure of the organization that built it”

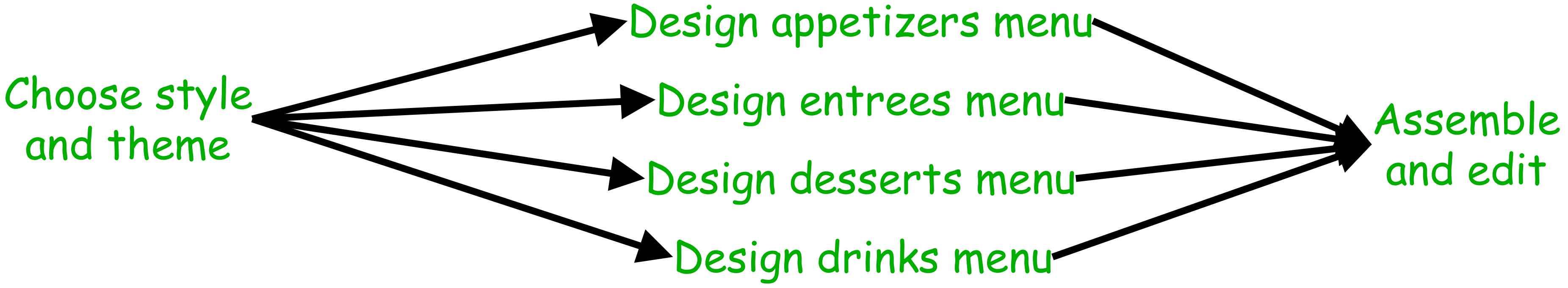
Conway’s Law



[Steve Easterbrook: http://www.cs.toronto.edu/~sme/CSC302/notes/04-package-diagrams.pdf]

Decomposition isn't always great

- ➜ **Decomposition can work well:**
  - E.g. designing a restaurant menu



Choose style and theme

Design appetizers menu

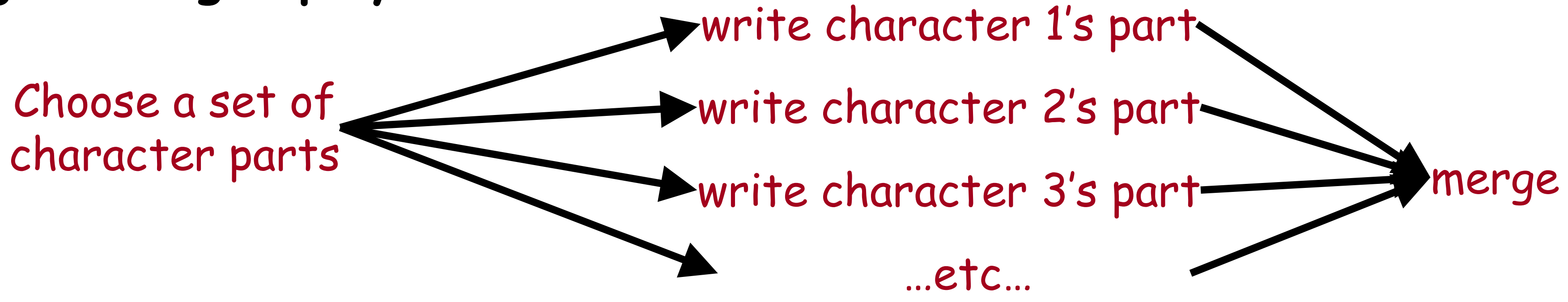
Design entrees menu

Design desserts menu

Design drinks menu

Assemble and edit

- ➜ **Decomposition doesn't always work**
  - E.g. writing a play:



Choose a set of character parts

write character 1's part

write character 2's part

write character 3's part

…etc…

merge

- ➜ **Decomposition isn't always possible**
  - for very complex problems (e.g. Managing the economy)
  - for impossible problems (e.g. Turning water into wine)
  - for atomic problems (e.g. Adding 1 and 1)

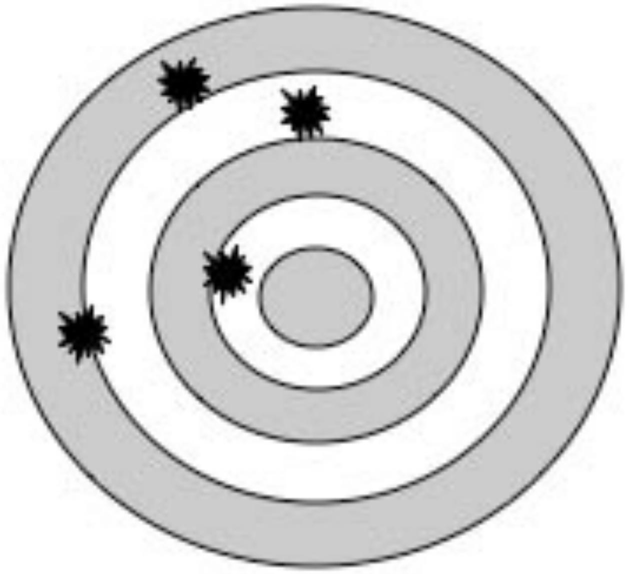
HTTP://WWW.CS.TORONTO.EDU/~SME/CSC444F/SLIDES/L05-DECOMPOSITIONABSTRACTION.PDF

Activity

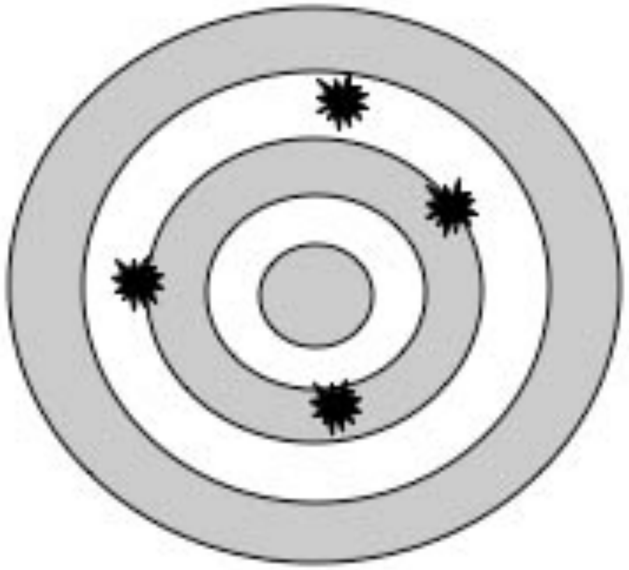
- Decompose the garage door opener example from last class.

Architectural representations

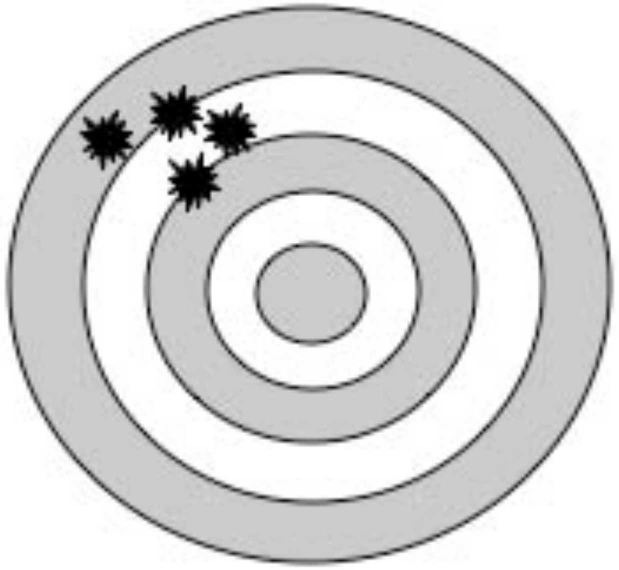
- Software architecture is fundamentally about facilitating technical communication between project stakeholders
- An opaque architecture has no value as it will not be adequately understood
- Properties of representations:
  - Ambiguity: Open to more than one interpretation?
  - Accuracy: Correct within tolerances
  - Precision: Consistent but not necessarily correct



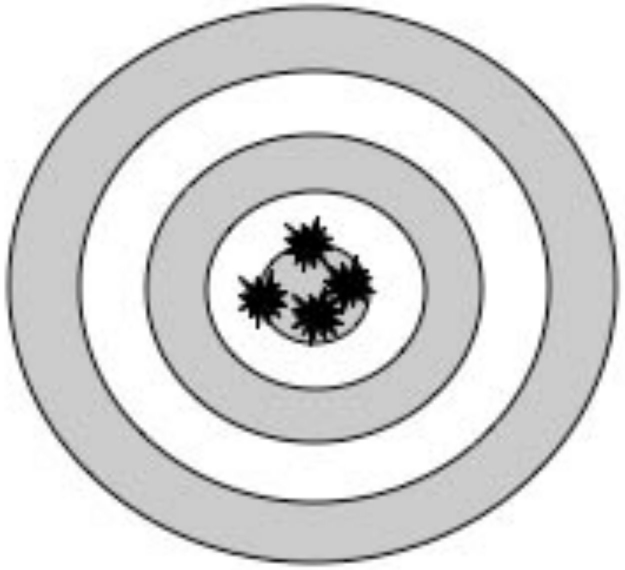
Not Accurate
Not Precise



Accurate
Not Precise



Not Accurate
Precise



Accurate
Precise

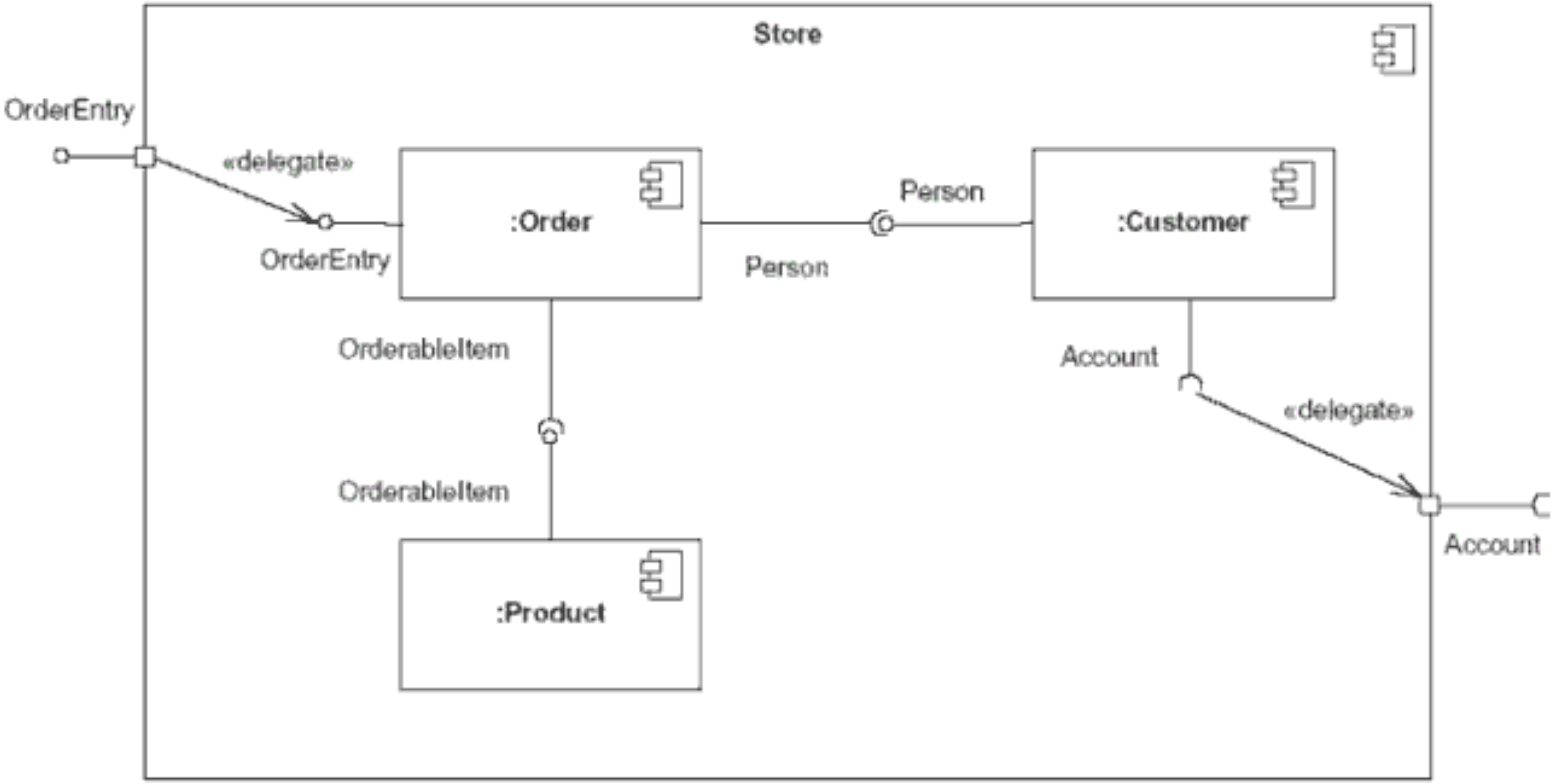
NOAA [http://celebrating200years.noaa.gov/magazine/tct/tct_side1.html]

Architectural views

- Architectural models can be overwhelming
  - Different views focus on specific subsets of elements or subsets of relationships
  - Views often focus on specific concerns or scenarios within a system
- Views overlap; maintaining consistency between views is challenging

Component diagram

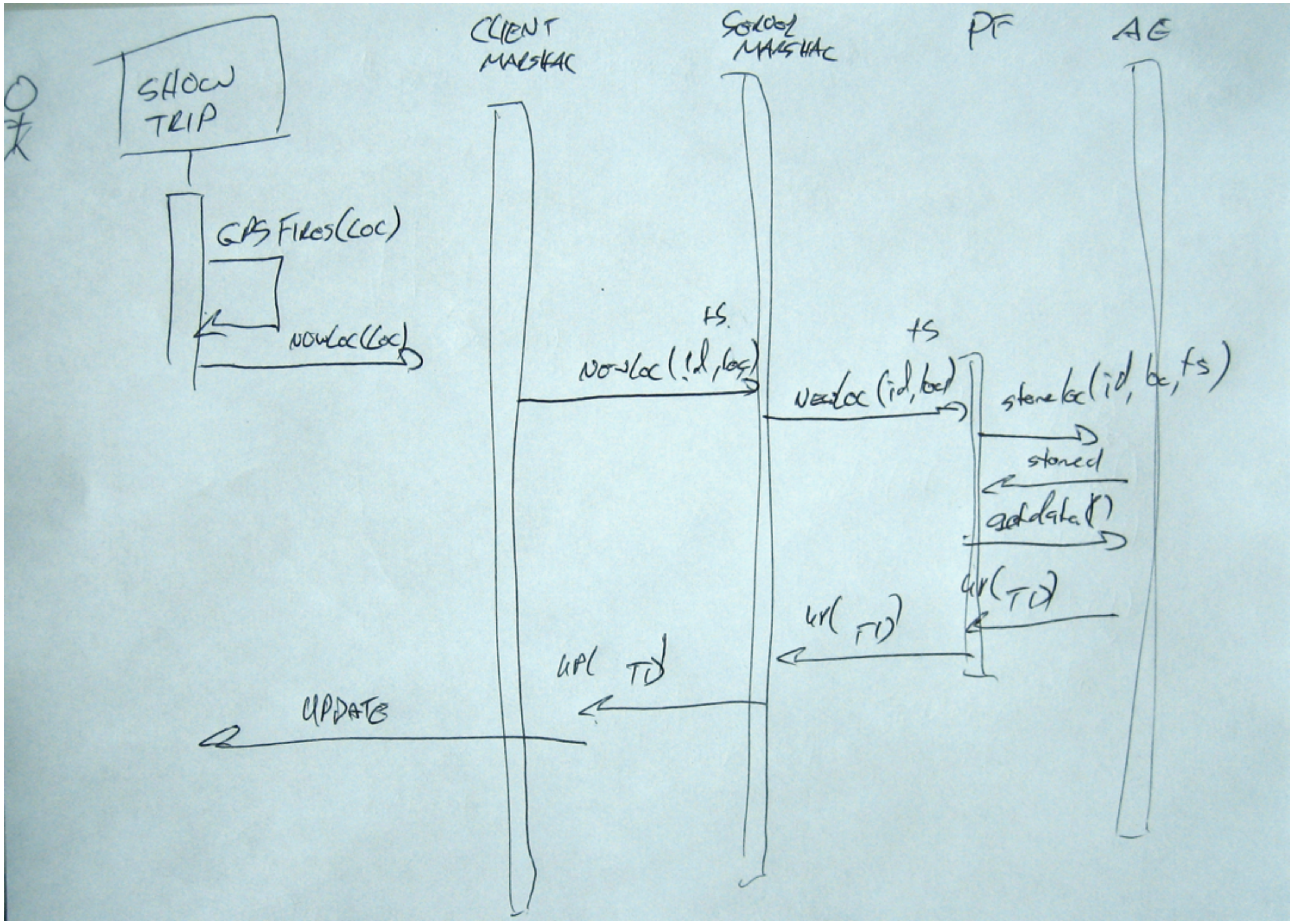
- Captures components and relationships.
  - Required and provided APIs explicitly recorded.



IBM [http://www.ibm.com/developerworks/rational/library/dec04/bell/]

Sequence diagram

- Focus on inter-component collaboration.
- Capture behaviour for specific runtime scenarios.

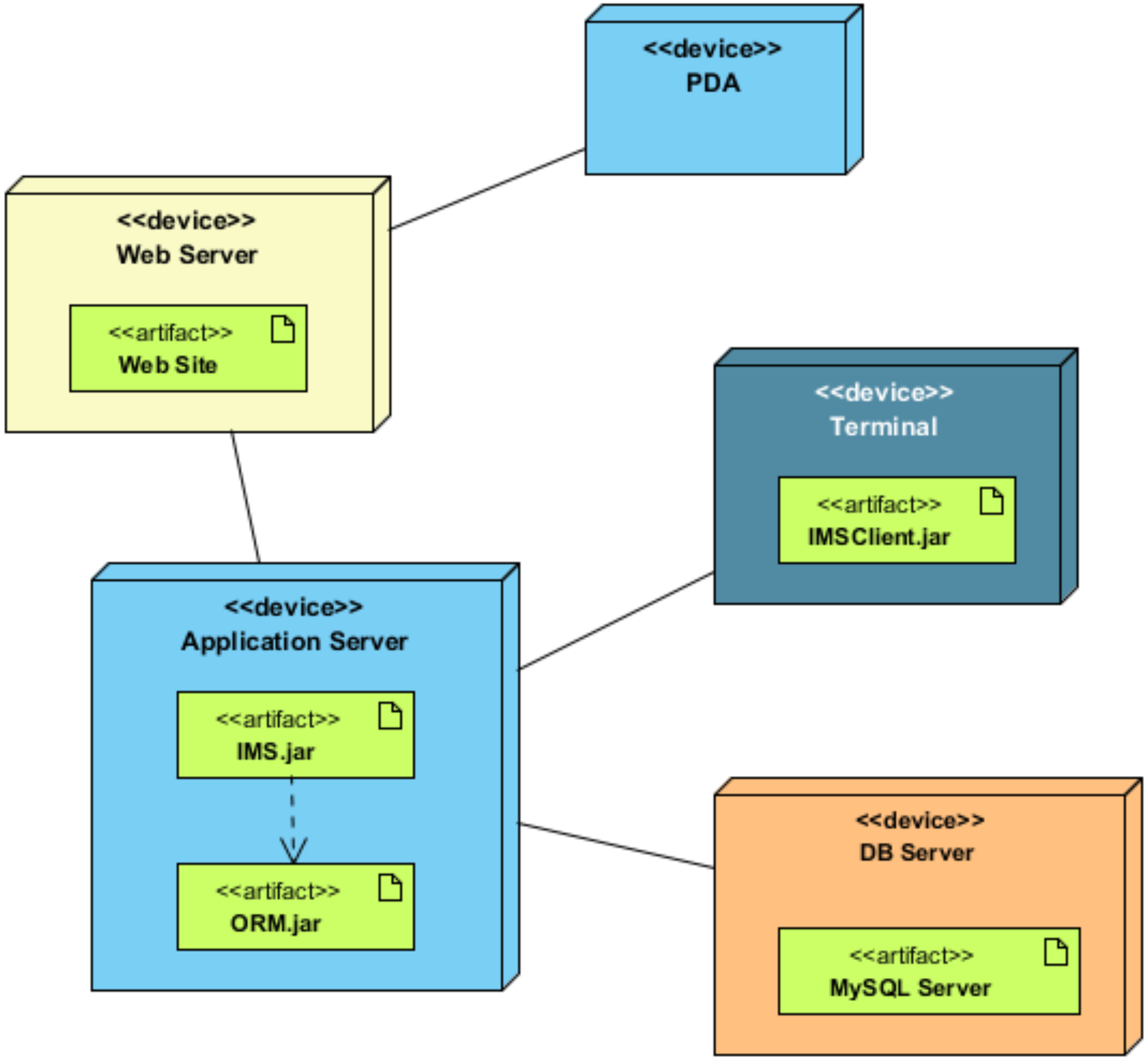


Activity

- Sequence diagram for one use case of the garage door opener.

Deployment diagram

- Provide mapping between physical devices



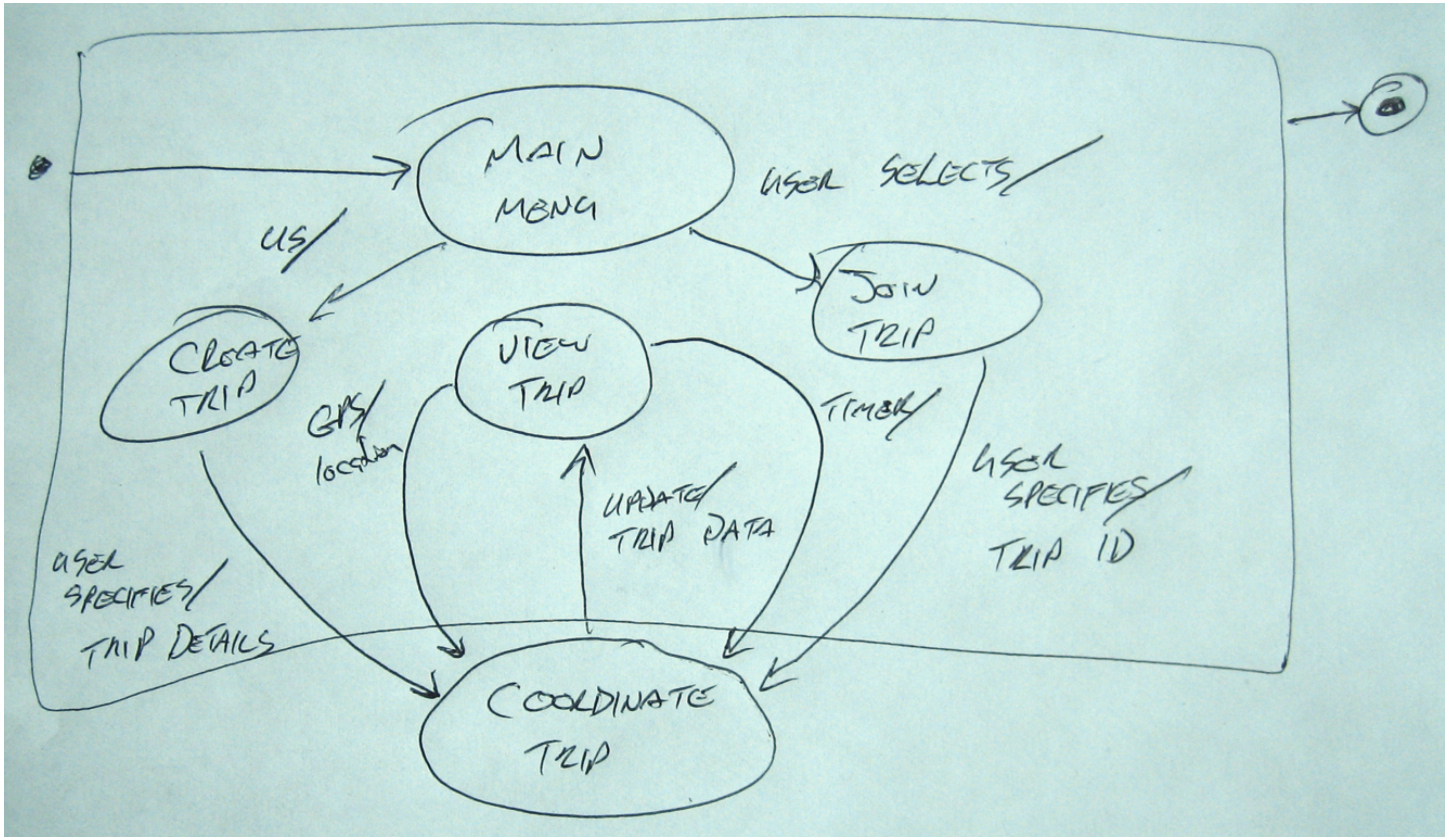
VP [http://www.visual-paradigm.com/VPGallery/diagrams/Deployment.html]

Activity

- Deployment diagram for the garage door opener example from last class.

Statechart diagram

- More formal description of system behaviour.
- Poor mapping between states and components.



Prescriptive vs descriptive

- Prescriptive architecture dictates how the system will be built *a priori*.
  - (as-conceived)
- Descriptive architecture captures how the system was actually built after the fact.
  - (as-implemented)

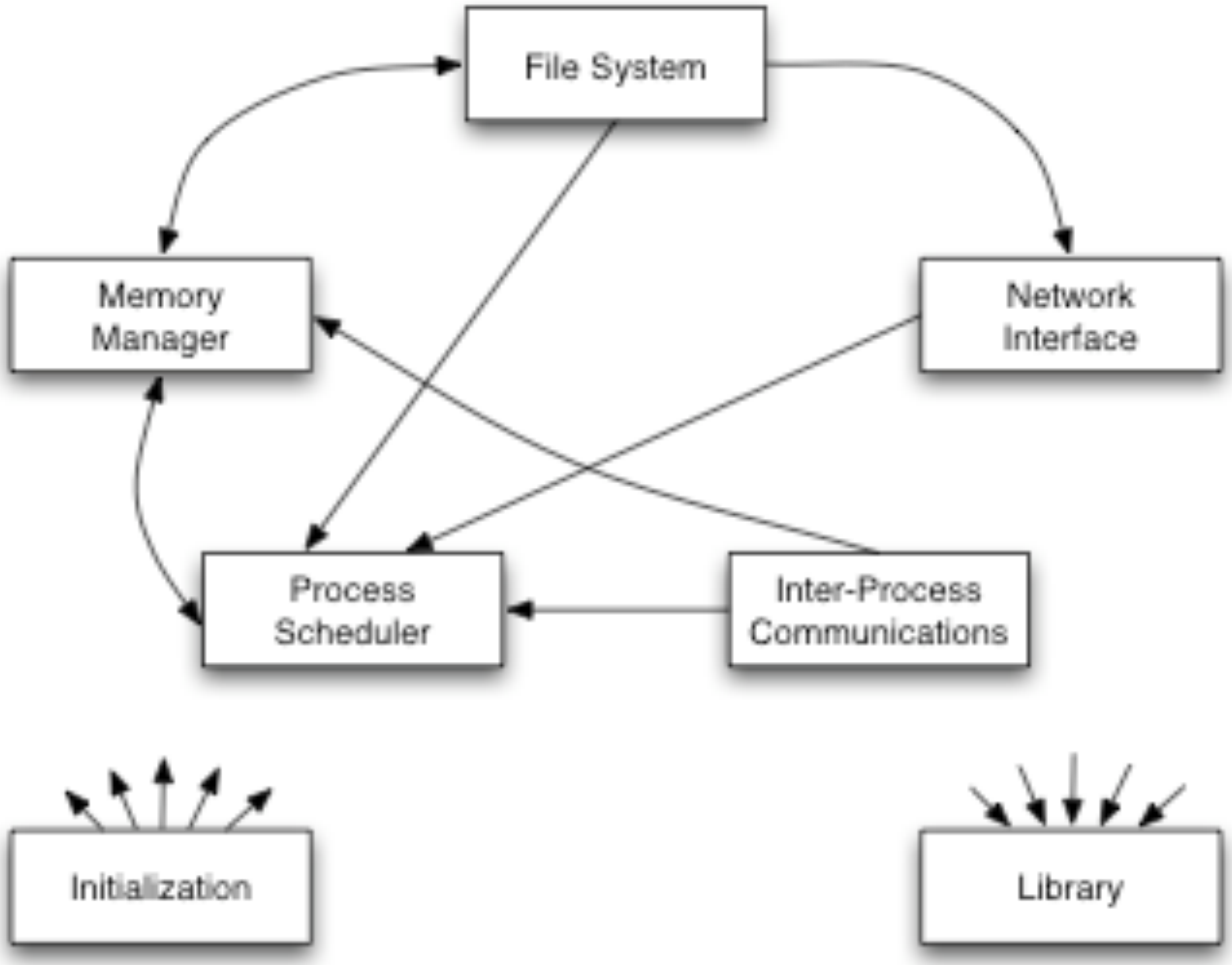
Architectural degradation

- Drift
  - Introduction of changes that are not captured in the current architecture but do not violate it.
- Erosion
  - Introduction of changes that violate the current architecture.

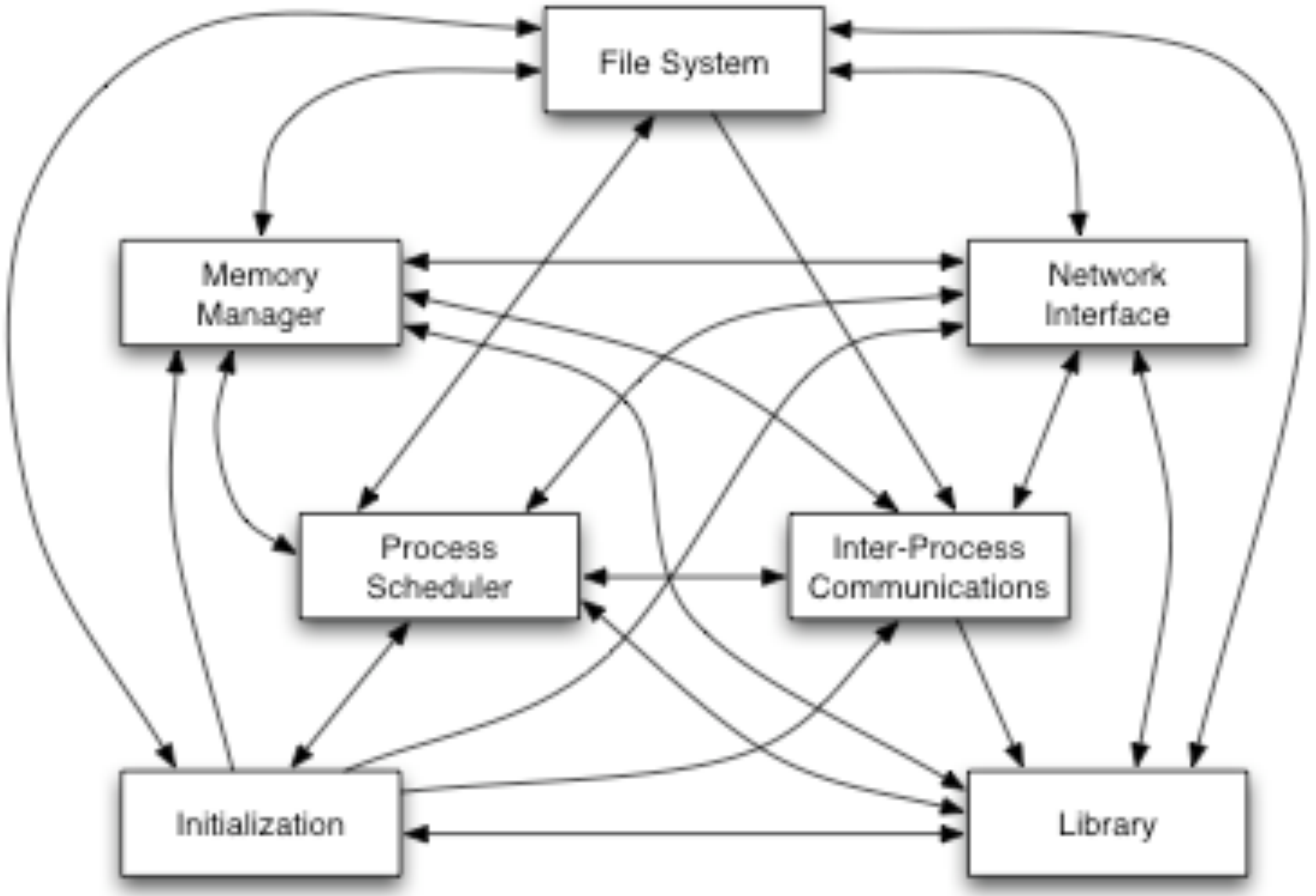
Architectural recovery

- [ICSE 1999: Bowman, Holt, and Brewster]
- Conceptual architecture
  - How developers think about the system.
  - Focuses on meaningful relationships.
- Concrete architecture
  - How the system was actually built.
  - Necessary: the devil is in the details.

Conceptual Architecture



Concrete Architecture


File System
Memory Manager
Network Interface
Process Scheduler
Inter-Process Communications
Initialization
Library