



Software Design and Architecture

Decomposition

Agenda

- what is decomposition
- how – principles
 - coupling and cohesion
 - information hiding
 - Conway's law

Revisit: What is Software Architecture?

- “Architecture is the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution”

-- ANSI/IEEE 1471-2000

- Three primary dimensions:

- Structure (components, subsystems, modules)
- Communication (relationship -> data flow, control flow, dependency, etc.)
- Non-functional requirements

principle design decisions

Architectural Entity Terminologies

- Subsystem

- larger “component” in the IEEE definition
- explicit interface
-  project, module, set of packages

- Component

- smaller “component” that compose a subsystem
-  package, set of classes

- Connector

- interaction mechanisms between subsystems
-  method call, RPC (remote procedure call), network connection, etc.

- Configuration (Topology)

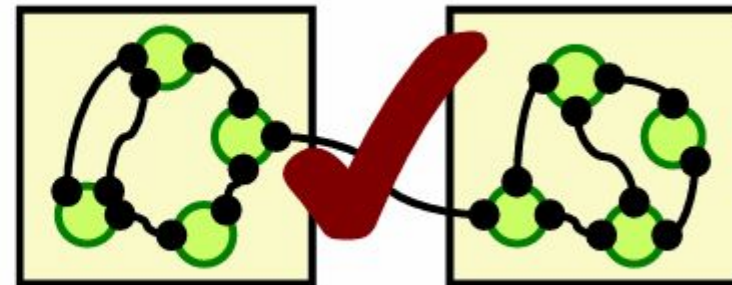
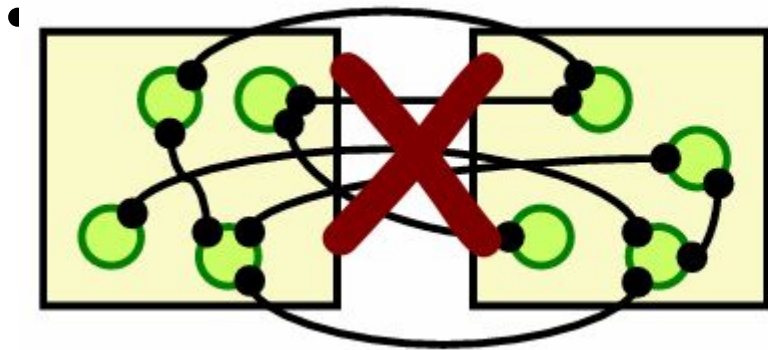
- a set of specific associations between the subsystems and connectors

Decomposition

- The top-down abstraction process of...
 - break down problem into subsystems -> components
 - decide their connectors and configuration
- with the goals of...
 - manage complexity (reason about parts, not the whole)
 - encapsulate domain knowledge about obvious partitions
 - parallel development and clear ownership
 - independent/localized evolution of parts
 - etc.

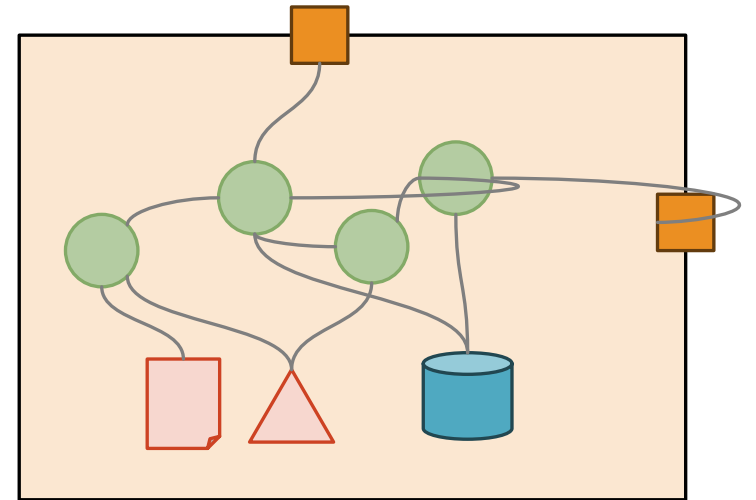
Coupling and Cohesion

- Minimize **coupling** between subsystems
 - the less that subsystems know about each other, the better
 - make future change easier (maintainability)
- Maximize **cohesion** within each subsystem
 - one subsystem should be responsible for one logical service



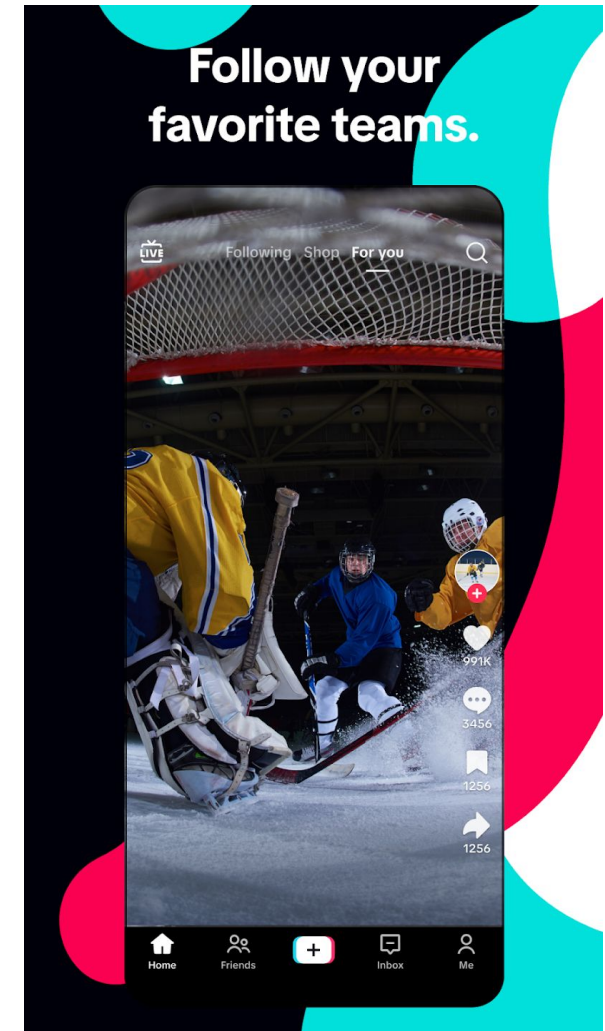
Information Hiding

- Subsystem: encapsulate a set of functionalities, hiding secrets about implementations
 - components, data representations, algorithms, databases...
- **Interface** connects to the outside under contracts
- Users of interfaces should not depend on the secrets



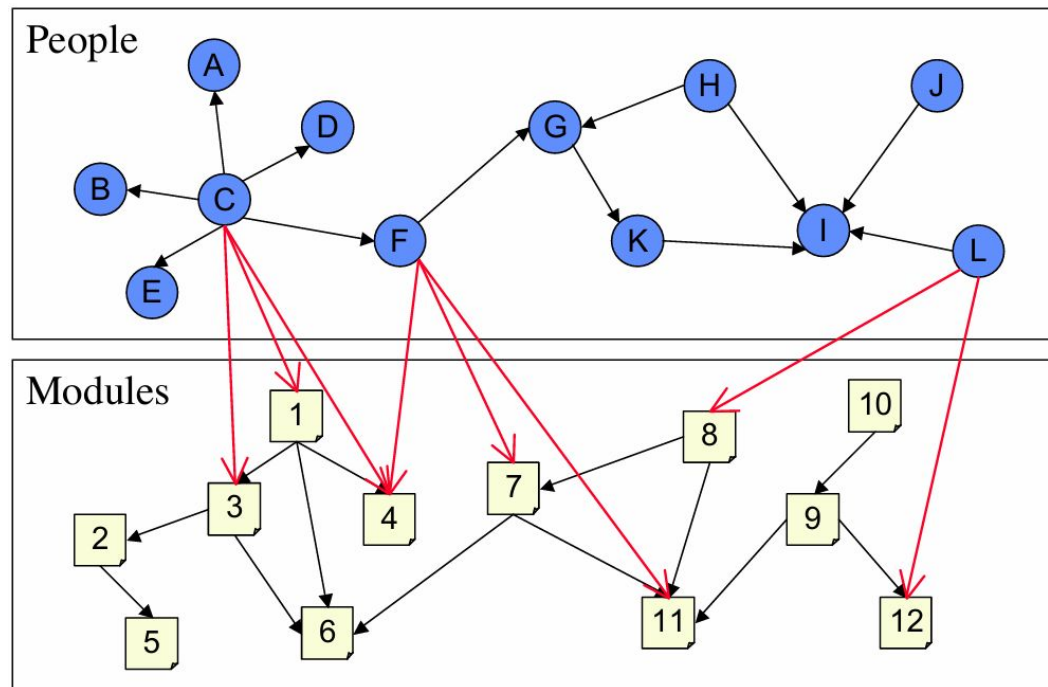
Practice: Decomposition driven by Domain

- Short video app
 - content creators: create and upload short videos
 - content consumers: home feed (recommendation), favorite, bookmark
 - user following, commenting, messaging, etc.
- How should we decompose the app (including its backend)?
 - think about subsystems and their interfaces
 - sketch a component diagram
 - adjust the boundary to minimize coupling + maximize cohesion
 - follow the information hiding principle



Conway's Law

- The software architecture often mirrors the team's communication structure
- Align team ownership with subsystem boundaries to reduce coordination cost



“

The concept of Amazon's two-pizza teams is straightforward: no team should be big enough that it would take more than two pizzas to feed them.

”

- decrease communication overhead
- increase ownership and empowerment
- increase employee satisfaction

Two-Pizza Team @ Amazon

Recap

- Decomposition: top-down abstraction into subsystems/components
- Principles
 - minimize coupling and maximize cohesion
 - information hiding
 - Conway's law (team organization)
- And more tradeoffs to consider...
 - overhead of integration and coordination
 - security, tracability (logging), complexity, etc.