Logical Components

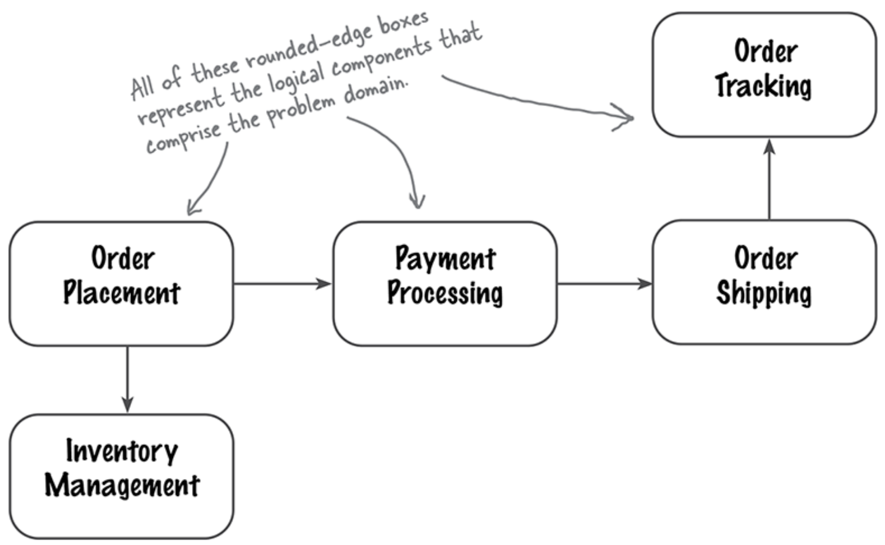
COMP2511, CSE, UNSW

These lecture slides are from the book “*Head First Software Architecture*”,

by Raju Gandhi, Mark Richards, Neal Ford, O'Reilly Media, Inc., March 2024

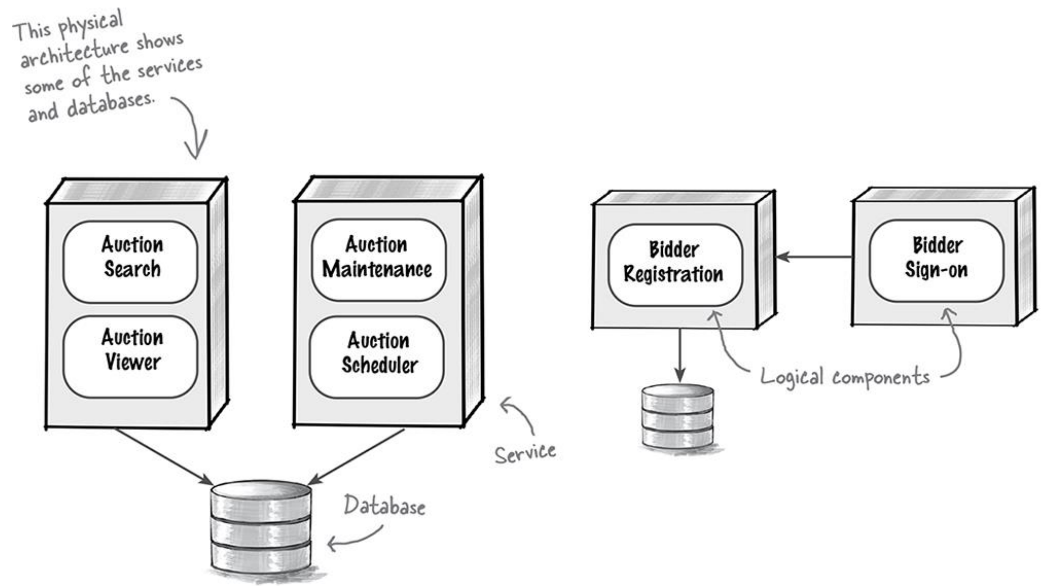
What Are Logical Components?

- Functional building blocks of the system
- Represent major features or responsibilities
- Typically map to folders or modules in the codebase


All of these rounded-edge boxes represent the logical components that comprise the problem domain.
Order Tracking
Order Placement
Payment Processing
Order Shipping
Inventory Management

Logical vs Physical Architecture

- Logical Architecture: Describes what the system does (functional perspective)
- Physical Architecture: Describes how the system is built and deployed (technical perspective)
- Example:
  - Logical: Bidding, Registration, Payment
  - Physical: APIs, databases, gateways, services



Creating a Logical Architecture

Follow a 4-step process:

- Identify core components
- Assign requirements
- Analyse roles & responsibilities
- Align with architectural characteristics

- Revisit this cycle whenever system changes are introduced

Align with Architectural Characteristics

- Break down or merge components based on:
  - Scalability
  - Availability
  - Performance
- Example: Move bid logging to separate Bid Tracker to improve speed and availability

Component Coupling

- Afferent (incoming): How many depend on this component
- Efferent (outgoing): How many this component depends on
- Total Coupling = Afferent + Efferent

Goal: Keep coupling low for flexibility and maintainability

The Law of Demeter

- Also known as the Principle of Least Knowledge
- Each component should only interact with its immediate neighbors
- Avoid tight coupling caused by too much knowledge about the system

Coupling Trade-offs

- Tightly Coupled System: Easier to trace workflow, harder to change
- Loosely Coupled System: More maintainable, but harder to understand in one place

Remember: Everything is a trade-off

Summary

- Logical components are your system’s functional map
- Use descriptive names based on responsibilities
- Avoid entity trap and generic components
- Reduce coupling using the Law of Demeter
- Regularly reevaluate components as requirements evolve