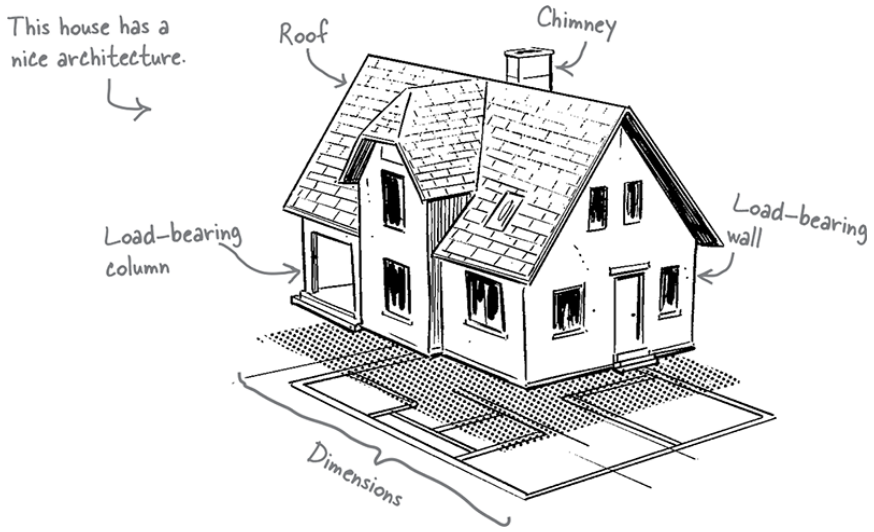
# Software Architecture


UNSW SYDNEY

These lecture slides are from the book "*Head First Software Architecture*",
by Raju Gandhi, Mark Richards, Neal Ford, O'Reilly Media, Inc., March 2024

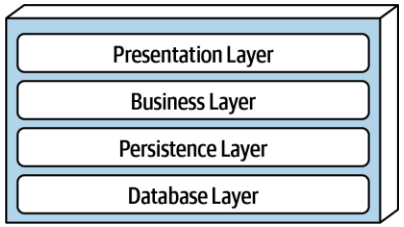
# Software Architecture as a Metaphor

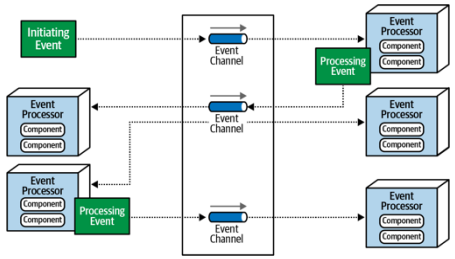
- While building a house, architectural decisions (rooms, floors, layout) are crucial and costly to change later.
- A poorly architectural house can lead to substandard and uncomfortable living conditions.

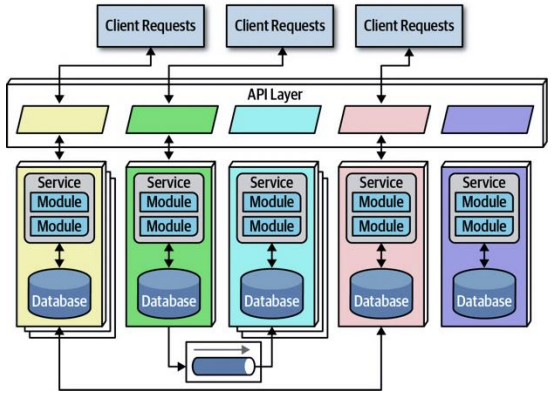


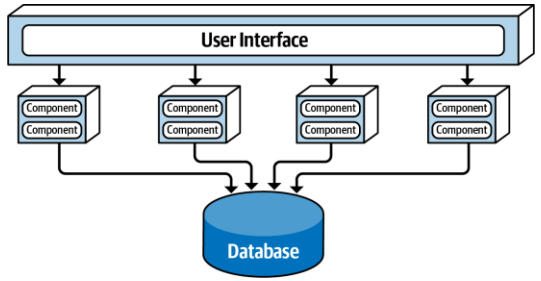

# What is Software Architecture?

- Software architecture defines the fundamental structure of a software system.
- Influences how effectively the software can adapt to changes, scale, perform, and maintain its reliability.
- *Software Architecture* diagrams represent relationships between components (e.g. databases, services, interfaces).



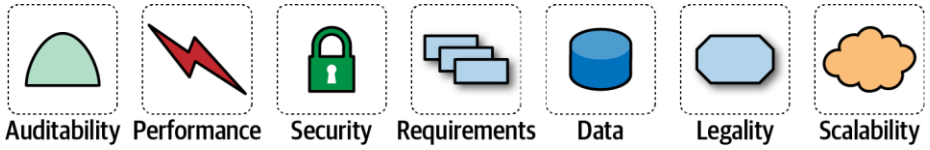
# The Four Dimensions of Software Architecture

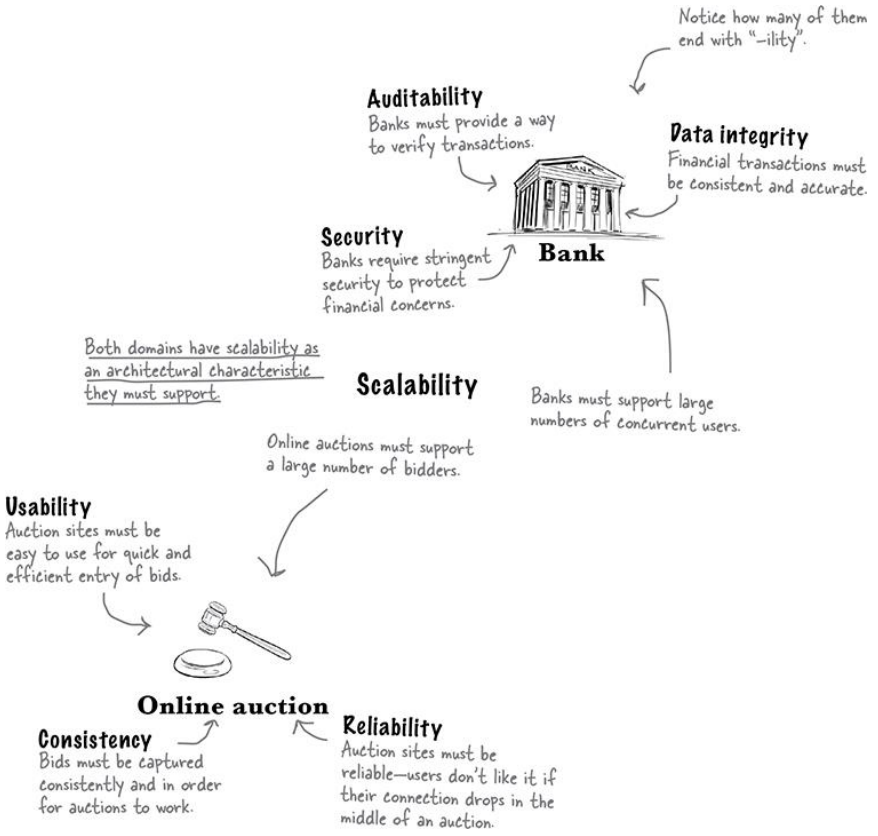
1. Architectural Characteristics
2. Architectural Decisions
3. Logical Components
4. Architectural Style

# Dimension 1: Architectural Characteristics

- Architectural Characteristics define fundamental qualities software architecture must support.

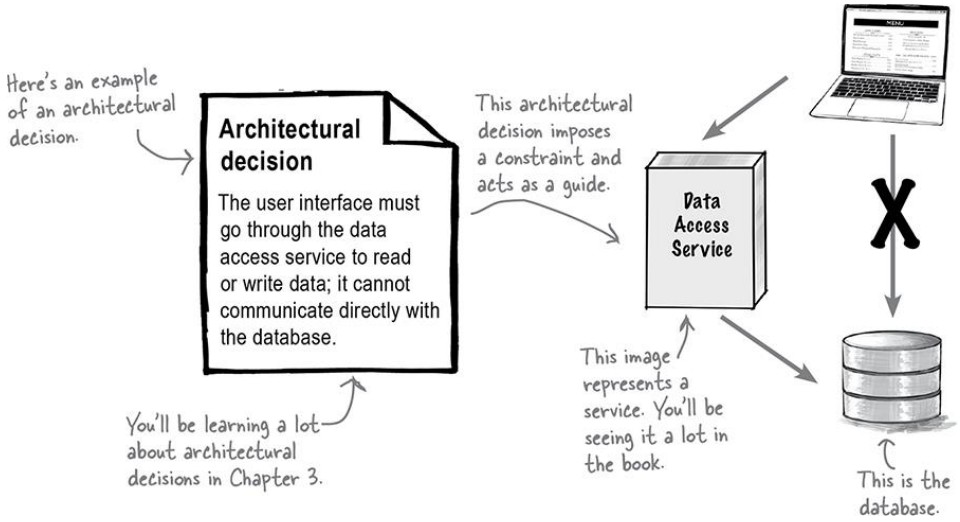
- Commonly used Architectural Characteristics:
  - Scalability (support growth)
  - Reliability (consistent operation)
  - Availability (system uptime)
  - Testability (ease of testing components)
  - Security



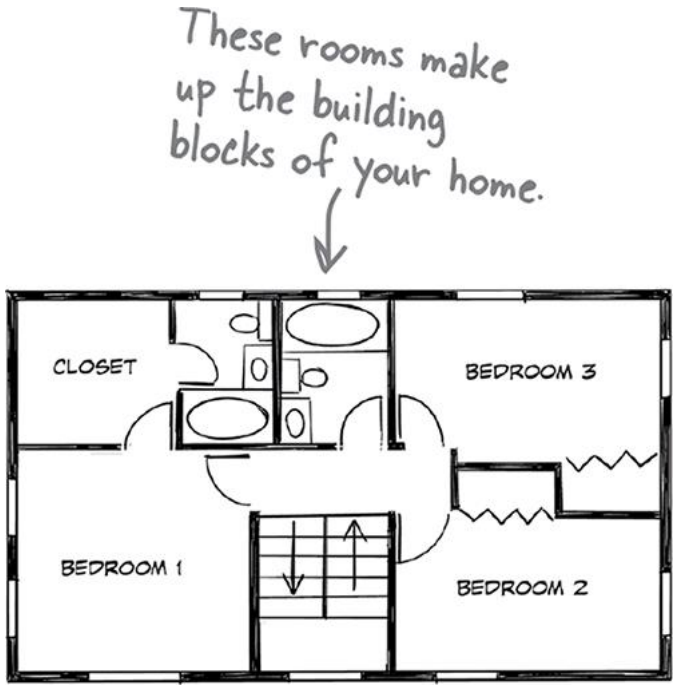
# Dimension 2: Architectural Decisions

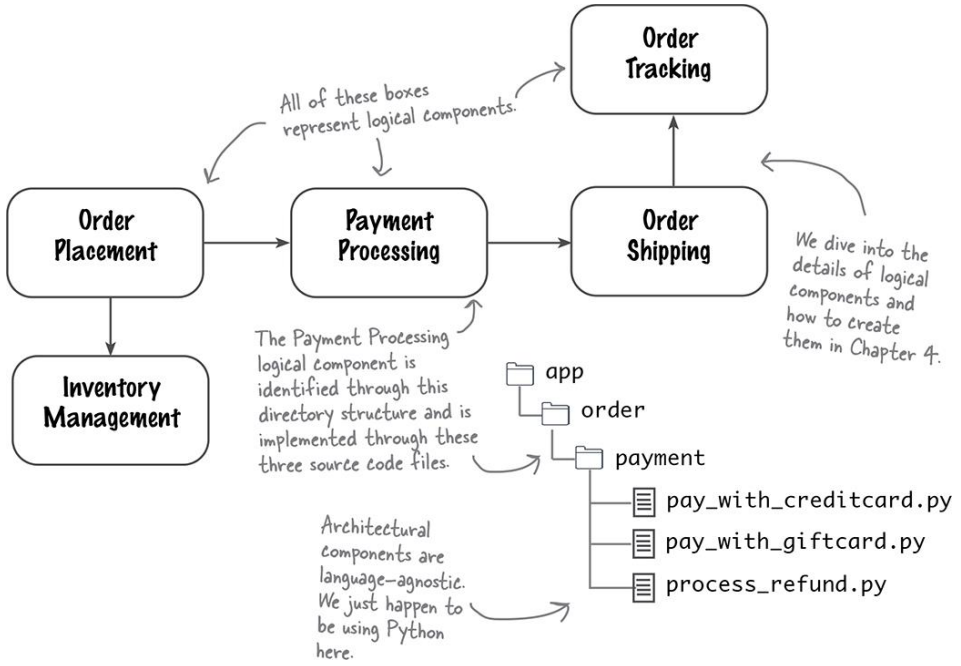
- Long-term structural decisions influencing software behaviour.
- Architectural Decisions set constraints guiding future development.

# Dimension 3: Logical Components

- Functional building blocks representing business features.

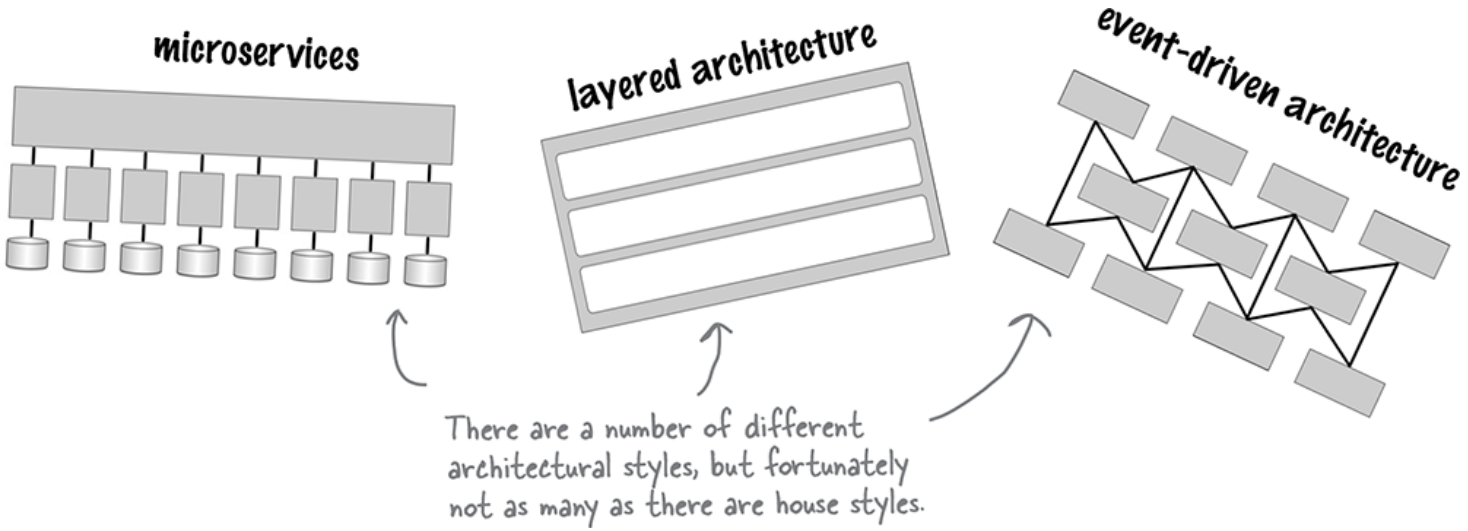


# Dimension 4: Architectural Styles

- Overall system shape and structural patterns.
- Common styles:
  - Layered (clear separation of concerns)
  - Microservices (highly scalable and agile)
  - Event-driven (responsive and scalable)

- Real-world Examples:
  - Netflix adopting microservices.
  - Traditional enterprise apps using layered architecture.



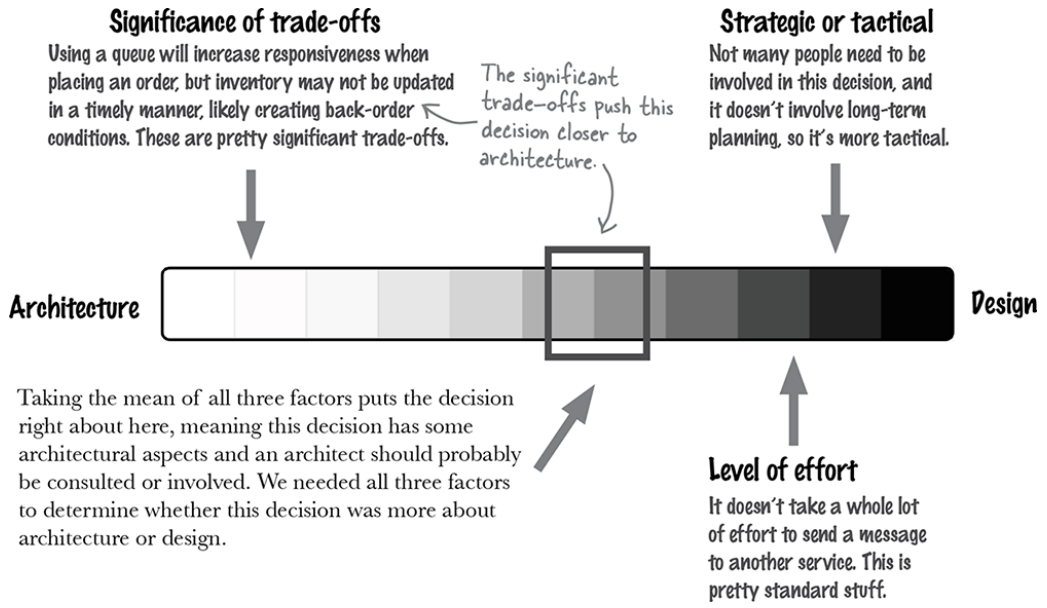
# Architecture vs. Design

- **Architecture**: Structural decisions (hard to change).
- **Design**: Appearance and detailed decisions (easy to change).

- Decisions exist on a spectrum from pure architecture to pure design.
- Strategic decisions (architecture): Long-term, high impact, high effort.
- Tactical decisions (design): Short-term, low impact, low effort.

Example:

- Choosing databases (architecture) vs. UI button colour (design).

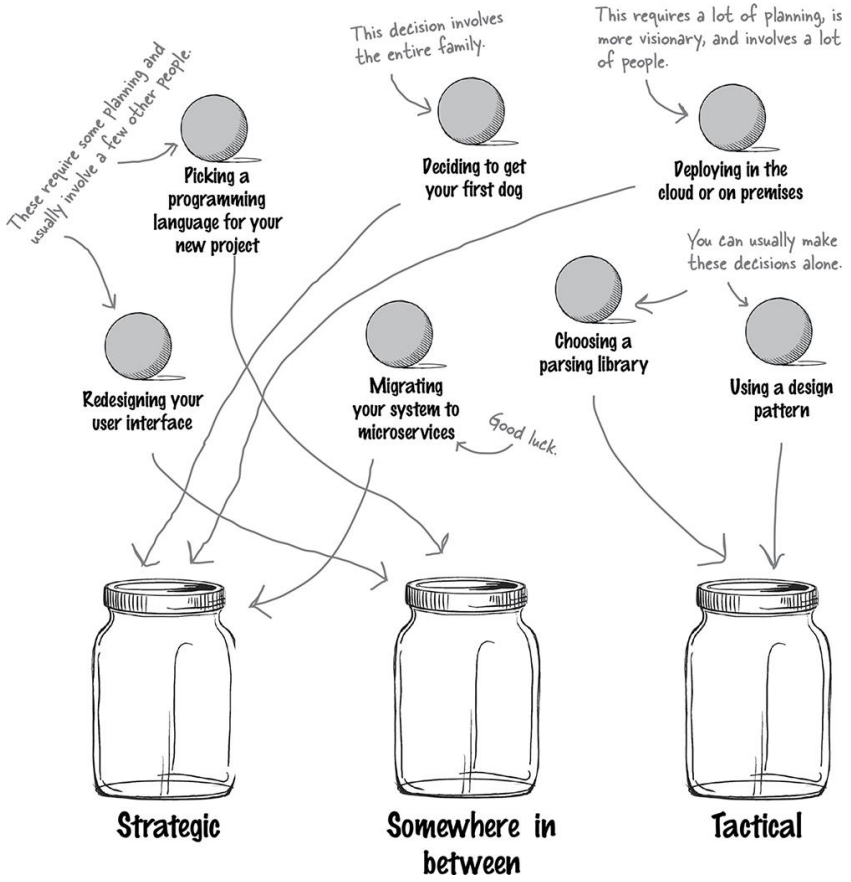


# Identifying Architectural Decisions

- Questions to consider:
  - Is it strategic (long-term) or tactical (short-term)?
  - Effort to change: high or low?
  - Does it involve significant trade-offs?

Examples:

- Migrating from monolith to microservices (architecture, strategic).
- Changing background colour of login page (design, tactical).



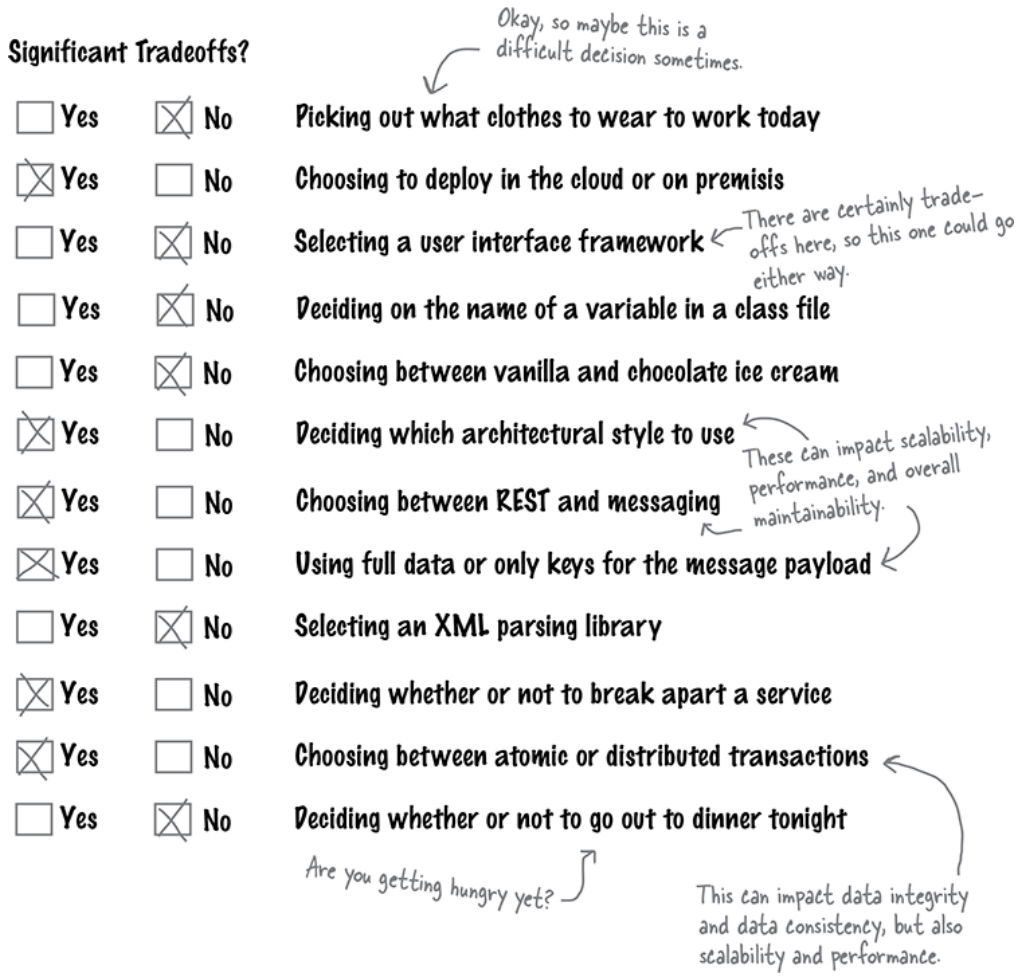
# Trade-offs in Decision Making

- Architectural decisions often involve significant trade-offs.

Example:

  - Cloud deployment: scalability vs. cost.
  - Async messaging: performance vs. complexity.
  - Choosing between performance and data consistency.

- Architects handle strategic choices; developers manage detailed tactical choices



# Summary (1)

- Architecture focuses on structure and system-wide qualities; design is more about code-level appearance and organization.

- Four essential dimensions of software architecture:
  - Architectural Characteristics – Foundation traits like scalability, availability, security.
  - Architectural Decisions – Guideposts that define the system's constraints and trade-offs.
  - Logical Components – Functional building blocks implemented in code.
  - Architectural Style – High-level patterns like layered, event-driven, or microservices.

# Summary (2)

- Software architecture is about making informed structural decisions, not just organising code.
- Understand and prioritise architectural characteristics for your system.
- Every architectural decision involves trade-offs, know the “why.”
- Use ADRs to document decisions and ensure long-term clarity.
- Choose an architectural style that supports your system’s most critical characteristics.
- Know when a decision is architectural (system-wide impact) or design-level (local impact).

“Good architecture supports change. Great architecture explains why.”