Software Architecture

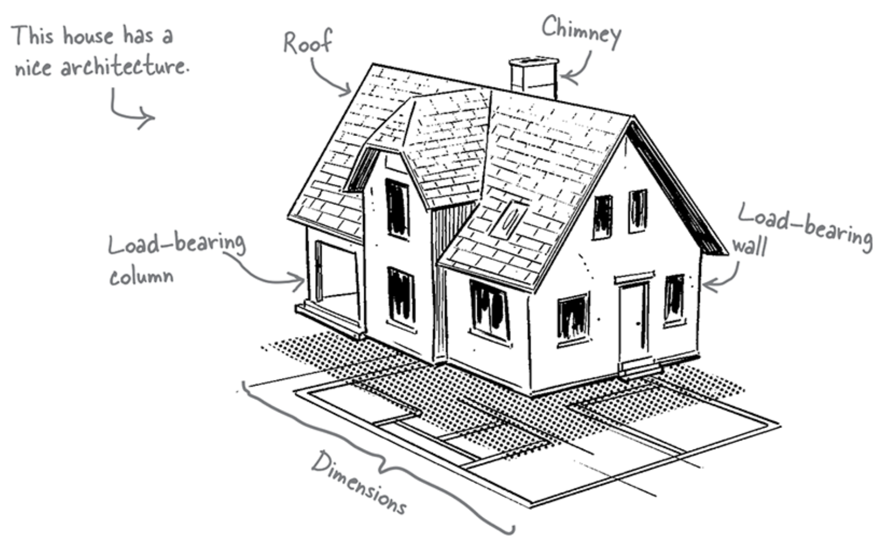
COMP2511, CSE, UNSW

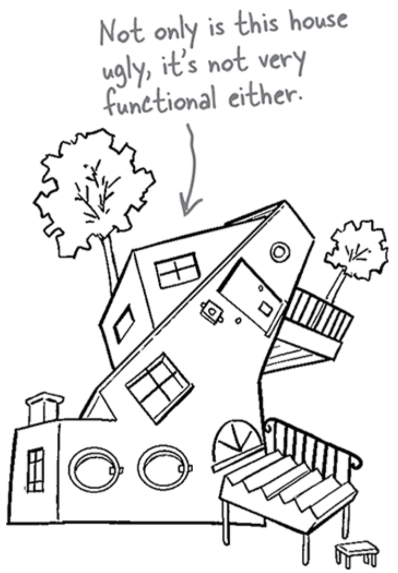
These lecture slides are from the book “*Head First Software Architecture*”,

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

Software Architecture as a Metaphor

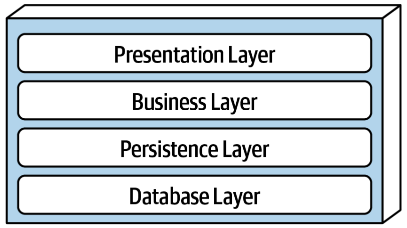
- While building a house, architectural decisions (rooms, floors, layout) are crucial and costly to change later.
- A poorly architectural house can lead to substandard and uncomfortable living conditions.

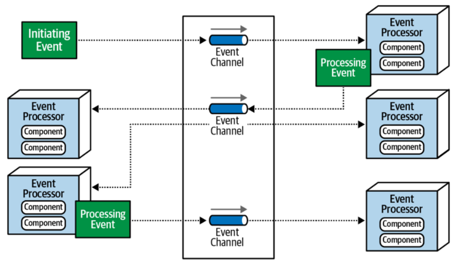


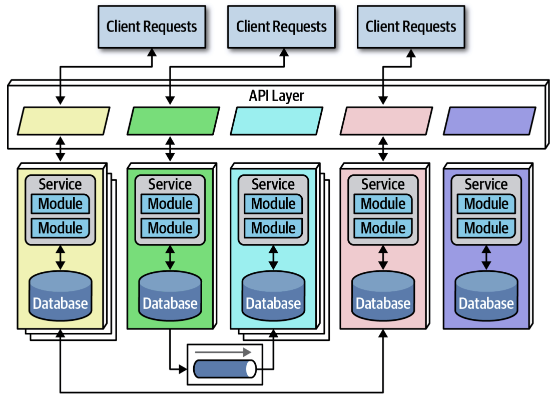

Not only is this house ugly, it's not very functional either.

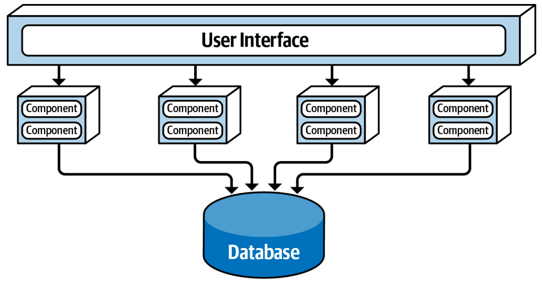
What is Software Architecture?

- ❖ 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

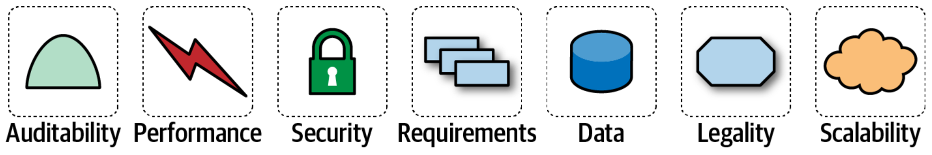
1. Architectural Characteristics
2. Architectural Decisions
3. Logical Components
4. Architectural Style

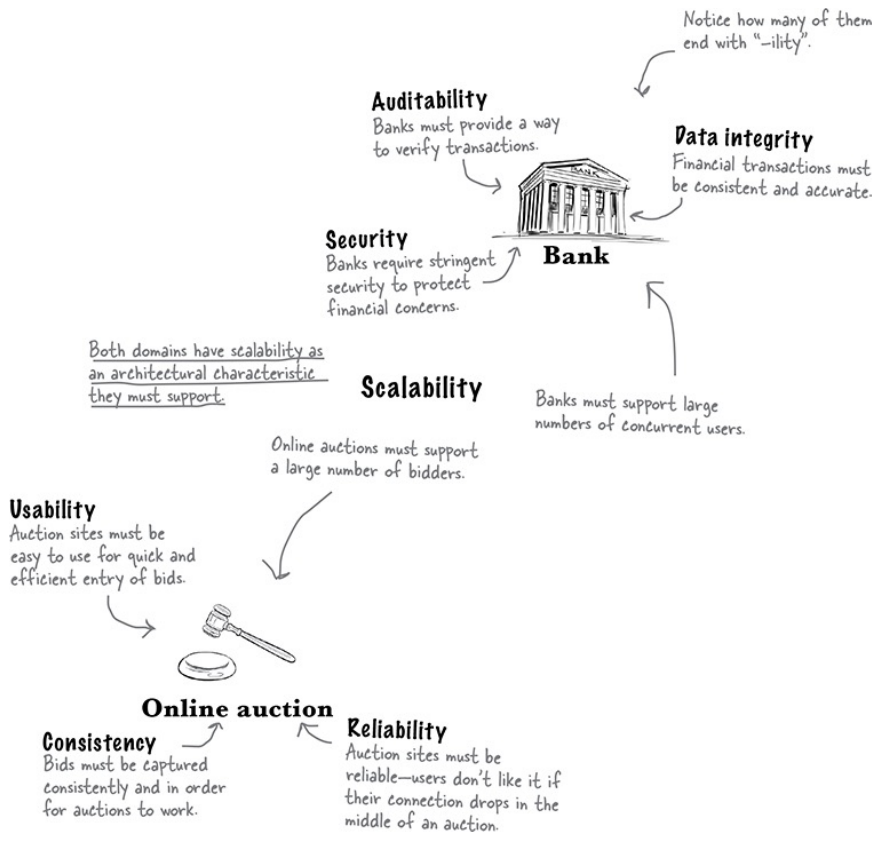
Dimension 1: Architectural Characteristics

❖ 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

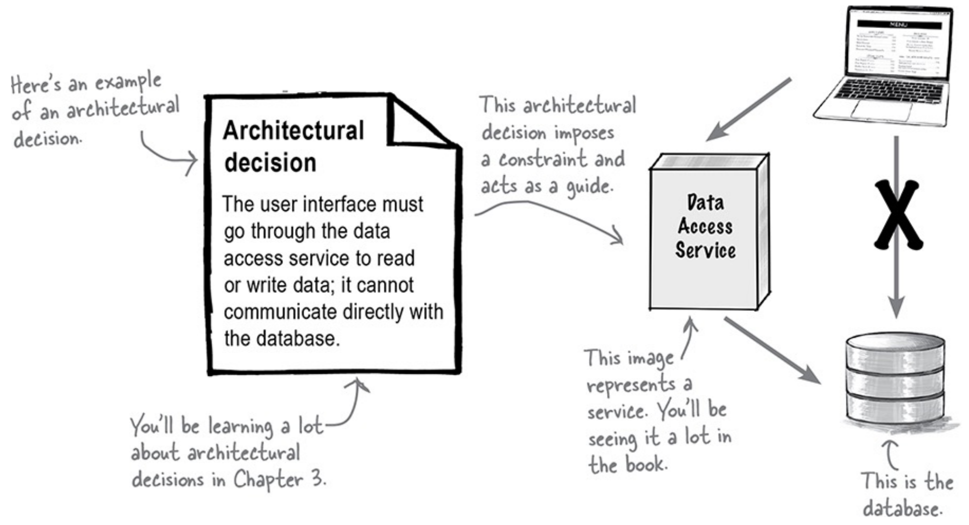



Notice how many of them end with "–ility".
Auditability
Banks must provide a way to verify transactions.
Data integrity
Financial transactions must be consistent and accurate.
Security
Banks require stringent security to protect financial concerns.
Bank
Both domains have scalability as an architectural characteristic they must support.
Scalability
Banks must support large numbers of concurrent users.
Online auctions must support a large number of bidders.
Usability
Auction sites must be easy to use for quick and efficient entry of bids.
Online auction
Consistency
Bids must be captured consistently and in order for auctions to work.
Reliability
Auction sites must be reliable—users don't like it if their connection drops in the middle of an auction.

Dimension 2: Architectural Decisions

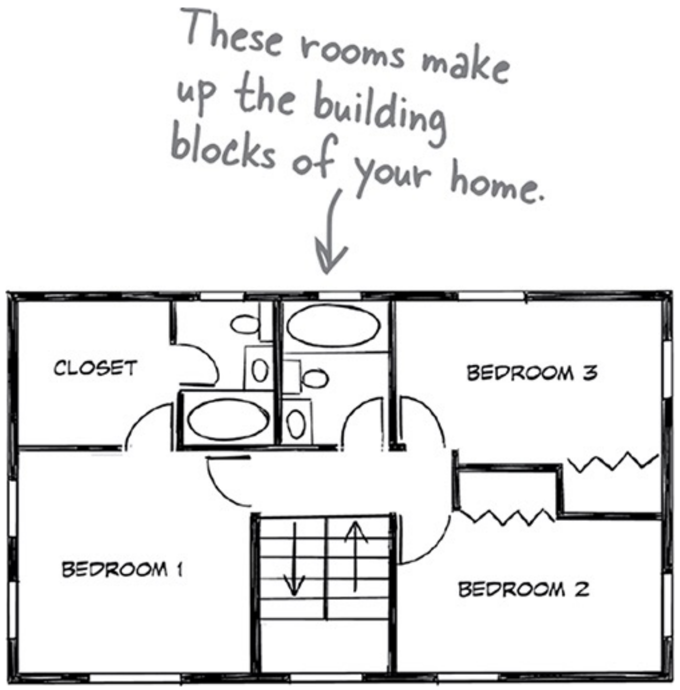
- ❖ Long-term structural decisions influencing software behaviour.
- ❖ Architectural Decisions set constraints guiding future development.

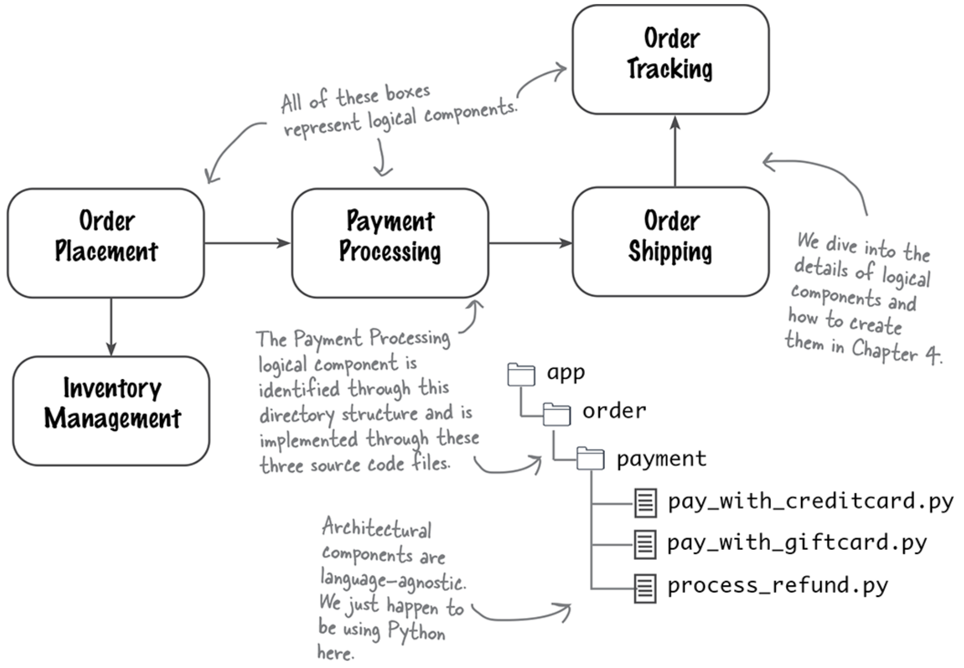
What should your home look like? This kind of decision is an architectural one.
?
?


Here's an example of an architectural decision.
Architectural decision
The user interface must go through the data access service to read or write data; it cannot communicate directly with the database.
This architectural decision imposes a constraint and acts as a guide.
Data Access Service
X
This image represents a service. You'll be seeing it a lot in the book.
This is the database.
You'll be learning a lot about architectural decisions in Chapter 3.

Dimension 3: Logical Components

- Functional building blocks representing business features.



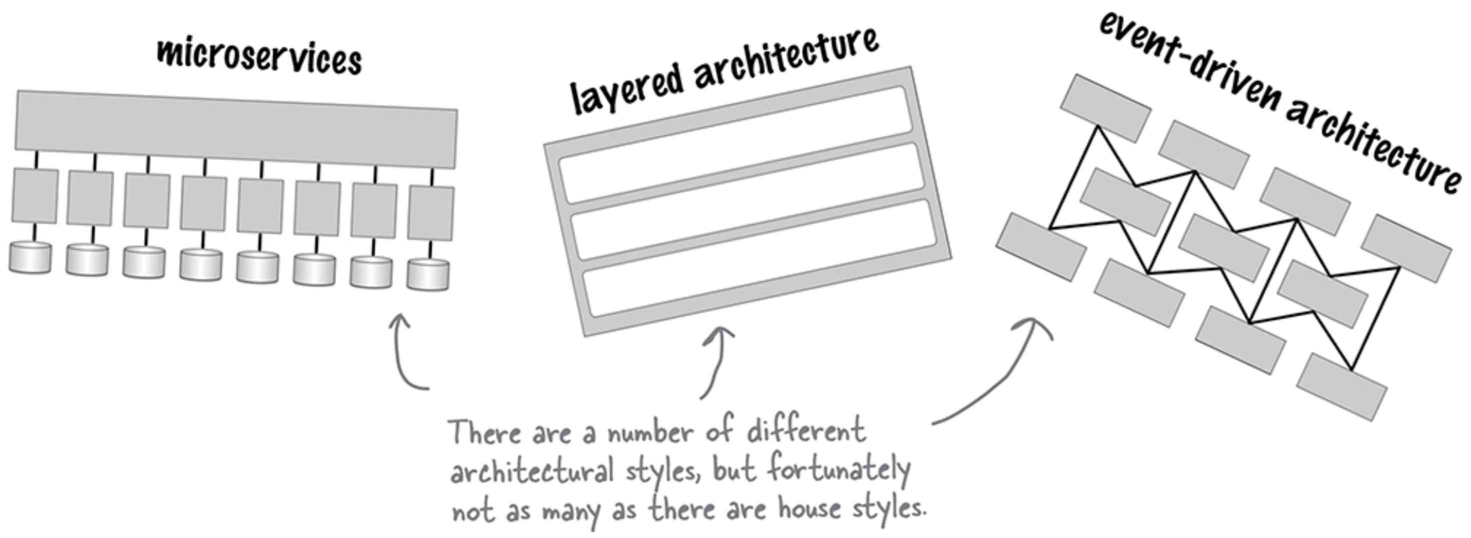


Order Tracking
All of these boxes represent logical components.
Order Placement
Payment Processing
Order Shipping
Inventory Management
We dive into the details of logical components and how to create them in Chapter 4.
The Payment Processing logical component is identified through this directory structure and is implemented through these three source code files.
app
order
payment
pay_with_creditcard.py
pay_with_giftcard.py
process_refund.py
Architectural components are language-agnostic. We just happen to be using Python here.

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.


microservices
layered architecture
event-driven architecture
There are a number of different architectural styles, but fortunately not as many as there are house styles.

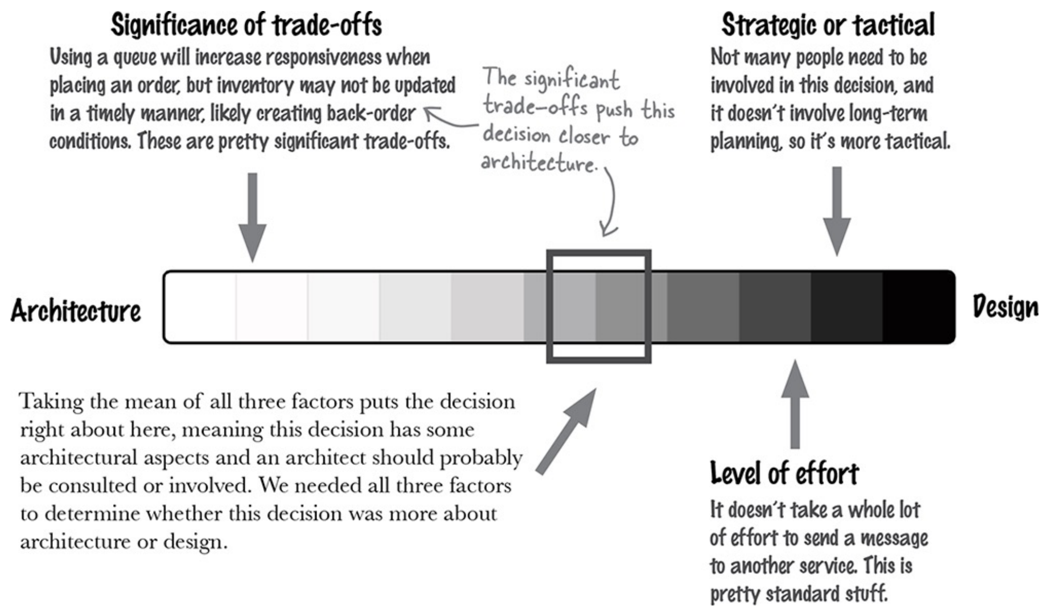
Architecture vs. Design

- **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).

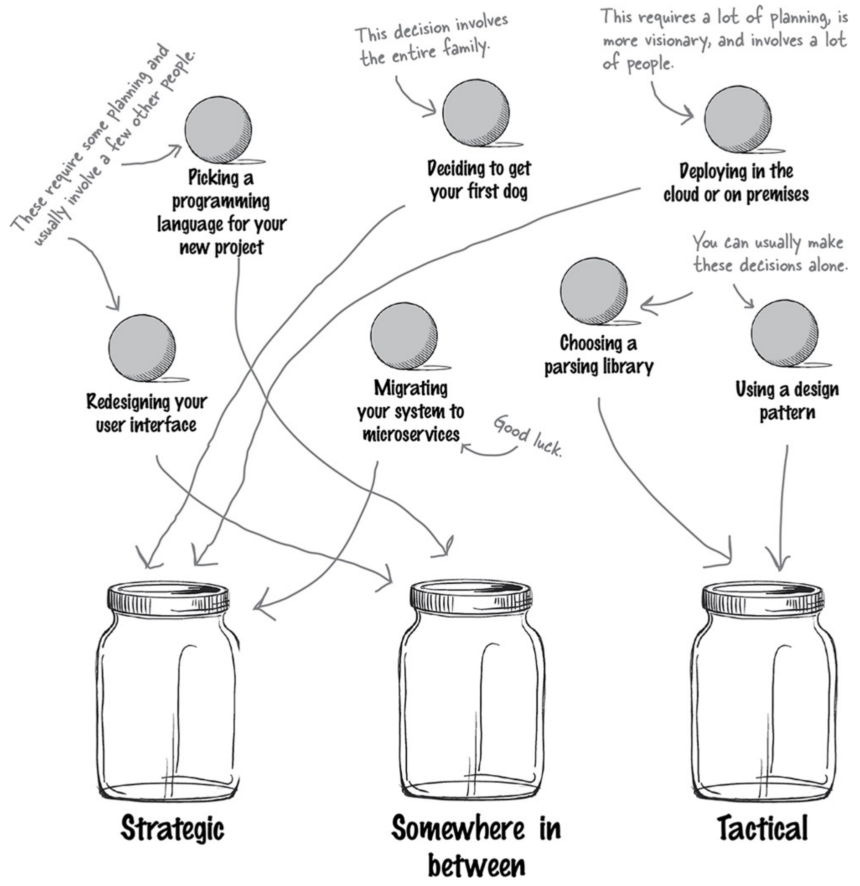

Significance of trade-offs
Using a queue will increase responsiveness when placing an order, but inventory may not be updated in a timely manner, likely creating back-order conditions. These are pretty significant trade-offs.
The significant trade-offs push this decision closer to architecture.
Strategic or tactical
Not many people need to be involved in this decision, and it doesn't involve long-term planning, so it's more tactical.
Architecture
Design
Taking the mean of all three factors puts the decision right about here, meaning this decision has some architectural aspects and an architect should probably be consulted or involved. We needed all three factors to determine whether this decision was more about architecture or design.
Level of effort
It doesn't take a whole lot of effort to send a message to another service. This is pretty standard stuff.

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).


These require some planning and usually involve a few other people.
This decision involves the entire family.
This requires a lot of planning, is more visionary, and involves a lot of people.
Picking a programming language for your new project
Deciding to get your first dog
Deploying in the cloud or on premises
You can usually make these decisions alone.
Redesigning your user interface
Migrating your system to microservices
Good luck.
Choosing a parsing library
Using a design pattern
Strategic
Somewhere in between
Tactical

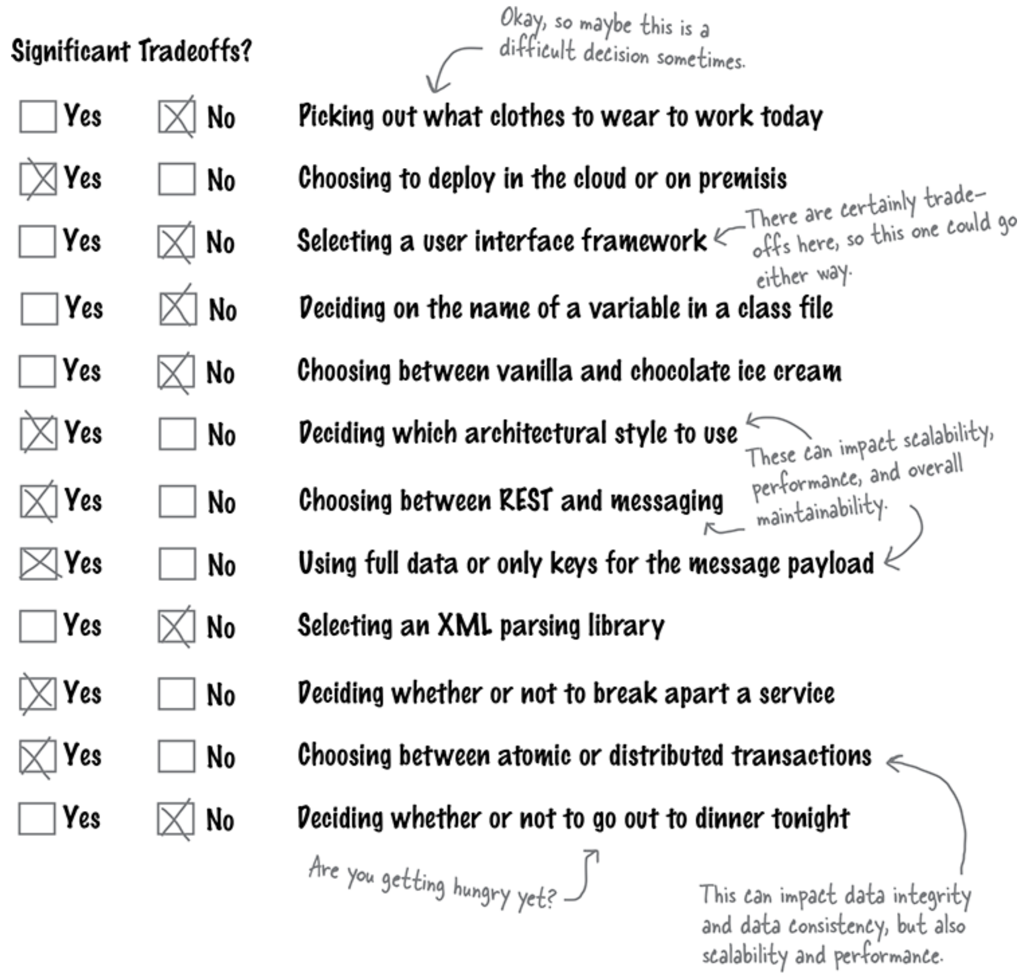
Trade-offs in Decision Making

- 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



Significant Tradeoffs?

Yes	[✓ No]	Picking out what clothes to wear to work today
[✓ Yes]	No	Choosing to deploy in the cloud or on premisis
Yes	[✓ No]	Selecting a user interface framework
Yes	[✓ No]	Deciding on the name of a variable in a class file
Yes	[✓ No]	Choosing between vanilla and chocolate ice cream
[✓ Yes]	No	Deciding which architectural style to use
[✓ Yes]	No	Choosing between REST and messaging
[✓ Yes]	No	Using full data or only keys for the message payload
Yes	[✓ No]	Selecting an XML parsing library
[✓ Yes]	No	Deciding whether or not to break apart a service
[✓ Yes]	No	Choosing between atomic or distributed transactions
Yes	[✓ No]	Deciding whether or not to go out to dinner tonight

Okay, so maybe this is a difficult decision sometimes.

There are certainly trade-offs here, so this one could go either way.

These can impact scalability, performance, and overall maintainability.

This can impact data integrity and data consistency, but also scalability and performance.

Are you getting hungry yet?

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.”