

Architecture Decisions and Modelling

Mojtaba Shahin

Week 7: Lectorial – Part 1

Content

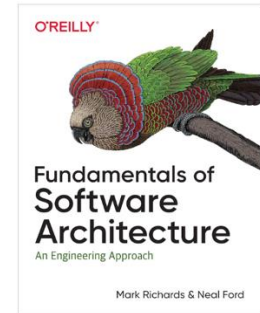
- **Part 1**
 - Architectural Decisions
- **Part 2**
 - Architecture Modelling and Views

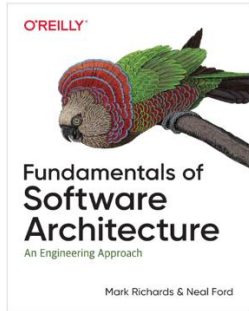
Acknowledgements

- Most of the **texts** and **images** in the slides come from the following sources:
 - Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2020 (First Edition).
 - Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2025 (Second Edition).
 - Pautasso, C., Software Architecture- Visual Lecture Notes, LeanPub, 2023 (<https://leanpub.com/software-architecture/>)
 - Brown, Simon. "The C4 model for visualising software architecture" (c4model.com)
 - <https://docs.structurizr.com/>
 - The University of Queensland's Software Architecture Course Materials, @ Richard Thomas, CC BY-SA 4.0 (<https://github.com/CSSE6400>)
 - Bass, L., Clements, P., Kazman, R., Software Architecture in Practice, Addison-Wesley, 2021.
 - Gandhi, R., Richards, M., Ford, N., Head First Software Architecture, O'Reilly Media, Inc. 2024

Architecture Decisions

- One of the core expectations of an architect is to make architecture decisions
- Steps of making an architecture decision
 - Gathering enough relevant information,
 - Justifying the decision
 - Documenting the decision, and
 - Effectively communicating that decision to the right stakeholders.





Question

What types of decisions made in the development process are considered “architecture decisions”?

Architecturally Significant Decisions

- Architecturally significant decisions (i.e., architecture decisions) are those decisions that affect the **structure, non-functional characteristics, dependencies, interfaces, or construction techniques**.

Example

If an architect makes a decision to use a particular technology because it directly supports a particular architecture characteristic (such as performance or scalability), then it's an architecture decision.

The Influences of Architecture Decisions

(1) Structure

refers to decisions that impact the patterns or styles of architecture being used.

Example: The decision to share data between a set of microservices. This decision impacts the bounded context of the microservice, and as such affects the structure of the application.

(2) Non-functional characteristics (QAs)

are the architecture characteristics (“-ilities”) that are important for the application or system being developed or maintained.

Example: If a choice of technology impacts performance, and performance is an important aspect of the application, then it becomes an architecture decision.

The Influences of Architecture Decisions

(3) Dependencies

refer to coupling points between components and/or services within the system, which in turn impact overall scalability, modularity, agility, testability, reliability, and so on.

(4) Interfaces

refer to how services and components are accessed and orchestrated, usually through a gateway, integration hub, service bus, or API proxy. Interfaces usually involve defining contracts, including the versioning and deprecation strategy of those contracts.

(5) Construction techniques


refer to decisions about platforms, frameworks, tools, and even processes that, although technical in nature, might impact some aspect of the architecture

Question

How do you know **why certain decisions** were made in the architectural design?

Answer

Architectural Decision Records (ADRs)

A purple starburst graphic with multiple points, containing text.

A way to document a software architecture

The Second Law of Software Architecture

Why is more important than ***how***.

Template for ADRs

- An ADR is a document that describes a specific architectural decision. You write one for every architectural decision you make.

Title ...

Status ...

Context ...

Decision ...

Consequences ...

Compliance ...

Notes ...

Template for ADRs

Title

a short phrase describing the architecture decisions

Example of a decision title

ADR 42. Use of Asynchronous Messaging Between Order and Payment Services.

Template for ADRs

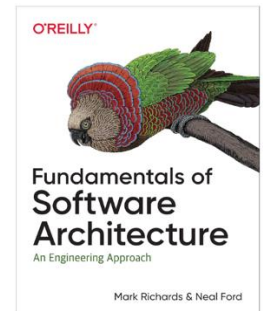
Status

Proposed

means the decision must be approved by another decision-maker.

Accepted

means the decision has been approved and is ready for implementation.



Superseded

means the decision has been changed and superseded by another ADR.

Template for ADRs - Status

Note

Superseded status always assumes the prior ADR status was accepted; in other words, a proposed ADR would never be superseded by another ADR but rather continued to be modified until accepted.

Example of status change

ADR 42. Use of Asynchronous Messaging Between Order and Payment Services

Status: Superseded by 68

ADR 68. Use of REST Between Order and Payment Services

Status: Accepted, supersedes 42

Template for ADRs

Context

specifies the forces at play. In other words, “What situation is forcing me to make this decision?”

It also includes the possible solutions/alternatives.

Template for ADRs

Context

Example of context

The order service must pass information to the payment service to pay for an order currently being placed.
This could be done using REST or asynchronous messaging.

Alternatives



Example of context

We need to simplify how we store customer survey responses. The data currently resides in a relational store, and its rigid schema requirements have become challenging as we evolve the surveys (for example, introducing different or extended surveys for our premium customers). There are various options available to us, like the JSONB data type in PostgreSQL or document stores such as MongoDB.

Alternatives



Template for ADRs

Decision

contains the architecture decision, along with a full justification for the decision

Note

- 1) More emphasis on the why rather than the how
- 2) Knowing why a decision was made and the corresponding justification for the decision **helps people better understand the context of the problem** and **avoids possible mistakes** through refactoring to another solution that might produce issues.

Template for ADRs

Decision

Example of decisions

We will use a document database for the customer survey.

The marketing department requires a faster, more flexible way to make changes to the customer surveys.

Moving to a document database will provide better flexibility and speed and will better facilitate changes by simplifying the customer survey user interface.



Justification

Template for ADRs

Consequences

documents the overall impact (good and bad) of an architecture decision

Example of a decision title

ADR 91. Use of Asynchronous (fire-and-forget) Messaging to Post a Review on a Website.

Example of a decision's consequence

This decision is to **significantly increase the responsiveness** of the post review request from 3,100 milliseconds to 25 milliseconds because users would not need to wait for the actual review to be posted (only for the message to be sent to a queue).

Template for ADRs

Compliance

describes how the architecture decision will be measured and governed from a compliance perspective.

Notes

include various metadata about the ADR, such as the following:

- Original author
- Approval date
- Approved by
- Superseded date
- Last modified date
- Modified by
- Last modification

ADR - Example

This is an example from the online auction system

Design Problem: What communication style should be used for inter-service communications?

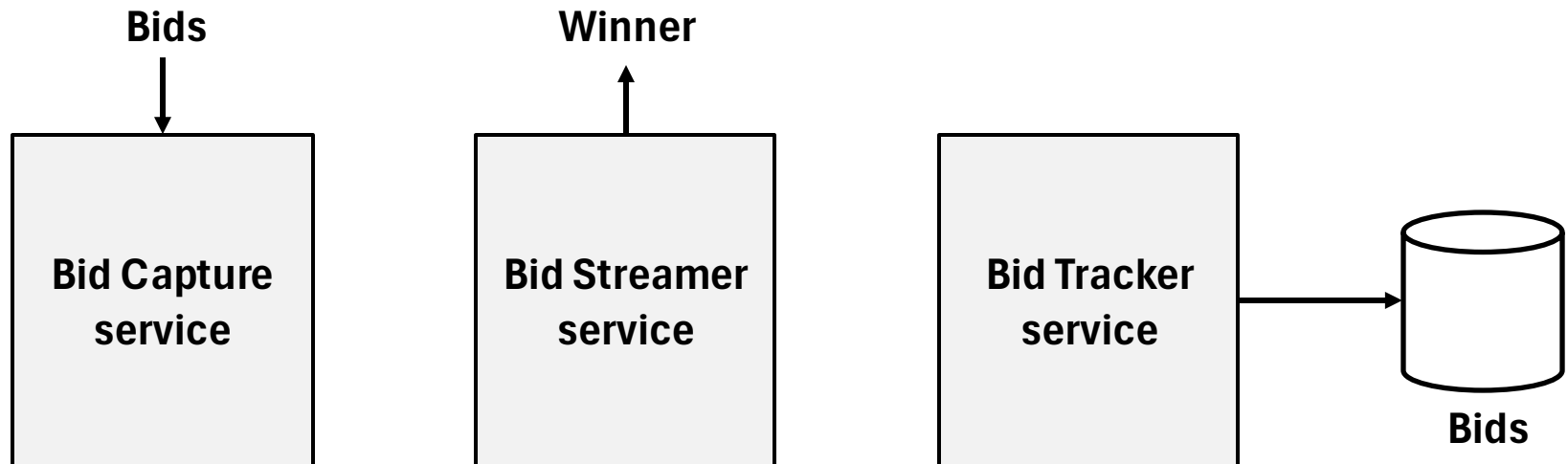


Image Source: Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2025.

ADR - Example

ADR 76. **Separate Queues** for Bid Streamer and Bidder Tracker Services

Status

Accepted

Context

The Bid Capture service, upon receiving a bid from an online bidder or a live bidder via the auctioneer, must forward that bid to the Bid Streamer service and the Bid Tracker service. This could be done using a

- REST**
- Single topic (pub/sub)**
- Separate queues (point-to-point) for each service**

ADR - Example

Option 1. REST API

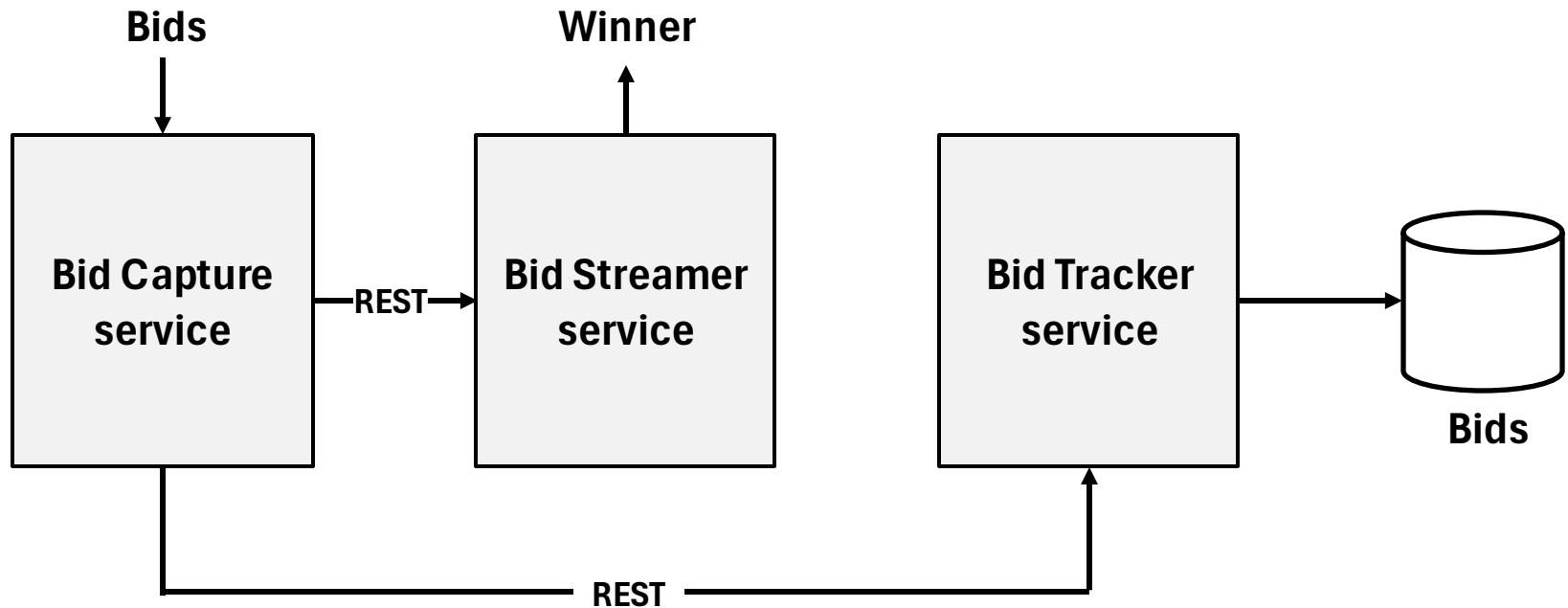


Image Source: Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2025.

ADR - Example

Option 2. Single topic (pub/sub)

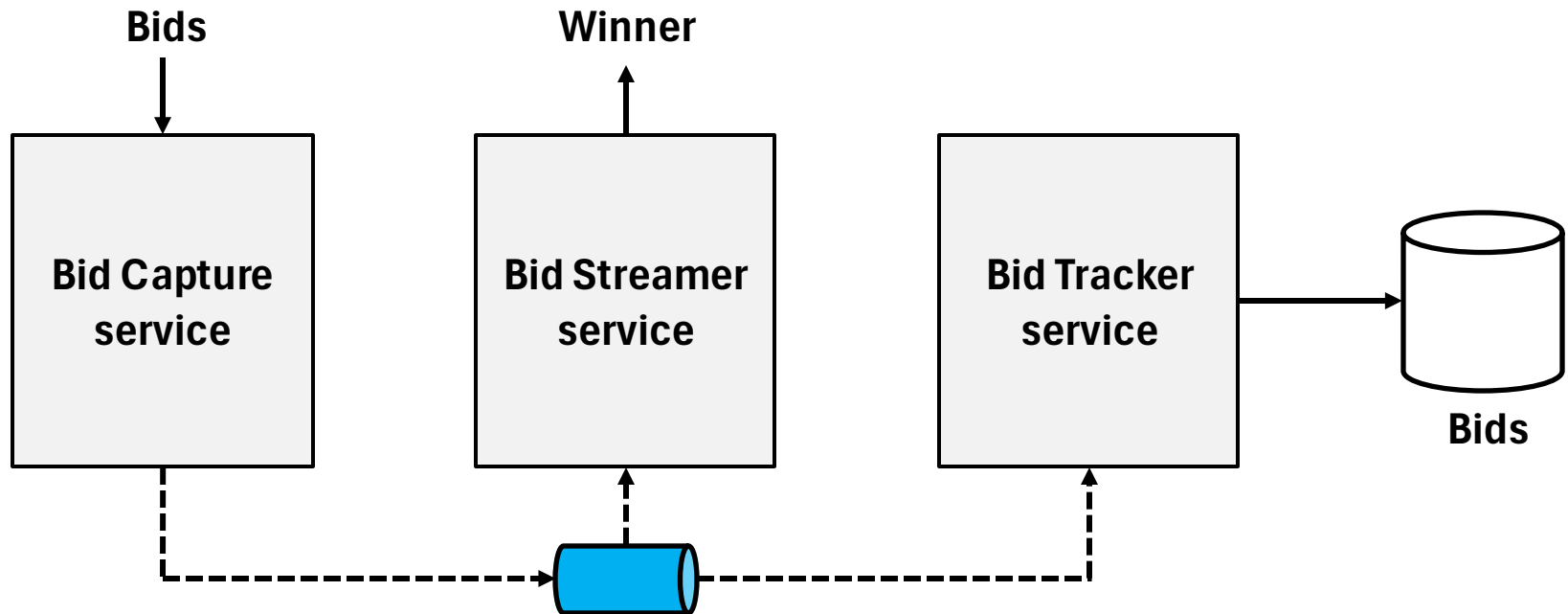


Image Source: Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2025.

ADR - Example

Option 3. Separate queues (point-to-point) for each service

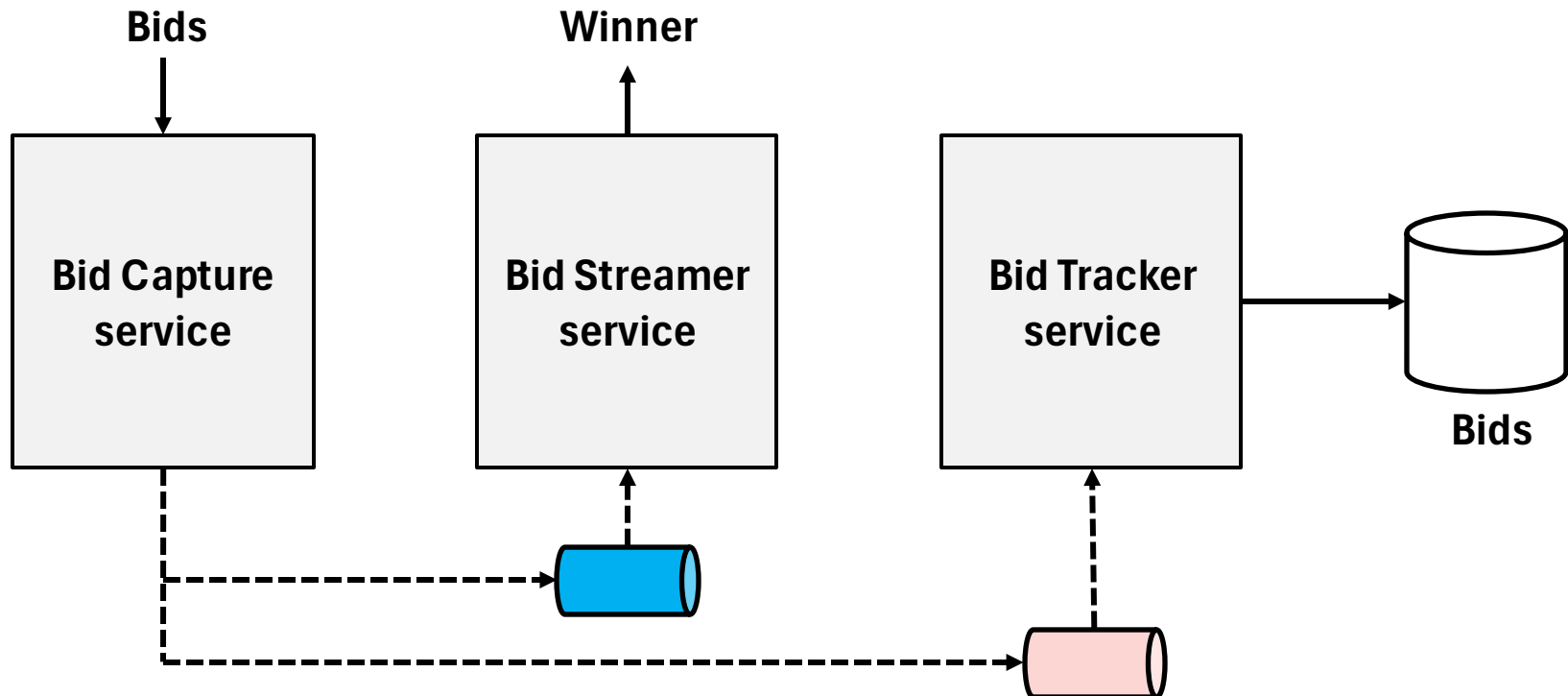


Image Source: Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2025.

ADR - Example

Decision

We will use **separate queues** for the Bid Streamer and Bidder Tracker services.

- The Bid Capture service **does not need any information back** from the Bid Streamer service or the Bid Tracker service (communication is only one-way).
- The Bid Streamer service must receive the **bids in the exact order** they were accepted by the Bid Capture service. Using messaging and queues automatically guarantees the bid order for the stream by leveraging first-in, first out (FIFO) queues.

ADR - Example

Decision (continued)

- In auctions, **multiple bidders can place the same bid amount** (e.g., multiple people saying “100”).
- For **Bid Steamer** service:
 - Only the first bid for a given amount needs to be displayed (to avoid duplicates on screen).
 - If this were done with a pub/sub topic, the Bid Steamer would have to **filter duplicates itself**, keeping track of state across servers (complex, error-prone).
- For **Bid Tracker** service:
 - It needs to **record every bid (duplicates included)**, because the auction history must be complete.
- Separate queues let each service handle bids differently **without interfering with each other**.

ADR - Example

Consequences

We will require clustering and high availability of the message queues.

Internal bid events will bypass security checks done in the API layer.

UPDATE: Upon review on April 14, 2020, the architecture review team decided that this was an acceptable trade-off and that **no additional security checks would be needed** for bid events between these services.

Compliance

We will use periodic manual code and design reviews to ensure that asynchronous point-to-point messaging is being used between these services.

Notes

Author: Jon. M

References

- Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2020 (First Edition).
- Richards, M., Ford, N., Fundamentals of Software Architecture: An Engineering Approach, O'Reilly Media, 2025 (Second Edition).
- Pautasso, C., Software Architecture- Visual Lecture Notes, LeanPub, 2023 (<https://leanpub.com/software-architecture/>)
- Brown, Simon. "The C4 model for visualising software architecture" (c4model.com)
- <https://docs.structurizr.com/>
- The University of Queensland's Software Architecture Course Materials, @ Richard Thomas, CC BY-SA 4.0 (<https://github.com/CSSE6400>)
- Philippe Kruchten, Architectural Blueprints- 4+1= View Model of Software Architecture, IEEE Software 12 (6). November 1995, pp. 42-50
- Bass, L., Clements, P., Kazman, R., Software Architecture in Practice, Addison-Wesley, 2021.
- Gandhi, R., Richards, M., Ford, N., Head First Software Architecture, O'Reilly Media, Inc. 2024