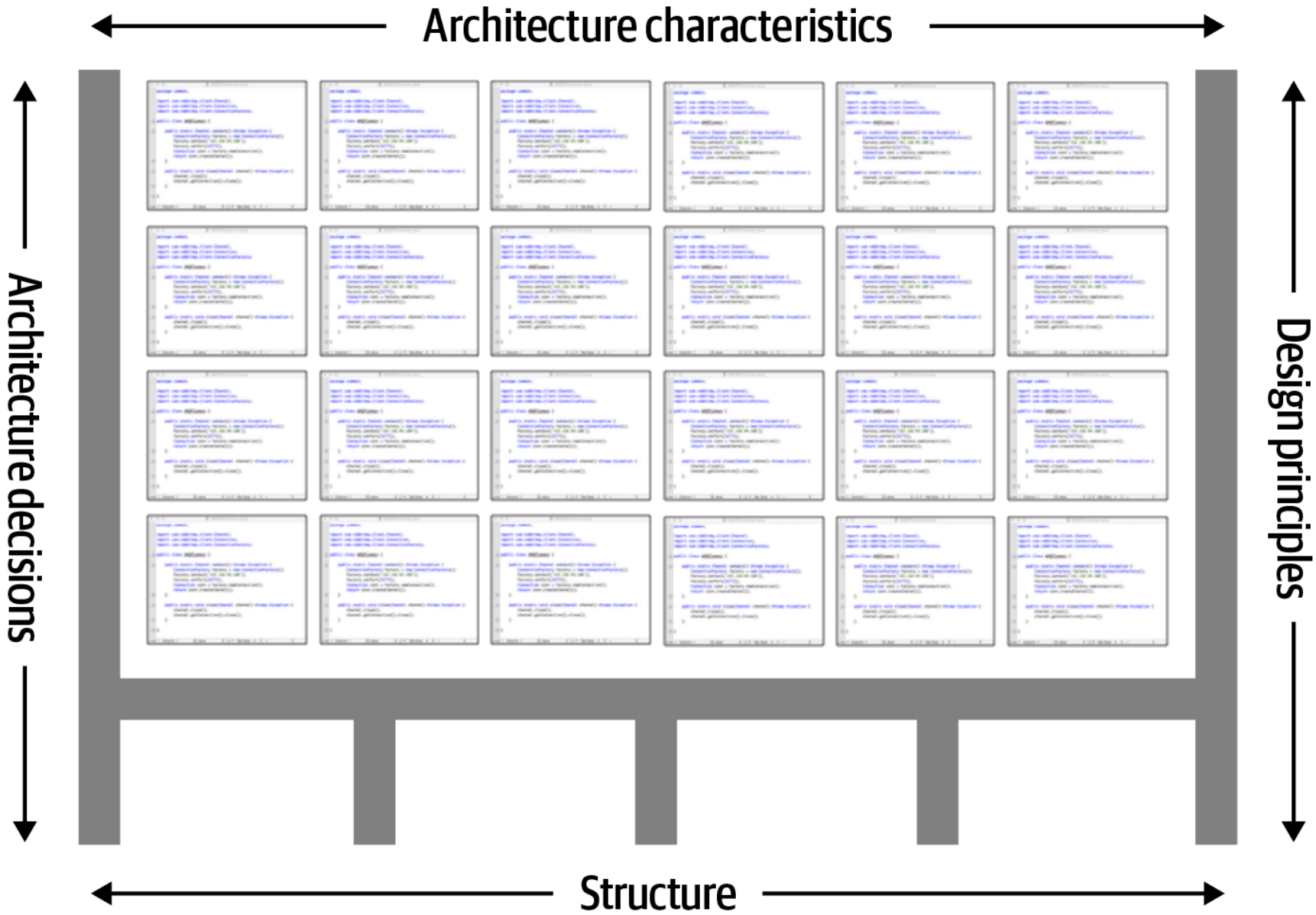
Defining Software Architecture

Software architecture consists of the **structure** of the system, combined with **architecture characteristics (“-ilities”)** the system must support, **architecture decisions**, and finally **design principles**.


Architecture characteristics
Architecture decisions
Design principles
Structure

Structure: the type of architecture style (or styles) the system is implemented in (such as microservices and layered).

Architecture characteristics: define the success criteria of a system (such as availability and scalability).

Architecture decisions: define the rules for how a system should be constructed, form the constraints of the system and direct the development teams on what is and what isn't allowed (such as "only the business and services layer can access the persistence layer").

Design principle: is a guideline rather than a hard-and-fast rule (such as "whenever possible, leverage async messaging between services to increase performance").

Laws of Software Architecture

> *Everything in software architecture is a trade-off.*
>
> —First Law of Software Architecture

Nothing exists on a nice, clean spectrum for software architects. Every decision must take into account many opposing factors.

> *If an architect thinks they have discovered something that isn't a trade-off, more likely they just haven't identified the trade-off yet.*
>
> —Corollary 1

> *Why is more important than how.*
>
> —Second Law of Software Architecture

Auction system example

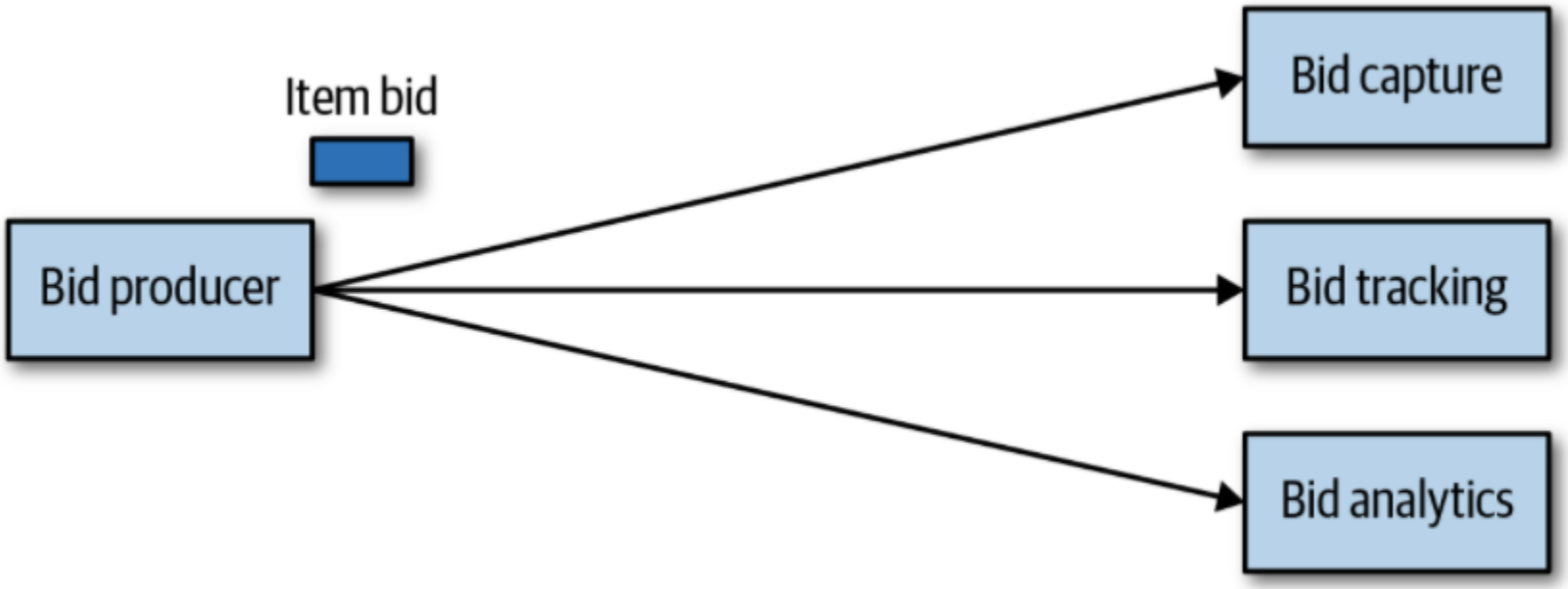


Figure 2-7. Auction system example of a trade-off—queues or topics?

The Bid Producer service generates a bid from the bidder and then sends that bid amount to the Bid Capture, Bid Tracking, and Bid Analytics services.

This could be done by using queues in a point-to-point messaging fashion or by using a topic in a publish-and-subscribe messaging fashion.

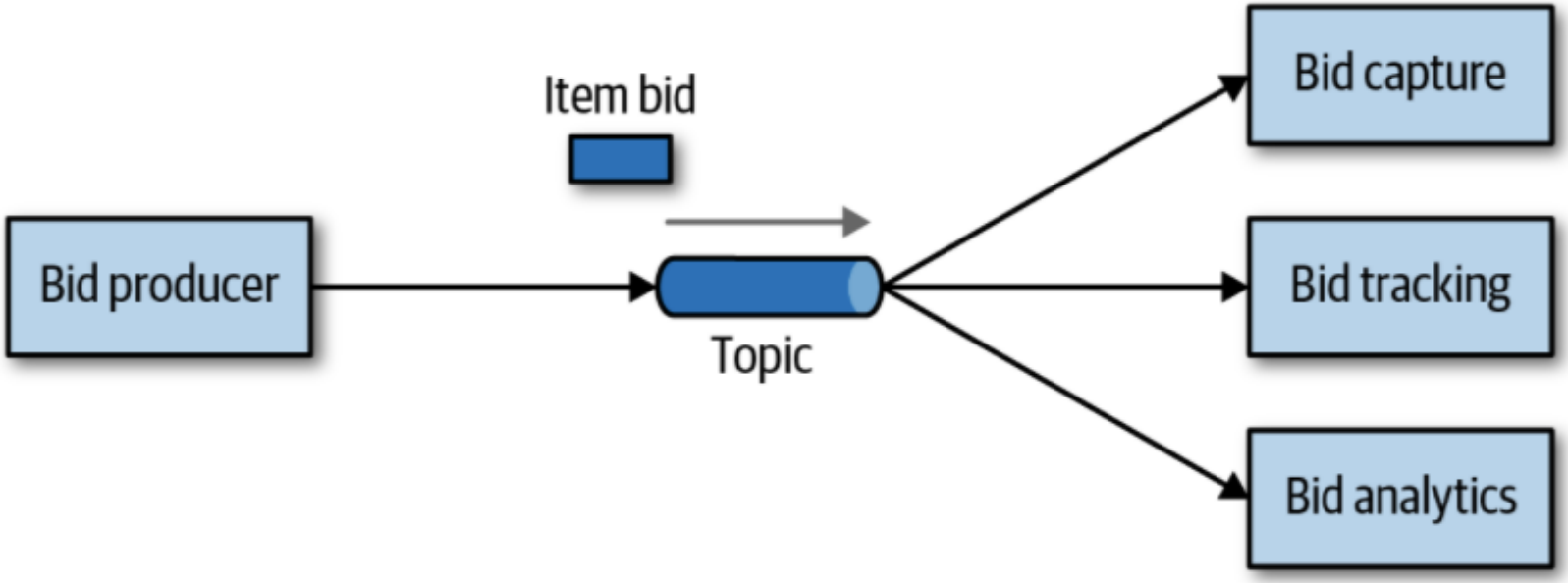


Figure 2-8. Use of a topic for communication between services

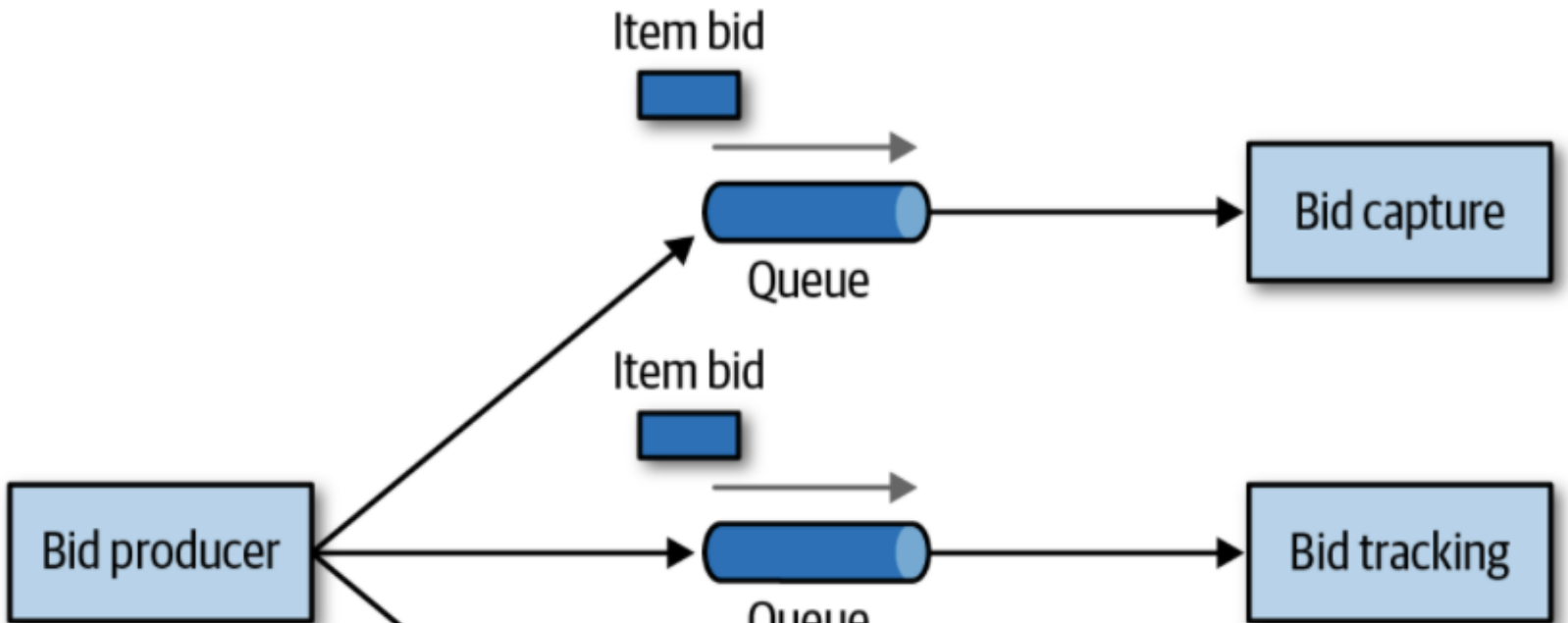


Table 2-1. Trade-offs for topics

Topic advantages	Topic disadvantages
Architectural extensibility	Data access and data security concerns
Service decoupling	No heterogeneous contracts
	Monitoring and programmatic scalability

Reference

Fundamentals of Software Architecture: An Engineering Approach by Mark Richards