# Event Driven Architecture

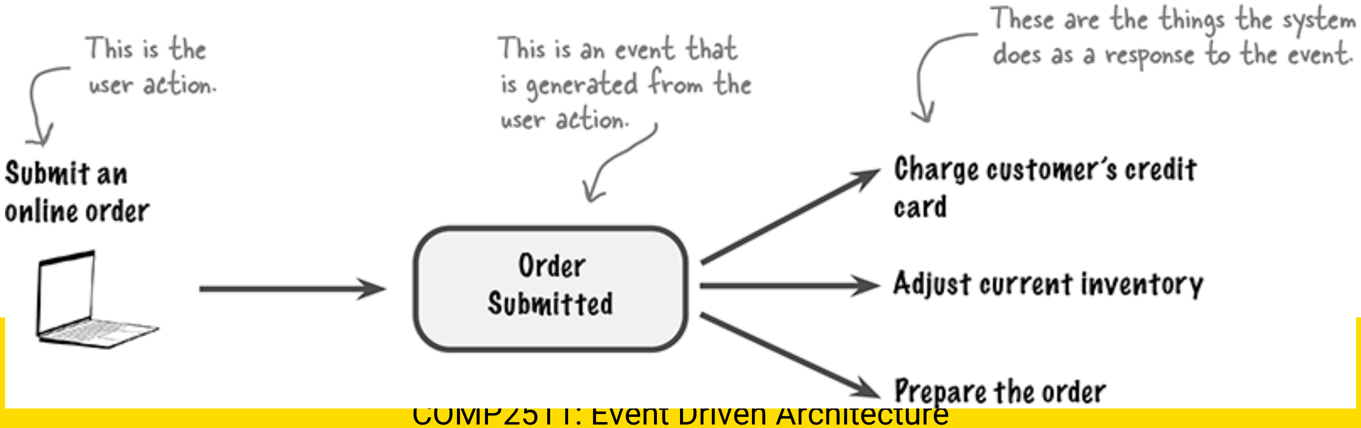
COMP2511, CSE, UNSW

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

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

# Introduction to Even-Driven Architecture

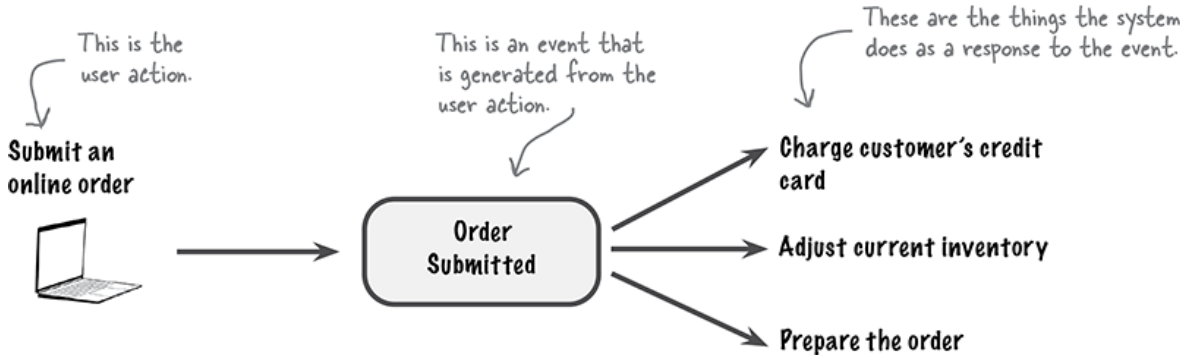
- Event-Driven Architecture (EDA) structures systems to respond to events, which are significant changes in system state.
- Unlike request-driven systems, EDA components don't directly call each other.
- Example:
  - An e-commerce system where placing an order triggers inventory updates, payment processing, and shipping—all asynchronously.


This is the user action.
This is an event that is generated from the user action.
These are the things the system does as a response to the event.
Submit an online order
Order Submitted
Charge customer's credit card
Adjust current inventory
Prepare the order

# What is an Event

- An event represents something that has already occurred and carries data about it.
- Events are immutable and often used as triggers.

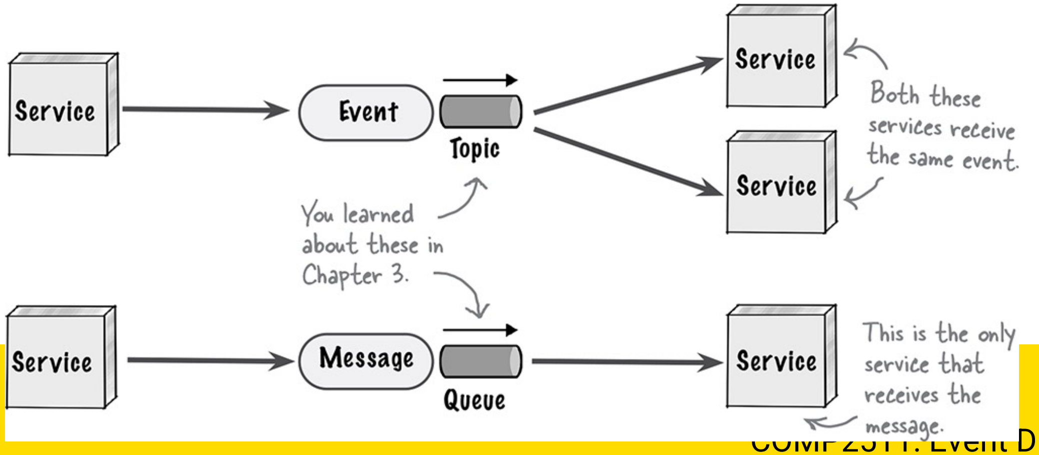
- Example:
  - "User Registered" event might include the user ID, name, and email.


This is the user action.
Submit an online order
This is an event that is generated from the user action.
Order Submitted
These are the things the system does as a response to the event.
Charge customer's credit card
Adjust current inventory
Prepare the order

# Even vs. Messages

- ❖ Events broadcast that something happened, with no expectation of response.
- ❖ Messages are more targeted, often demanding action.

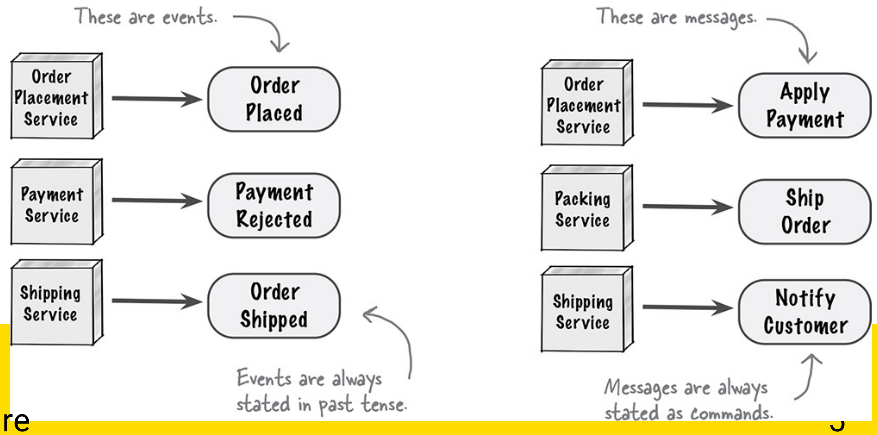
- ❖ Example:
  - ○ *Event*: "Item Added to Cart" (anyone can listen).
  - ○ *Message*: "Process Payment" sent directly to payment service.



Both these services receive the same event.

You learned about these in Chapter 3.

This is the only service that receives the message.



These are events.

Events are always stated in past tense.

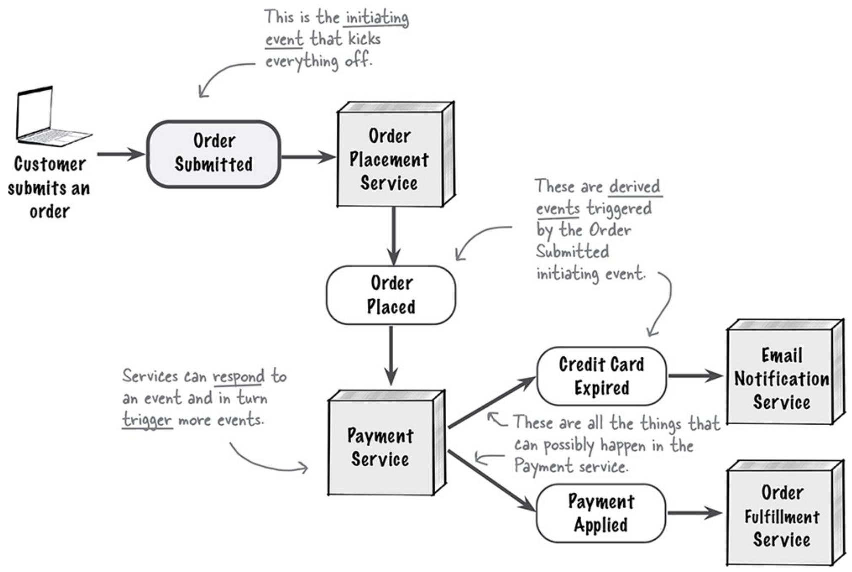
These are messages.

Messages are always stated as commands.

# Initiating and Derived Events

- Initiating events are triggered by users or external systems.
- Derived events are consequences of those events and triggered by services

- Example:
  - *Initiating*: "Order Placed"
  - *Derived*: "Payment Authorized", "Inventory Deducted", "Shipping Scheduled"



This is the initiating event that kicks everything off.
Customer submits an order
Order Submitted
Order Placement Service
These are derived events triggered by the Order Submitted initiating event.
Order Placed
Services can respond to an event and in turn trigger more events.
Payment Service
Credit Card Expired
Email Notification Service
These are all the things that can possibly happen in the Payment service.
Payment Applied
Order Fulfillment Service

# Why Publish Events Others May Not Care About?

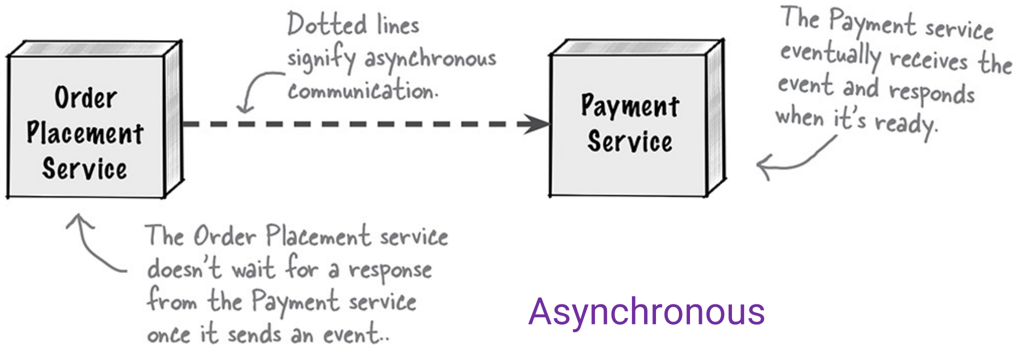
- Broadcasting all events allows new services to listen without modifying existing systems.

- Example:
  - Initially, only billing listens to "Order Placed".
  - Later, analytics can subscribe to the same event to track order trends.

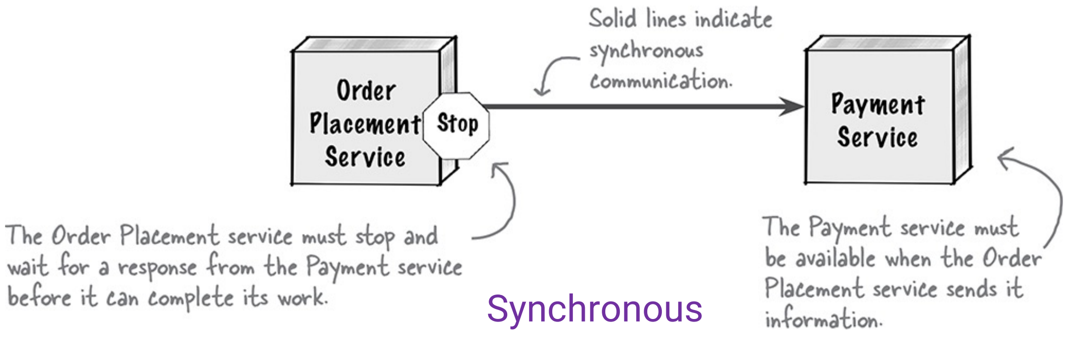
# Asynchronous vs. Synchronous Communication

- In synchronous calls, the sender waits for a response.
- In asynchronous communication, it continues immediately after sending.

- Example:
  - *Synchronous*: REST API call to get shipping quote
  - *Asynchronous*: Order service emits "Order Placed" and moves on


Order Placement Service
Dotted lines signify asynchronous communication.
Payment Service
The Payment service eventually receives the event and responds when it's ready.
The Order Placement service doesn't wait for a response from the Payment service once it sends an event..

Asynchronous


Order Placement Service
Stop
Solid lines indicate synchronous communication.
Payment Service
The Order Placement service must stop and wait for a response from the Payment service before it can complete its work.
The Payment service must be available when the Order Placement service sends it information.

Synchronous

# Benefits of Asynchronous Communication

- Loose coupling allows services to operate and scale independently, increasing speed and fault tolerance.

Example:

- With async processing, an online store confirms an order in 600ms; with sync processing, it takes 1800ms.

# Trade-Offs of Async Communication

- Asynchronous systems complicate debugging and tracing since there's no immediate response.

- Example:
  - When inventory update fails, the order service might not know.
  - You need to monitor event failures separately.

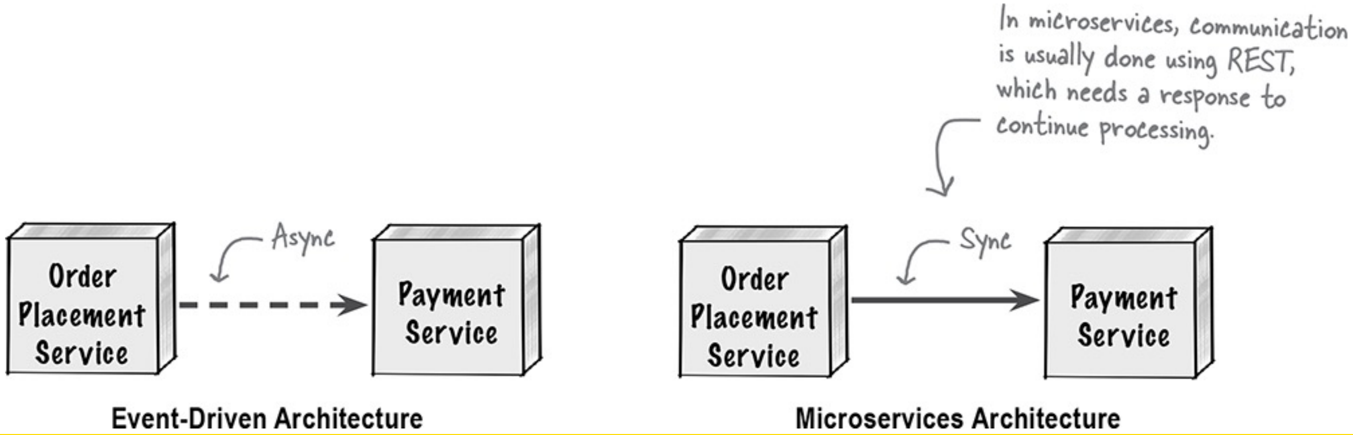
# Database Topologies in EDA

- How services access and manage data affects modularity and scaling.
- Examples:
  - Monolithic DB: All services write to one database (fast, tightly coupled).
  - Domain-Partitioned: Related services share DB (moderately coupled).
  - DB-per-Service: Each microservice owns its DB (fully decoupled, but complex joins require events).

# EDA vs. Microservices

- Microservices focus on small, self-contained services communicating via HTTP or RPC.
- EDA emphasizes event-based coordination.

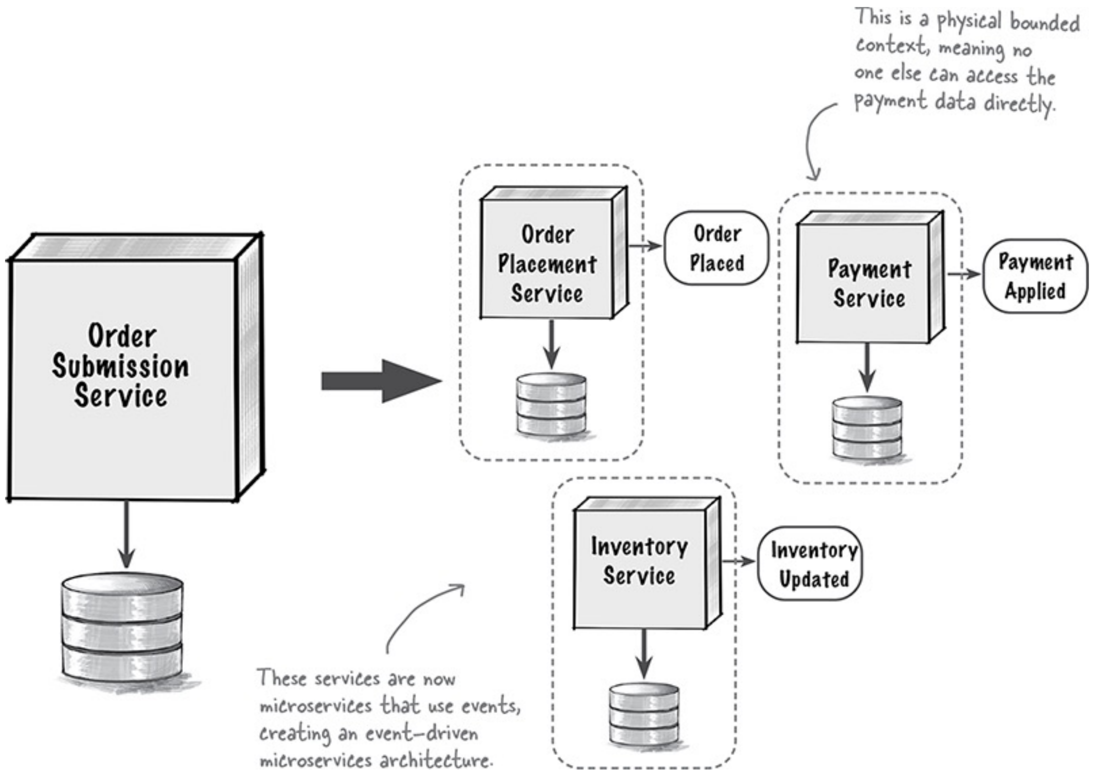
- Example:
  - Microservice model uses REST to trigger payment.
  - EDA emits "Order Placed", and payment service listens and reacts.


Order Placement Service
Async
Payment Service
Event-Driven Architecture
In microservices, communication is usually done using REST, which needs a response to continue processing.
Order Placement Service
Sync
Payment Service
Microservices Architecture

# Event-Driven Microservices

- This hybrid model combines independent services with event communication, boosting flexibility.

- Example:
  - Inventory, Shipping, and Billing each own their DB and listen to "Order Placed" to act asynchronously.



This is a physical bounded context, meaning no one else can access the payment data directly.

These services are now microservices that use events, creating an event-driven microservices architecture.

# EDA Challenges

- EDA introduces challenges with observability and testing due to distributed asynchronous operations.

- Examples:
  - Event debugging is complex—errors are not immediate.
  - Testing sequences requires simulating full event flows.

# EDA Advantages

- EDA shines in environments requiring responsiveness, scalability, and autonomy.

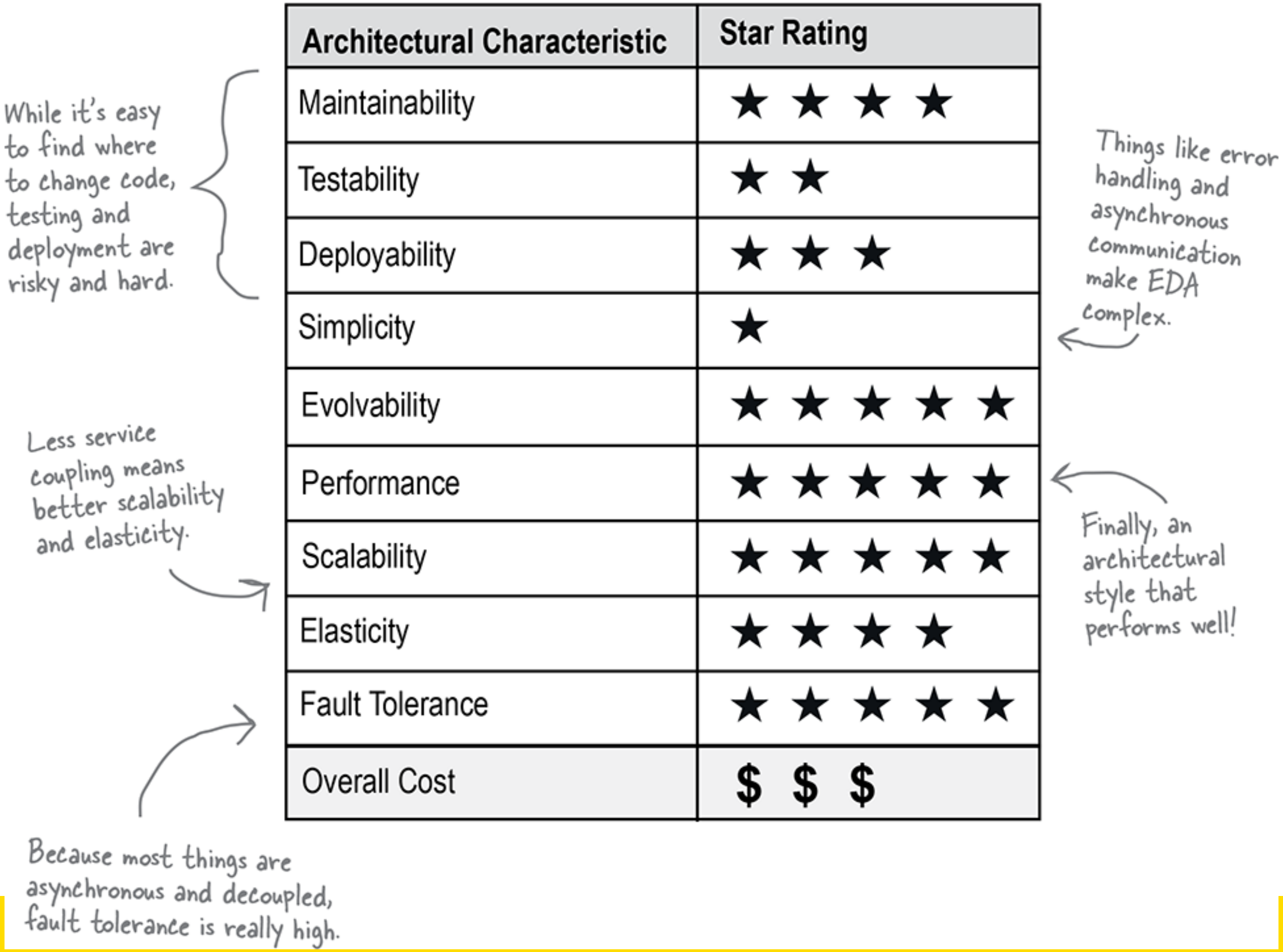
- Examples:
  - Maintainability: Add new features without changing existing services.
  - Performance: Events processed in parallel.
  - Scalability: Individual components scale independently.

# Key Concepts

- EDA emphasizes responsiveness and extensibility but requires thoughtful design to manage complexity.

- Key Points:
  - Events = Immutable notifications of state change
  - Asynchronous = Fire-and-forget
  - Combined with microservices for modern architectures

# Event-driven Architecture Star Ratings



| Architectural Characteristic | Star Rating |
|---|---|
| Maintainability | ★ ★ ★ ★ |
| Testability | ★ ★ |
| Deployability | ★ ★ ★ |
| Simplicity | ★ |
| Evolvability | ★ ★ ★ ★ ★ |
| Performance | ★ ★ ★ ★ ★ |
| Scalability | ★ ★ ★ ★ ★ |
| Elasticity | ★ ★ ★ ★ |
| Fault Tolerance | ★ ★ ★ ★ ★ |
| Overall Cost | $ $ $ |

While it's easy to find where to change code, testing and deployment are risky and hard.

Things like error handling and asynchronous communication make EDA complex.

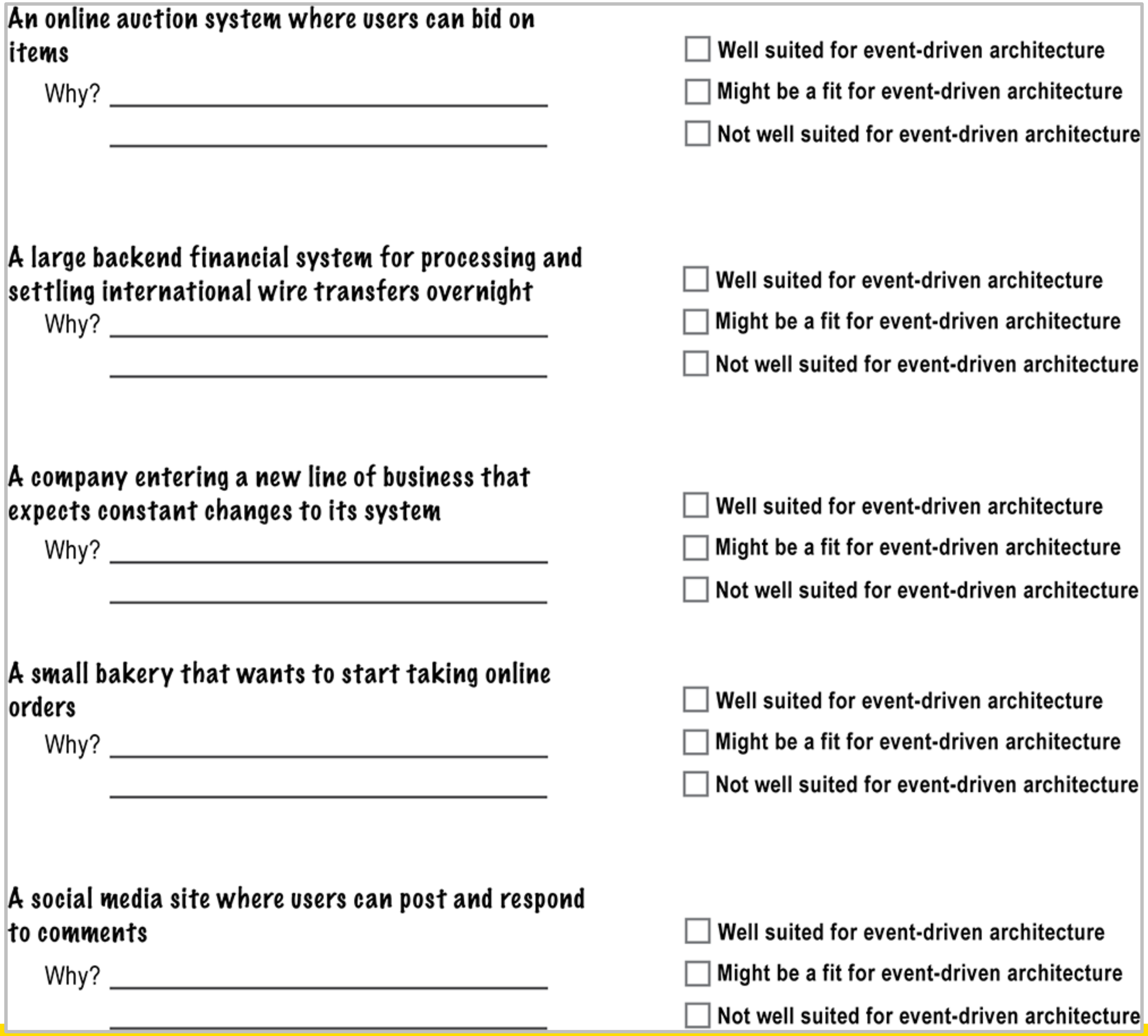
Less service coupling means better scalability and elasticity.

Finally, an architectural style that performs well!

Because most things are asynchronous and decoupled, fault tolerance is really high.

# Exercise

Which of the following systems might be well suited for the event-driven architectural style, and why?



An online auction system where users can bid on items

Why? ______________________________

______________________________

□ Well suited for event-driven architecture
□ Might be a fit for event-driven architecture
□ Not well suited for event-driven architecture

A large backend financial system for processing and settling international wire transfers overnight

Why? ______________________________

______________________________

□ Well suited for event-driven architecture
□ Might be a fit for event-driven architecture
□ Not well suited for event-driven architecture

A company entering a new line of business that expects constant changes to its system

Why? ______________________________

______________________________

□ Well suited for event-driven architecture
□ Might be a fit for event-driven architecture
□ Not well suited for event-driven architecture

A small bakery that wants to start taking online orders

Why? ______________________________

______________________________

□ Well suited for event-driven architecture
□ Might be a fit for event-driven architecture
□ Not well suited for event-driven architecture

A social media site where users can post and respond to comments

Why? ______________________________

______________________________

□ Well suited for event-driven architecture
□ Might be a fit for event-driven architecture
□ Not well suited for event-driven architecture