

Open in app ↗



Search

Photo by [Greg Rosenke](#) on [Unsplash](#)

Architecture Style : Event-Driven Architecture (EDA)



Pier-Jean Malandrino

Published in Scub-Lab

5 min read · Mar 4, 2024



Listen



Share



More

Event-Driven Architecture (EDA) is a software architecture style promoting the production, detection, consumption, and reaction to events. This paradigm has become increasingly popular in the development of scalable, loosely coupled, and flexible systems. EDA enables applications to asynchronously communicate with each other through events, enhancing responsiveness and facilitating a more

dynamic application behavior. This paper explores the concept of EDA, its benefits, trade-offs, and concludes with a synthesis of its importance in modern software development.

What is Event-Driven Architecture?

EDA is structured around the generation and handling of events, discrete occurrences signifying a change in state or the occurrence of an action within a system. An event-driven system decouples the event producer, which emits an event, from the event consumer, which reacts to the event. This decoupling is achieved through the use of an event broker or bus that intermediates between producers and consumers, routing events to the appropriate consumers based on subscriptions or patterns.

Components of EDA

Event Producers: Entities that generate events. They are responsible for detecting or causing changes within the system which are then encapsulated as events.

Event Consumers: Entities that react to events. They subscribe to specific types of events and process them as they occur.

Event Broker/Bus: Middleware that decouples event producers from consumers, ensuring the delivery of events from the former to the latter.

Events: The central element of EDA, representing a significant change or occurrence within the system.

Benefits of Event-Driven Architecture

Scalability: EDA supports horizontal scaling as components can be scaled independently, allowing systems to handle increased loads by adding more consumers or producers without significant redesign.

Flexibility and Agility: Changes to the system, such as adding new event types or processing logic, can be implemented with minimal impact on existing components.

Loose Coupling: Producers and consumers communicate through events without direct knowledge of each other, promoting modularity and making the system easier to extend and maintain.

Responsiveness: By processing events asynchronously, systems can respond to inputs or changes in state more quickly, enhancing user experience and system performance.

Resilience: The decoupled nature of EDA allows parts of the system to fail or be upgraded without affecting the entire system’s availability.

Trade-Offs of Event-Driven Architecture

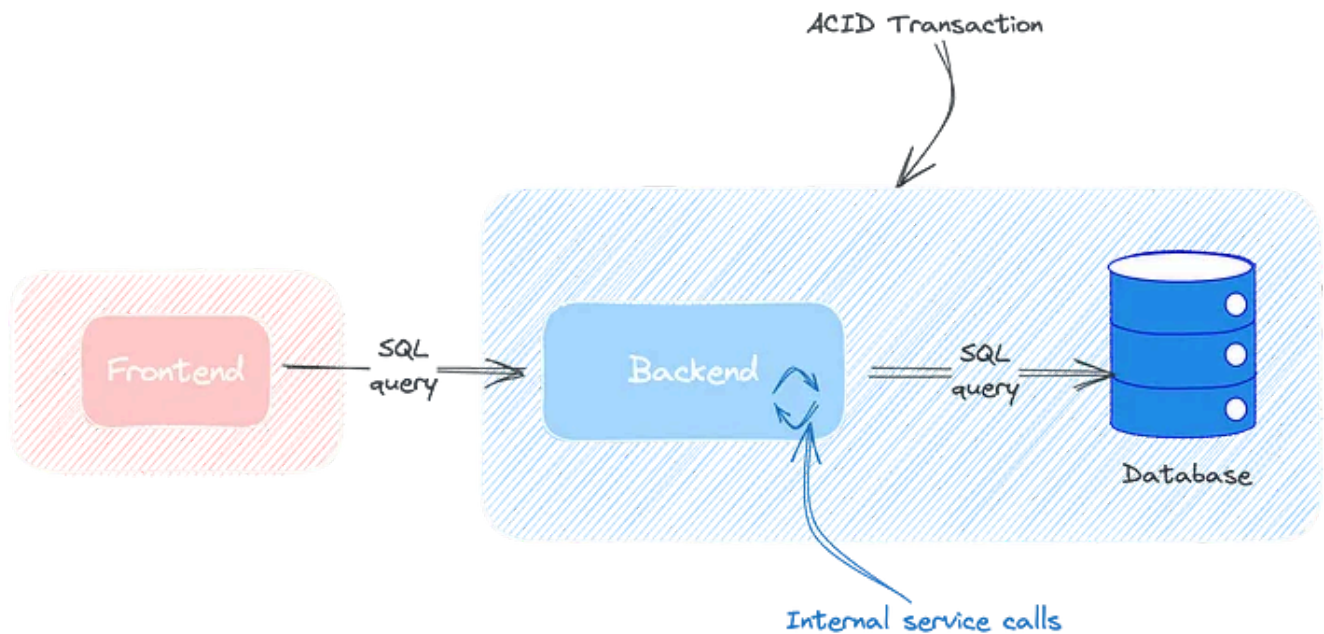
Complexity: The asynchronous nature of event-driven systems can make them more complex to understand, debug, and monitor compared to synchronous systems.

An example is the compensation of a failed message, you should determine a policy of retries for the message and then if all attempts fail you should determine what to do with that message. Is it a producer’s fault, or your broker’s, etc.?

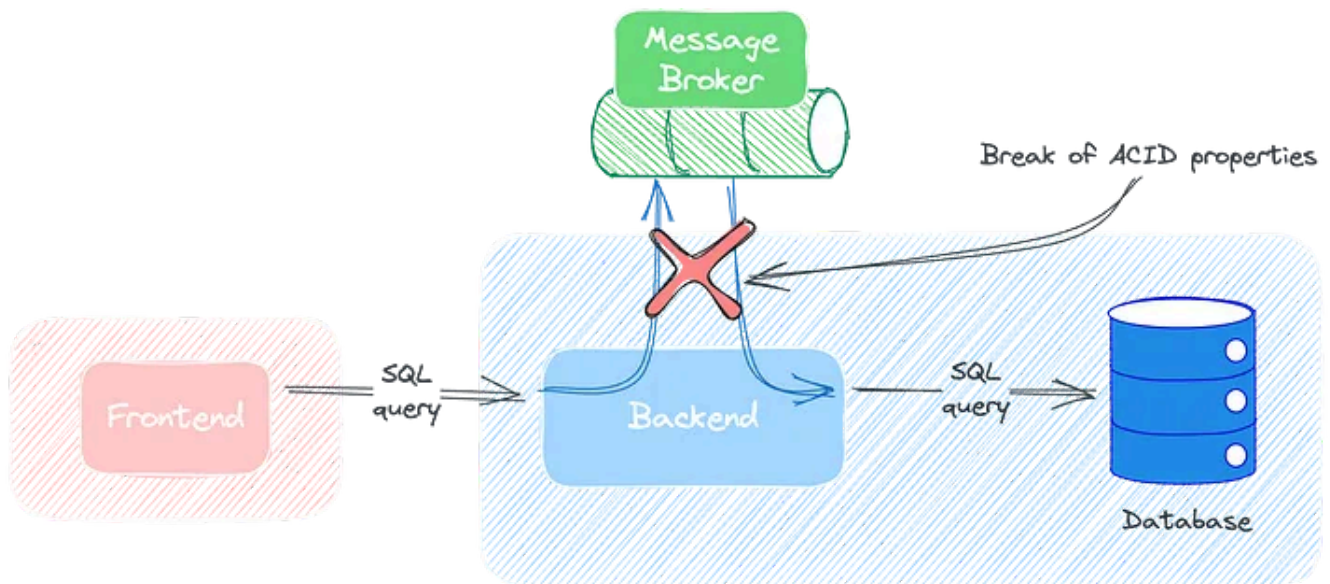
Data Consistency: Ensuring data consistency across different services can be challenging, especially in distributed systems where events may be processed out of order.

In EDA, ACID properties are more difficult to guarantee. You can use Saga patterns to approach it, but it is much more complex than traditional transactions.

<p>Database : ACID transactions</p> <p>This paper discusses transactions, which are essential to information systems because they encapsulate data...</p> <p>lab.scub.net</p>	
--	--



Synchronous REST communication with direct SQL query



EDA architecture breaking ACID properties

Event Management: As systems grow, managing a large number of event types, ensuring event schema compatibility, and handling event versioning can become difficult.

Note: The trade-off mentioned above is accurate but needs to be considered in context. Managing a large number of events is challenging, but this would also be the case with any other strategy. Therefore, the complexity is due to the context of use.

When to use it ?

Deciding to implement an Event-Driven Architecture (EDA) should be based on a thorough analysis of the system's specific requirements. While EDA offers numerous benefits, it also introduces considerable complexity. It's essential to evaluate whether the investment in developing and maintaining such an architecture is justified.

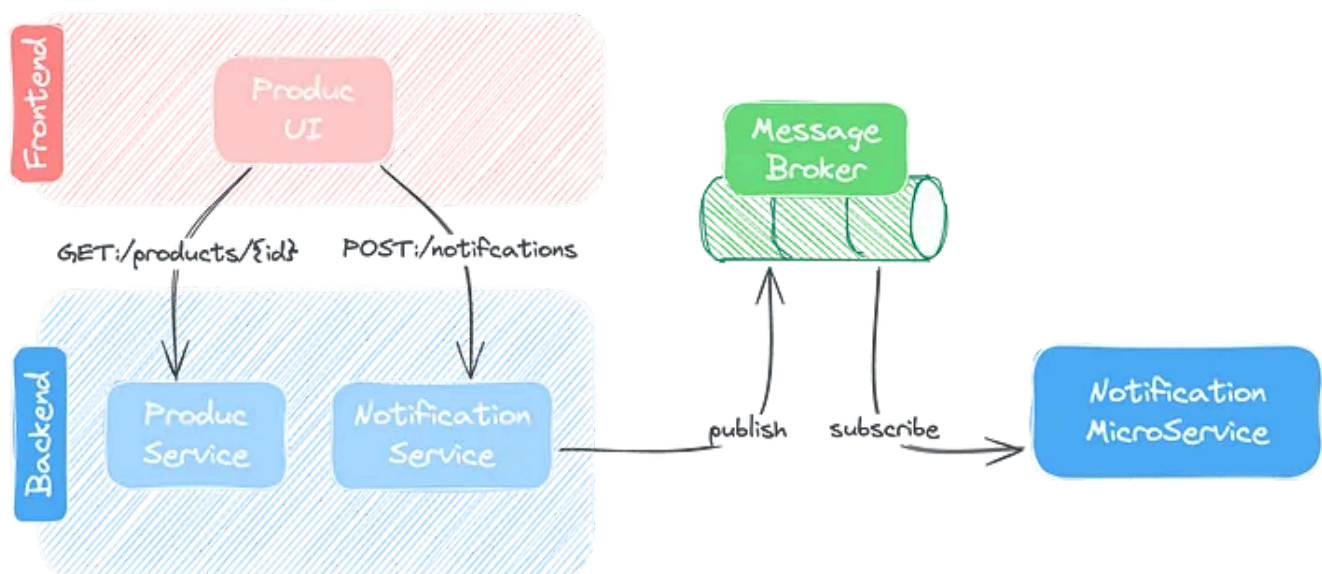
It's crucial to recognize that EDA may not be necessary for the entire system but could be applied to specific parts of your architecture. For instance, an EDA could be utilized for managing a notification system, while a synchronous REST API could handle other exchanges.

Furthermore, EDA can be integrated with various architectural styles, including SOA, Microservices, Modulith, or even Monolith. Although some styles may be more compatible with EDA, it's important to understand that while architectural styles like SOA and Microservices are structural, EDA focuses on communication. This distinction highlights that EDA often requires a decentralized and decoupled approach.

Architecture Style : Modulith (vs Microservices)

Modulith architecture: A blend of Monolith's cohesion and Microservices' independence. Break your monolith into pieces...

levelup.gitconnected.com



Basic representation of an EDA

Conclusion

In conclusion, Event-Driven Architecture (EDA) emerges as a pivotal paradigm in the realm of software development, especially for systems requiring high degrees of scalability, flexibility, and responsiveness.

By enabling asynchronous communication and fostering decoupling between system components, EDA empowers organizations to build adaptable and robust systems capable of navigating the complexities and demands of the modern digital landscape.

However, the adoption of EDA is not without its challenges, notably in terms of architectural complexity and the intricacies of event management and data consistency. Therefore, the decision to implement EDA should be a deliberate one, grounded in a comprehensive understanding of the system's requirements and the strategic objectives it seeks to achieve.

EDA's potential to enhance system performance and agility makes it a compelling choice for certain scenarios, yet its effectiveness is contingent upon a thoughtful consideration of its benefits against the operational and developmental complexities it introduces.

As software architecture continues to evolve, EDA stands as a testament to the industry's ongoing pursuit of more dynamic, scalable, and resilient systems.

I am the CTO and Head of an architectural unit at Scub. I participate in the development of technological strategy, design solutions, and lead R&D projects.

Thank you for reading! If you enjoyed this article, please feel free to 🙌 and help others find it. Please do not hesitate to share your thoughts in the comments section below.

Scub Lab

Thank you for being a part of our community! Before you go:

- Be sure to **clap** and **follow** the writer! 🙌
- You can find even more content at lab.scub.net 🚀

- *Sign up for our free weekly newsletter.* 📧
- *Follow us on Twitter(X), LinkedIn, and our site web.*

Programming

Technology

Software Development

Software Architecture

Microservices

[Edit profile](#)

Written by Pier-Jean Malandrino

3.5K Followers · Editor for Scub-Lab

CTO & Head of Architecture at a digital firm, I drive technological strategy, design innovative solutions, & lead R&D projects.

More from Pier-Jean Malandrino and Scub-Lab



Pier-Jean Malandrino in Scub-Lab

Architecture : The cheat sheet

This paper presents a concise summary of various software architecture patterns, methods and models.

5 min read · Dec 25, 2023

1.92K



6

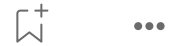



Boris Bodin in Scub-Lab

What is it ?—Gherkin for test naming

Learn how using Gherkin syntax for test naming improves test readability and maintenance in SpringBoot/Angular applications.

★ · 9 min read · Nov 13, 2023



 Boris Bodin in Scub-Lab

Focus on: Factory Pattern

Demystifying the Builder Pattern: A practical guide for software architects and developers.

★ · 11 min read · Dec 11, 2023





Pier-Jean Malandrino in Scub-Lab

Architecture Anti-Patterns : The DARK side of the Architect

Amid the realm of logic and structured thought, architects, much like the mythical creatures of old, harbor a shadowy underside. Like the...

3 min read · Oct 31, 2023



434



7



See all from Pier-Jean Malandrino

See all from Scub-Lab

Recommended from Medium



Mirek Stanek in Practical Engineering Management

How to lead a team of senior engineers

Managing people who are more skillful than you

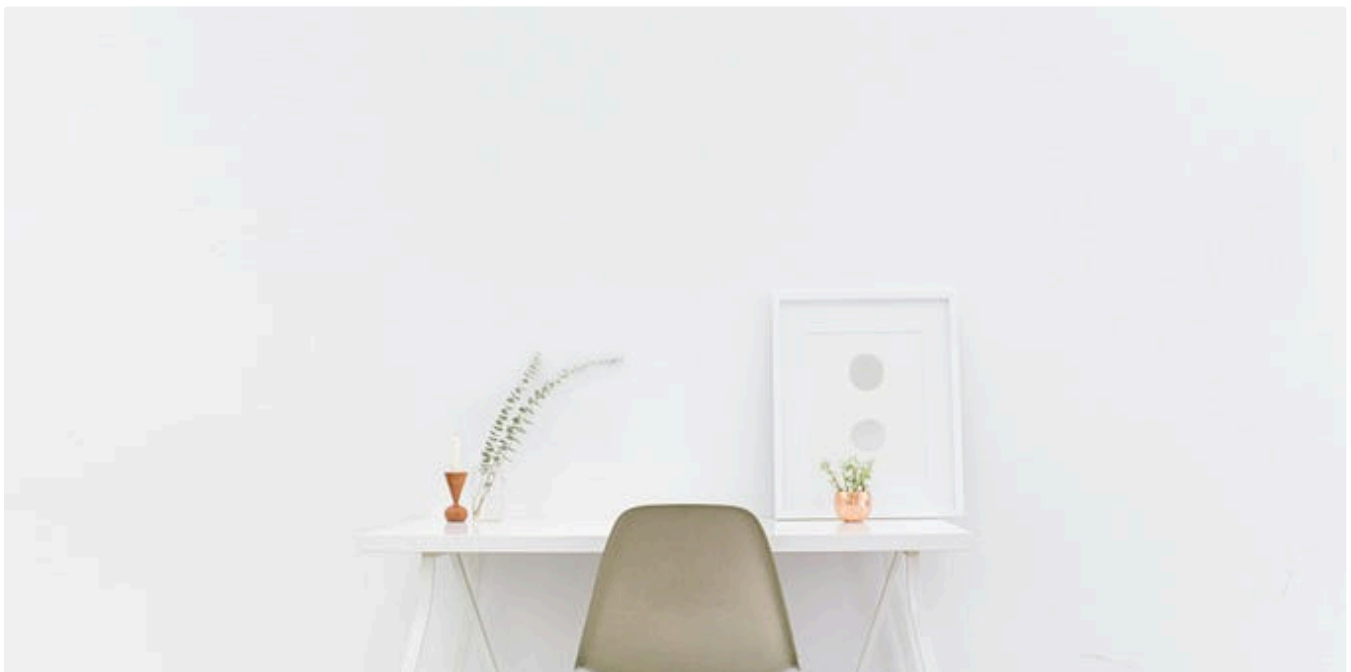
8 min read · Jan 17, 2024



1.6K



14



Aashish Peepra

How to design clean API interfaces

This is going to be a tough read. This contains too much code to read and no pictures. It's not for people with faint hearts. So read if...

7 min read · Mar 3, 2024


 525


 7







Lists


- 

General Coding Knowledge
20 stories · 1210 saves
- 

Stories to Help You Grow as a Software Developer
19 stories · 1053 saves
- 

Coding & Development
11 stories · 610 saves
- 

ChatGPT prompts
47 stories · 1547 saves

 Ali Arsanjani

The GenAI Reference Architecture

26 min read · Apr 29, 2024

 614

 6





 huizhou92 in Level Up Coding

If Google No Longer supports Golang

Analyzing the Potential Impact of Google's Discontinuation of Support for Go Language and Possible Future Scenarios

★ · 3 min read · May 7, 2024

 512  17

 Dario Radečić  in Towards Data Science


Python One Billion Row Challenge—From 10 Minutes to 4 Seconds

The one billion row challenge is exploding in popularity. How well does Python stack up?

★ · 10 min read · May 8, 2024

 3K  40


 Oliver Foster in Stackademic

Redis: How to Count the Number of Online Users?

My article is open to everyone; non-member readers can click this link to read the full text.

★ · 5 min read · May 2, 2024

 131  1

See more recommendations