

Home > Course > Design Your Software Architecture Using Industry-Standard Patterns > Event-Driven Architecture


Design Your Software Architecture Using Industry-Standard Patterns

4 hours  Hard

Last updated on 6/29/20



Event-Driven Architecture

 [Log in](#) or [subscribe](#) for free to enjoy all this course has to offer!

In this chapter, we are going to cover event-driven architecture, and in what situations it is a perfect solution. At the end of the chapter, I will show you a real-word example (RNC), and then you will solve a similar case applying what you've learned. Let's jump in!

What Is Event-Driven Architecture?



Have you ever wanted to buy something on a website only to realize the item was way out of your budget? Luckily, some websites can send you an alert when the price of an item goes down to a set value. All you have to do is enter an alert on the website: "If the item reaches X price, please notify me." It's like telling the site, "If X event happens, then do this."

Event-driven architecture works under the same principle: it produces, detects, and acts upon events in the system that are relevant to users. If no events happen, nothing happens in the system. This architecture defines these critical events as triggers. When these triggers happen, they cause specific behaviors - like sending you an alert when the item you want drops in price.

Moving forward, here are some **definitions of essential vocabulary** in this chapter:

- **Event** - something that happens that affects a user of a system.
- **Trigger** - a type of event. It's a small event that provokes a behavior.
- **Behavior** - a reaction to an event.

How do we know which behaviors to add?

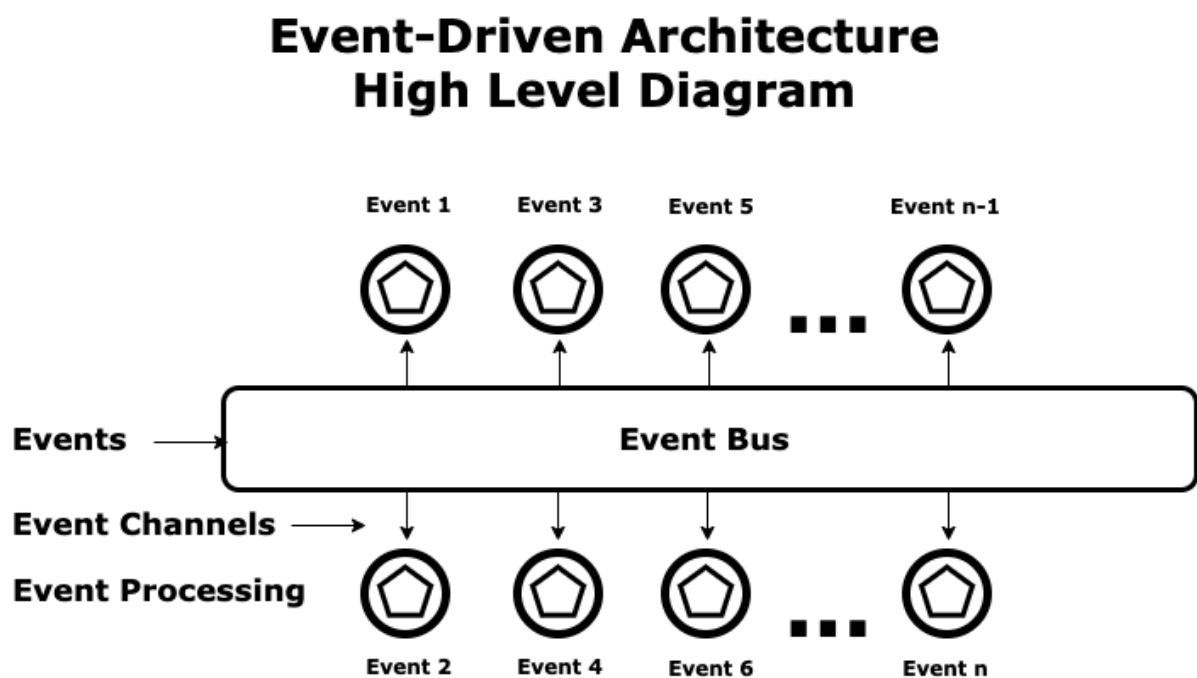
Well, clients usually specify what kind of events they want to be informed about, and what they want the system to do in each case. For example:

- If the price of stock X reaches \$40 (trigger), please send me an email (behavior).
- If a salesperson in our organization enters a purchase order (trigger), then display a message on the initial screen of the deposit manager when he or she logs into the system (behavior).
- If a customer buys a ticket on our airline for a certain flight and the payment gets approved (trigger), then mark the reserved seat in the corresponding flight as “occupied” immediately (behavior).

Let's check out how to put this in place.

What Is the Structure of the Event-Driven Architecture? ✓

Let's see what this looks like:



Event-driven architecture

As you can see in the diagram above, a standard event-driven architecture has at least these components:

- **Event bus:** This is a communication line that links all users to events. This line isn't a physical cable but refers to a logical connection, similar to the one between your laptop and

a website you are visiting. Events are carried through this abstract event bus and reach all users of the system. The bus doesn't actually exist, but it is used to describe the interchange of data between many components or users.

- **Event channels:** An event channel is a label for the kind of events that users subscribe to. For example, the channel "sales" would publish all events in the sales area to all subscribed users. On Twitter, a hashtag like #awesomejapan is a channel that brings all Tweets published by other users containing this same hashtag. If a user is subscribed to a certain channel, he or she will get all related messages.
- **Event processing:** All actions taken *after* a certain event are executed in the event processing module. This module acts upon a certain event to serve the user. For instance: "If I get this kind of message, do this," i.e., "If I get a new purchase message, then bill the customer."

How does this work in practice? Let's revisit our earlier examples and examine how an event-driven architecture supports their functionality:

- *If the price of stock X reaches \$40, please send me an email.* How does this happen? The user has a list of stocks to be notified about. The user picks stock X and selects a condition (greater than \$40). The user selects an option from a menu of possible ones like, "Send me an email," "Alert me when I log into the system," "Send me an SMS," etc. When the corresponding event happens, it is published in the event bus, the system reacts to it, and subscribed users get a message.
- *If a salesperson in our organization enters a purchase order, then display a message in the initial screen of the deposit manager when he or she logs into the system.* How does this happen? There is a pre-existing channel in the system called "Purchase_Orders." Users who need to know that a new purchase has occurred (stock manager, marketing manager, logistics manager, etc.) go to a "Subscribe to channels" option in a menu and see all possible channels predefined by the system. They select the channel to subscribe. When the corresponding event happens, it is published in the event bus, and subscribed users get a message.
- *If a customer buys a ticket on our airline for a particular flight and the payment gets approved, then mark the reserved seat on the corresponding flight as "occupied" immediately.* How does this happen? There is a pre-existing channel in the system called "New_Ticket." Users who need to know that a new ticket was purchased (operations manager, ground support staff, etc.) go to a "Subscribe to channels" option in a menu and they see all possible channels predefined by the system. They select the channel to subscribe. When a ticket is sold, it is published in the event bus, and the system reacts to it.

There are several **advantages** to using event-driven architecture:

- If you have users who need to listen (subscribe to) to different messages on a specific topic, the event-driven architecture is the right approach. This happens when the system must react to events with no predictive behavior: for example, if I subscribe to the "News from

Africa” channel, I have no control over what news I will get or how much. The system is not predictive; it reacts to events generated by other users.

- If these users are outside the organization, events can be carried in a public event bus and reach all users immediately.
- Many events can be handled at once, even millions of them.

There are also a few key **disadvantages**:

- The event bus can get overloaded.
- If the event bus fails, the whole system fails.
- There is no control of event flow: many events can happen at once, creating chaos for users.

When Would I Use the Event-Driven Architecture?

▼

Here are some situations that this would be useful for:

Example Name	Definition	Real-World Example	Advantages
Micro blogging system	A system that publishes short messages and allows users to receive them in their feed according to channels or topics.	<ul style="list-style-type: none">• Twitter.com.	<p>There can be millions of channels since they are defined by users.</p> <p>Users subscribe or unsubscribe to channels as they please.</p>
Provisioning system	A provisioning system is an application in a mobile telephone company that creates a phone line when the customer buys one, defines the corresponding billing procedures for the customer, and ensures that the device reaches the customer.	<ul style="list-style-type: none">• Oracle Business Suite (oracle.com).• Amdocs Provisioning System (Amdocs.com).	<p>Mobile operators have thousands of POS (points of sale) in shopping malls, in the street, and other public places.</p> <p>When a line is sold to a customer, an event is published in the event bus. Many applications in the company's internal systems are subscribed to this kind of event</p>

			(channel) and act accordingly.
Plant automation system	<p>A plant automation system is a software system that controls production in a plant, usually in a robot environment.</p> <p>This system usually acts upon events that occur in the production process, for example, a certain engine that surpasses a temperature threshold.</p>	<ul style="list-style-type: none"> • Siemens STEP7 (siemens.com). • Rockwell Factory Talk (rockwell.com). 	<p>If no error occurs, production in the plant goes uninterrupted. If an error, exception, or warning is detected by the control system, an event is published in the event bus and many control modules may react to it.</p>

Case-Study: Solving a Business Problem With Event-Driven Architecture



Let's analyze a real case where event-driven architecture was used.

RNC is a news agency. Here are some **quick facts**:

- RNC employs more than 12,000 journalists that produce news, articles, and information sold as subscription plans to companies and individuals all over the world.
- RNC is a truly global company: it can reach any country or region in the world within hours if news coverage is required.
- News can be divided into categories. The company is implementing a content management system to improve category management. Clients subscribe and pay for these news categories, and receive a constant stream of updates over the internet.
- Categories may be by region: a specific country or city, a continent, a group of countries.
- They can be by topic, like science news, and they can be divided by subtopics: science-medicine, science-physics, etc.
- Customers sign a 12-month contract to receive news from a particular category in real-time.

What's the Business Problem?

RNC needs to improve the way news is delivered to its clients all over the world. There are two main issues with the existing system:

1. They produce thousands of news items every hour. The current system is slow to process the news topics, which delays reaching clients for about five minutes.

2. Each journalist must tag the piece of news produced with a group of categories to which it is related. The rule for adding tags is not clear nor enforced. Therefore, many news items reach the system with only one category or none at all. The news piece is essentially wasted, because, without a category, the user won't be able to consume it.

We need to use an architecture that will support these business needs.

Got it! Shall we start with the facts?

Exactly! The quantity of data is overwhelming, which is the real problem:

- News categories are vast, varied, and likely to change over time.
- There are more than 12,000 journalists who produce (probably) multiple news stories.
- There are more than 120,000 customers subscribed to a paid (monthly) news plan.

We have three major players:

1. Categories.
2. Journalists producing news.
3. Customers who want to be notified about news.

An event-driven architecture is perfect for this situation because it would naturally **link these three players through event channels and subscriptions**, instead of having to foresee all possible combinations (millions!).

In an event system like that, all **categories will be predefined**, and **no article can be written without tags**. This reduces the amount of data that needs to be managed (because there aren't infinite categories) and is easier to code because the scope is limited. Now, all the journalist has to do is tag the piece of news with at least one category, and this piece of news will get to all users subscribed.

Specifically, how would an event-driven architecture work?

Journalists generate events ("I've written a piece of news in the Science-Medicine category") and send them over a channel (Science-Medicine). Customers sign service contracts to be subscribed to a channel. A customer can then see a channel feed in their browser with news about the categories they've subscribed to in real-time.

The real improvement for RNC would be to produce and sell the news in the same way: associating them to a channel.

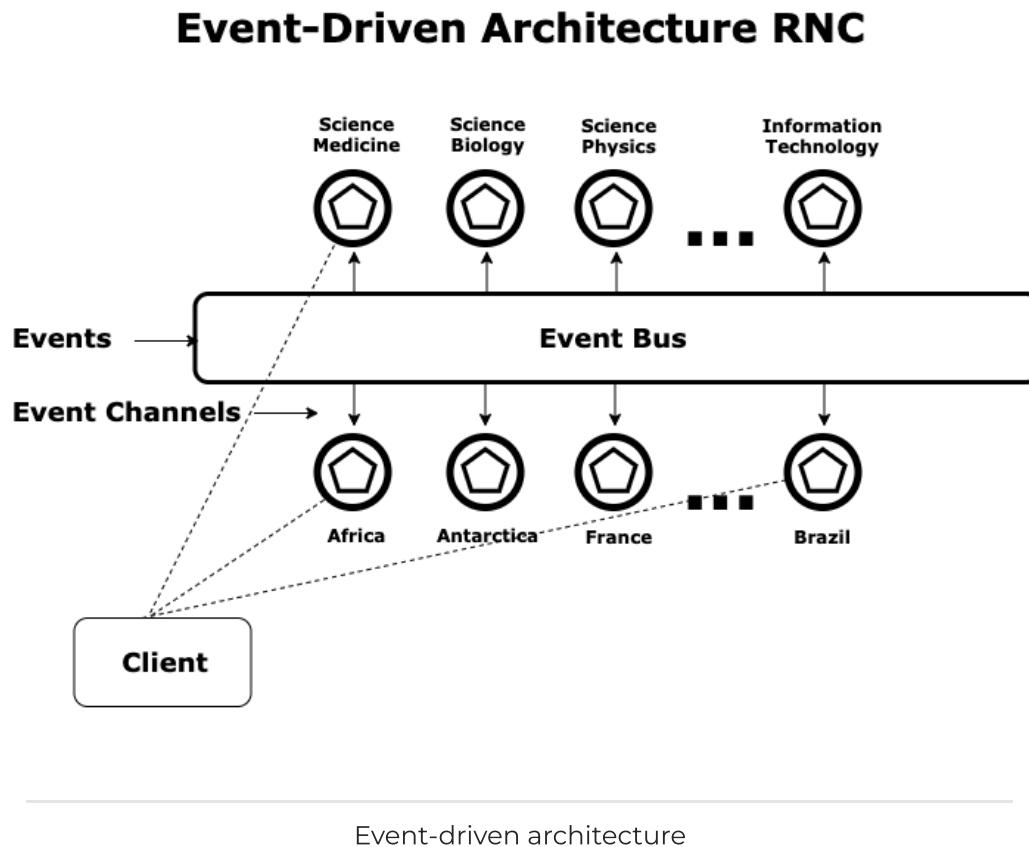
So to sum up:

- We know that we have journalists, news, and customers.

- We know that some news is not categorized correctly.
- We know that news get delayed getting to client feeds.

What's the Solution?

Now that we've walked through the thought process, let's see the detailed architecture diagram:



Let's explain each component of the content management system:

- **Event bus:** This is a communication line that links all users to events. Events are carried through the event bus and reach all users of the system. Each new piece of news is an event; the event bus carries this piece of news and delivers it to users who have subscribed to a topic (channel).
- **Event channels:** An event channel is a label for some news categories that users subscribe to. For example, a user wants to receive all news from South Africa, he or she subscribes to the channel "South Africa." Each piece of news tagged as "South Africa" by the journalist that wrote it gets to this user.
- **Clients:** Users who subscribe to channels and receive the related messages.

Try it out for yourself!

Now that you've seen one solution, see if you can apply what you've learned to another!

Context: You work for a major stockbroker. You are a software architect, and your boss is Linda, the company's chief information officer (CIO).

Your Mission: You have been asked to develop a software system for managing alerts on stock prices. These alerts are sent to customers.

Here are some **key questions** you should ask yourself:

1. Customers define alerts they want to have, like “If stock X surpasses \$40, please send me an email.” How could this be done efficiently?
2. There are thousands of company stocks trading in the market. How can we manage such a high number of possible stock alerts?

Can you produce an architecture diagram for this business setting?

Once you’ve written out your version, check it against my **solution**.

Let’s Recap!



Architecture Model	Description	Advantages	Disadvantages	When to use it
Event-driven	Events are produced, transported, and interpreted over an event bus.	Ability to handle millions of events at once.	If the event bus breaks, the system does not work at all.	When you have many events at once.
	Clients subscribe to a group of events (called channel) and act upon receiving the messages.	Ability to communicate different technologies and platforms to the same event bus.	The event bus may get overloaded and suffer performance problems.	When you have to act upon an event in real-time (synchronic execution).
				When you have different platforms.

[< CLIENT-SERVER ARCHITECTURE](#)[SERVICE-ORIENTED ARCHITECTURE >](#)

Teacher

José Esterkin

Software engineer, project manager and trainer. Director of Positive, a project management consulting firm based in Buenos Aires.

OPENCCLASSROOMS

▼

OPPORTUNITIES

▼

SUPPORT

▼

FOR BUSINESS

▼

MORE

▼

 English

▼



