

Event-Driven Architecture

Softwarearchitektur für reaktive und robuste Anwendungen

Anwendungen müssen heutzutage beliebig erweiterbar, einfach änderbar sowie schnell ersetzbar sein. Die Event-Driven Architecture definiert eine flexible Anwendungsstruktur, die die Bedürfnisse abdecken kann. Sie passt sich dabei an die fachlichen Anforderungen an, skaliert mit der Anzahl der Anwender und bleibt auch bei hoher Auslastung stets antwortbereit.

Definition

Event-Driven Architecture ist eine Softwarearchitektur. Sie ist einerseits durch lose gekoppelte Teilsysteme geprägt, die für Flexibilität und Skalierbarkeit sorgen. Damit agieren die Teilsysteme parallel und unabhängig voneinander. Andererseits bildet sie die Grundlage für eine fachlich motivierte, asynchrone Kommunikation. Tritt in einem Teilsystem bei der Verarbeitung ein bestimmtes Ereignis (engl. event) auf, so berichtet der Erzeuger (engl. publisher) dieses Ereignisses an eine zentrale Instanz, die Middleware. Ist ein anderes Teilsystem an einem bestimmten Ereignis interessiert, dann abonniert (engl. subscribe) es diesen Ereignistyp. Wichtig: Teilsysteme können sowohl Erzeuger als auch Abonnent sein.

Events besitzen eine Zustellgarantie. Die Zustellgarantie sagt aus, dass ein Event maximal einmal, mindestens einmal oder

exakt einmal zugestellt wird. Kommt die Middleware mit der Zustellung der Events nicht hinterher, sammelt sie eingehende Events in einer Ereigniswarteschlange (eng. event queue). Publisher können ihr Event kommunizieren, weiterarbeiten und sind nicht blockiert. Die Middleware speichert wiederum in ihrem Ereignisprotokoll (engl. event log) die verarbeiteten Events.

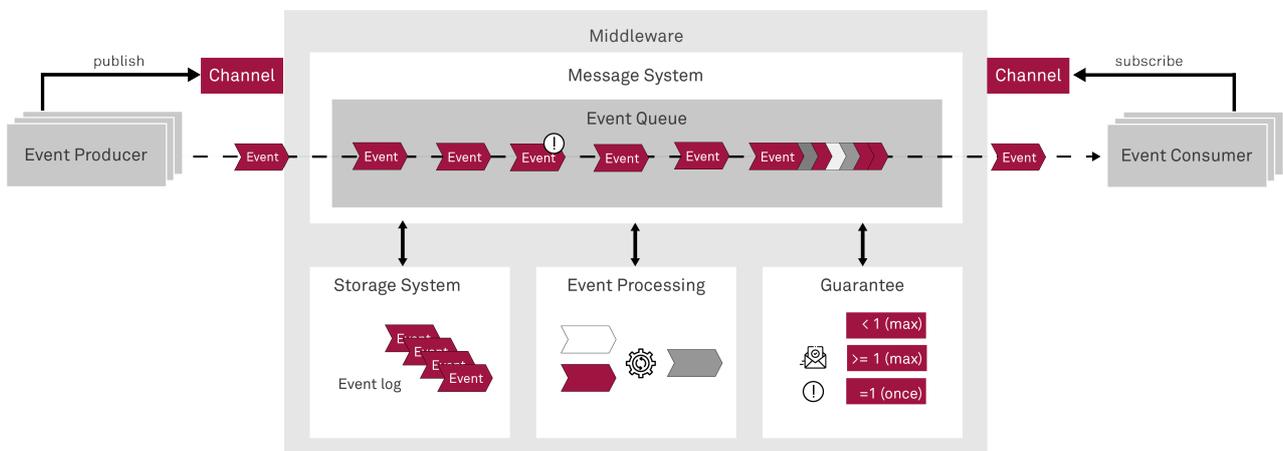
In verschiedenen Fällen langt es nicht aus, Events nur weiterzureichen. Dann analysieren Ereignisprozessoren (engl. event processor) der Middleware die eingehenden Events, gruppieren und verarbeiten sie, um anschließend neue Events zu generieren.

Event-Driven Architecture ist hoch generisch, eignet sich insbesondere für verteilte Systeme und lässt sich auf unterschiedlichen Architekturebenen einsetzen.

Sie ist etwa auf unterster Architekturebene bei UI-Frameworks nicht mehr wegzudenken, denn sie garantiert verzögerungsfreie Benutzerinteraktionen. Selbst auf der Ebene der Applikationslandschaft bleibt das System antwortbereit, elastisch, nachrichtenorientiert und widerstandsfähig.

Referenzszenario

Im Zuge der Anforderungsanalyse haben sich einige grundlegende Rahmenbedingungen für die zu implementierende Anwendung ergeben. Diese Anforderungen verlangen ein robustes System. Es soll unter allen Umständen zeitgerecht antworten. Zudem ist absehbar und erforderlich, dass die Anwendungslandschaft mit der Zeit wachsen wird und stetig neue Anforderungen erfüllen muss. Der Softwarearchitekt entscheidet sich deshalb für eine Event-Driven Architecture.



Verteilte Systeme

- Service-Oriented Architecture
- Microservices
- Serverless Computing

Architekturebenen

- Modul
- Applikation
- Applikationslandschaft

EDA

Facharchitektur

- Event Storming
- Prozesse und Events
- Case Management
- Business Process Management

Reaktiv

- widerstandsfähig
- elastisch
- nachrichtenorientiert
- antwortbereit

Ein Vorteil ist zudem, dass einige Nachbarsysteme bereits auf eine Event-Driven Architecture setzen und untereinander mit Events kommunizieren.

überarbeitete, im besten Fall sogar gänzlich neue Funktionen bereit.

Potenzial

Aus Unternehmenssicht verkürzt die Event-Driven Architecture dank ihrer Flexibilität die Auslieferzeit (engl. time-to-market) neuer Funktionen und Anwendungen. Ihre Skalierbarkeit hält die variablen Kosten in einem angemessenen Rahmen und kommt gut mit steigender Nachfrage seitens der Anwender zurecht. Das alles wirkt sich positiv auf die Anwenderzufriedenheit aus.

Reifegrad

Seit dem Jahr 2000 hat Event-Driven Architecture mehrfach bewiesen, dass sie sich an immer neue Anforderungen anpassen und mit der schnellen technologischen Entwicklung mithalten kann. Entsprechend viele Produkte konnten sich auf dem Markt platzieren und ihre Produktivität beweisen. Typische Problemfälle und Lösungsmaßnahmen sind bekannt. Es gibt umfangreiche kommerzielle Unterstützung als auch eine große Open-Source-Community.

Alternativen

Ein Ersatz im Rahmen reaktiver Architekturen ist Message-Driven Architecture. Message-Driven Architectures verschickt hingegen mehr oder weniger eindeutige Nachrichten an klar adressierbare Empfänger. Tritt bei der Verarbeitung der Nachrichten ein Fehler auf, dann kann die Anwendung den ordnungsgemäßen Zustand direkt wiederherstellen. Allerdings ist die Wartung der Software aufwendiger.

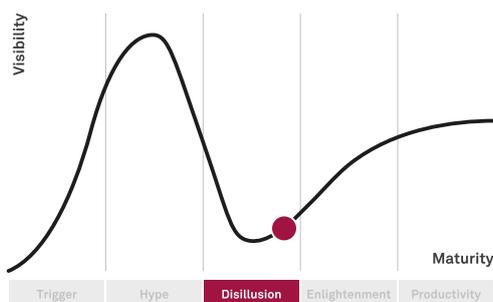
Zusätzlich profitieren die Anwender davon, dass alle ihre Eingaben flüssig und insbesondere zuverlässig verarbeitet werden. Diese Garantie lässt sich unabhängig von der Auslastung der Anwendung geben. Dank der hohen Flexibilität der Event-Driven Architecture stehen den Anwendern in der Regel auch zeitnah

Marktübersicht

Eine Event-Driven Architecture ist nicht an eine bestimmte Programmiersprache gebunden. Sie lässt sich deshalb sowohl für Bestandssysteme als auch auf völlig neuen Plattformen umsetzen. Auf Basis des Standards #CloudEvents können Publisher ihre Events zudem einheitlich

Fazit

- + ist skalierbar, robust, erweiterbar
- + ermöglicht parallele Bearbeitung
- + nutzt fachliche Ereignisse statt technischer Abläufe
- + erlaubt wiederholbare und somit korrigierbare Zustandsänderungen
- ist undurchsichtig im Ablauf
- erfordert eine hohe Investition, wenn die Ereignissequenz wichtig ist
- bietet keinen direkten Rückkanal in der Kommunikation



Buzzword Factor (Ent./Customer)

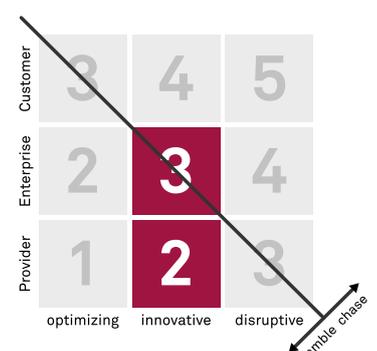
1 low	2 medium	3 high
----------	-------------	-----------

Entry Barrier (Provider)

1 low	2 medium	3 high
----------	-------------	-----------

Benefit Level (Provider)

1 low	2 medium	3 high
----------	-------------	-----------



<https://msg.direct/techrefresh>

Stand: September 2020

msg systems ag

Robert-Bürkle-Straße 1 | 85737 Ismaning/München | Telefon: +49 89 96101-0 | Fax: +49 89 96101-1113 | www.msg.group | info@msg.group