Complex Event Processing im Überblick 2

Complex Event Processing ist eine von David Luckham entwickelte Softwaretechnologie, um kontinuierliche Ereignisströme zu verarbeiten [17]. Dieses Kapitel stellt die Grundkonzepte des Complex Event Processing schrittweise im Überblick vor.

2.1 Charakterisierung von Ereignissen

Complex Event Processing rückt *Ereignisse* in den Fokus der Softwarearchitektur. Genauso vielfältig wie die vielen verschiedenen Anwendungsdomänen sind auch die unterschiedlichen *Typen* von möglichen Ereignissen.

> **Ereignis (event):** Ein *Ereignis* kann alles sein, was passiert oder von dem erwartet wird, dass es passiert [18]: eine Aktivität, ein Vorgang, eine Entscheidung etc. Im Allgemeinen bezieht sich ein Ereignis auf die Veränderung eines *Zustands*, also typischerweise auf die Änderung des Wertes einer Eigenschaft eines realen oder virtuellen Objekts.

Beispiele für Ereignisse sind *technische Ereignisse*, wie z. B. die Veränderung der gemessenen Temperatur an dem Sensor einer Fertigungsmaschine oder die Geschwindigkeit eines vorbeifahrenden Fahrzeugs an einem Messpunkt, oder auch *Geschäftsereignisse*, wie z. B. die Kündigung eines Liefervertrags oder die Unterschreitung des erforderlichen Lagerbestandes einer Ware.

Um ein Ereignis verstehen, einordnen und weiterverarbeiten zu können, muss es *sämtliche Informationen*, die für die weiteren Verarbeitungsvorgänge erforderlich

R. Bruns, J. Dunkel, *Complex Event Processing,* essentials,
DOI 10.1007/978-3-658-09899-5_2

sind, als Ereignisdaten mit sich führen. Die *Ereignisdaten* beschreiben den *Kontext*, in dem das Ereignis aufgetreten ist, und bestehen aus *allgemeinen Metadaten* und *spezifischen Kontextdaten*.

- *Metadaten*: Jedes Ereignis muss allgemeine, *domänenunabhängige Metadaten* beinhalten. Ein Ereignis benötigt eine eindeutige *Ereignis-ID*, um es von anderen Ereignissen zweifelsfrei unterscheiden zu können, und einen *Zeitstempel* (timestamp), der den Zeitpunkt des Auftretens des Ereignisses repräsentiert. Weiterhin ist es in vielen Fällen wichtig, dass ein Ereignis auch Informationen über die Ereignis*quelle* und den Ereignis*typ* enthält.
- *Ereigniskontext*: Darüber hinaus muss ein Ereignis natürlich auch noch den eingetretenen Sachverhalt beschreiben, also die eigentlichen *Nutzdaten* (payload) mit sich führen. Beispielsweise enthält ein Ereignis, das durch einen Feuchtigkeitssensor ausgelöst wurde, den Wert der gemessenen Luftfeuchtigkeit.

Ein *Ereignisstrom* (event stream) ist eine linear *geordnete* Sequenz von *kontinuierlich* eintreffenden Ereignissen [18].

2.2 Grundbegriffe des Complex Event Processing

Complex Event Processing ermöglicht es, sehr große Mengen von Ereignissen dynamisch zu verarbeiten, den fachlichen Informationswert aus diesen Ereignissen zu extrahieren und für die Steuerung des Kontrollflusses von Anwendungen zu nutzen.

Complex Event Processing (CEP): *Complex Event Processing* ist eine Softwaretechnologie für die *dynamische* Analyse von *massiven Daten- bzw. Ereignisströmen* in *Echtzeit*. Mit CEP ist es möglich, *kausale*, *temporale*, *räumliche* und andere Beziehungen zwischen Ereignissen auszudrücken. Diese Beziehungen spezifizieren *Muster*, nach denen die Ereignismenge durchsucht wird (*event pattern matching*) [17].

Ereignismuster

Um Zusammenhänge zwischen Ereignissen in einem massiven Ereignisstrom zu erkennen, muss der Ereignisstrom auf das Auftreten von spezifischen *Ereignismustern* hin untersucht werden. Die Suche nach Mustern betrachtet einen Ereignisstrom über einen längeren Zeitraum. Je nach Mächtigkeit der Operatoren lassen sich unterschiedliche Arten der Mustererkennung differenzieren.

- *Erkennen einfacher Ereignismuster*
  Die einfache Mustererkennung identifiziert *Einzelereignisse* oder *boolesche Kombinationen* von Ereignissen in einer Ereignismenge, beispielsweise Ereignisse bzw. Ereigniskombinationen, die auf eine problematische Situation hindeuten. Zum Beispiel beschreibt das einfache Muster $(A\ and\ B\ and\ not(C))$ das Auftreten von zwei Ereignissen A und B, jedoch bei Abwesenheit von Ereignis C.
- *Erkennen komplexer Ereignismuster*
  Oft reichen jedoch einfache Muster nicht aus. In solchen Fällen sind weitere Operatoren für die Spezifikation von komplexeren Mustern erforderlich. Von besonderer Bedeutung sind dabei Operatoren, um die *Reihenfolge* von Ereignissen zu spezifizieren oder um *Zeitfenster* für das Auftreten für Ereignisse festzulegen.
- *Abstraktion von Ereignismustern*
  Wird in einer Folge *einfacher* Ereignisse ein Muster erkannt, so kann ein *komplexes* Ereignis auf einer *höheren Abstraktionsebene* erzeugt werden, das somit das erkannte Muster repräsentiert. Diese Abstraktion von Ereignismustern liefert verständliche Sichten auf komplexe Situationen und reduziert die Anzahl der relevanten Ereignisse.
  Typische Operationen zur Abstraktion von Ereignismustern sind das Bilden von Summen oder Durchschnittswerten. Zum Beispiel kann in einem System zum Wertpapierhandel für eine Aktie der durchschnittliche Kurswert über die vergangenen 200 Tage berechnet werden. Dieser ‚200-Tage'-Wert kann in der Chartanalyse von Aktien weiterverwendet werden.

Ereignisverarbeitung mit Regeln

In CEP ist das Wissen über die Ereignisverarbeitung *explizit* und *deklarativ* in Form von *Regeln* repräsentiert.

Ereignisregel: Eine *Ereignisregel* (event processing rule) beschreibt eine spezifische Reaktion, die bei Erkennen eines Musters erfolgen soll. Jede Regel setzt sich aus einem Bedingungs- und einem Aktionsteil zusammen:

- Der *Bedingungsteil* besteht aus einem oder mehreren miteinander verknüpften Mustern, mit denen der Ereignisstrom abgeglichen wird. Wenn im Ereignisstrom das Muster vorkommt, dann ist der Bedingungsteil erfüllt, d. h. die Regel *matched*, und der *Aktionsteil* der Regel wird ausgeführt, d. h. die Regel *feuert*.
- Der *Aktionsteil* legt eine *Reaktion* oder *Aktion* (action) fest, die ausgeführt wird, sobald das Muster im Ereignisstrom vorkommt. Mögliche Aktionen sind das Erzeugen eines neuen Ereignisses oder das Anstoßen eines Dienstes in einem Anwendungssystem.

Da menschliche Fachexperten ihr Wissen oft in Form von Regeln formulieren, bilden Regeln eine adäquate Form der Repräsentation von Fachwissen. Die Regeln können beispielsweise ein Ereignisformat in ein anderes Format *transformieren*, Ereignisse *ausfiltern*, die bei der weiteren Verarbeitung nicht mehr benötigt werden, mehrere einfache Ereignisse zu einem komplexeren Ereignis *aggregieren* und ein neues abgeleitetes Ereignisobjekt *generieren*. Die Verarbeitung der Regeln erfolgt *zentral* in der CEP-Komponente.

Die Formulierung von Regeln erfolgt mit speziellen Regelsprachen, den sogenannten *Event Processing Languages* (EPLs) [18]. Mithilfe von EPLs lassen sich Ereignismuster und -reaktionen *deklarativ*, also auf einer hohen Abstraktionsebene beschreiben. Statt selber die einzelnen Schritte eines Algorithmus zur Mustererkennung zu implementieren, müssen lediglich die gesuchten Muster in einer dafür speziell vorgesehenen Deklarationssprache definiert bzw. modelliert werden.

Ein Beispiel für eine EPL ist die Sprache RAPIDE-EPL von Luckham [17]. In RAPIDE-EPL können Ereignismuster unter anderem durch die Operatoren *and*, *or* und $\rightarrow$ wie folgt definiert werden:

- *when* (A *and* B *and* C) *then* action_X
  Die Regel feuert, wenn die Ereignisse A, B und C im Ereignisstrom aufgetreten sind.
- *when* (A $\rightarrow$ (B *or* C)) *then* action_Y
  Die Regel feuert, wenn zuerst das Ereignis A und danach das Ereignis B oder das Ereignis C aufgetreten sind.

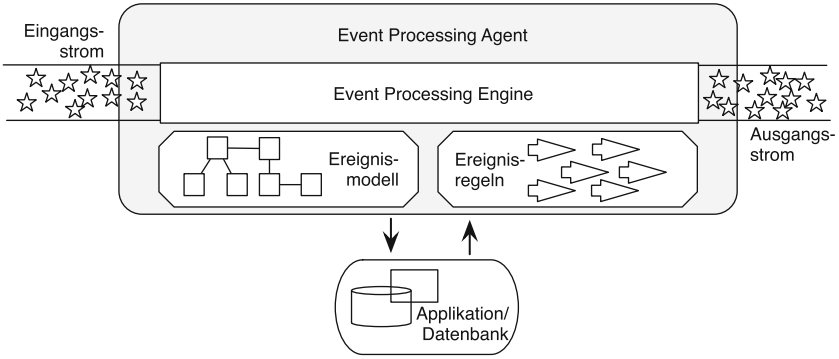

Abb. 2.1 Elemente eines Event Processing Agent

Derzeit existiert kein etablierter Standard für EPLs. Einen kurzen Überblick über in der Praxis eingesetzte CEP-Produkte gibt das Kap. 5. In Abschn. 3.3 wird exemplarisch die EPL des Open-Source-Systems *Esper* vorgestellt.

2.3 Event Processing Agents

Die Kernkomponente für die Ereignisverarbeitung in einem ereignisgesteuerten System ist ein *Event Processing Agent* [17].

> **Event Processing Agent (EPA)**: Ein *Event Processing Agent* ist ein Softwaremodul, das *komplexe* Ereignisse verarbeiten und Ereignis*muster* in einem Strom von Ereignissen erkennen kann [18].

Die Abb. 2.1 visualisiert einen Event Processing Agent mit seinen grundlegenden Elementen:

- *Ereignismodell*
  Nur mit einem exakt festgelegten *Ereignismodell* (event model) ist die Verarbeitung von Ereignisströmen möglich. Das Ereignismodel spezifiziert die erlaubten *Typen* von Ereignissen, die zu jedem Ereignistyp gehörenden Datenattribute sowie die Beziehungen und Abhängigkeiten zwischen den Typen.

- *Ereignisregeln*
  Auf der Basis des Ereignismodells werden *Ereignisregeln* zur Verarbeitung der Ereignisse mithilfe einer EPL definiert. Reichen die Daten in den Ereignissen für die Verarbeitung nicht aus, so müssen diese durch *Kontextwissen* aus Anwendungssystemen oder Datenbanken angereichert werden.
- *Event Processing Engine*
  Die eigentliche Musterkennung in den Ereignisströmen wird von der *Event Processing Engine* durchgeführt. Die Event Processing Engine ist ein spezieller *Regelinterpreter*, der kontinuierlich den Strom der Ereignisse mit den Mustern in den Bedingungsteilen der Regelmenge abgleicht und, sobald ein Muster in der Ereignismenge auftritt, die entsprechende Ereignisregel ausführt.
  Da für die *Mustererkennung* auch in der Vergangenheit aufgetretene Ereignisse relevant sein können, müssen die Ereignisdaten permanent im *Arbeitsspeicher* vorgehalten werden (*in-memory computing*). Typischerweise ist die Ereignismenge jedoch viel zu groß, um vollständig gespeichert zu werden. Betrachtet werden deshalb Ereignisse in einem gewissen Intervall (Zeit- oder Längenfenster). Neue Ereignisse kommen beim Eintreffen zu dem Intervall hinzu, alte Ereignisse fallen als Folge aus dem Intervall heraus.

Komplexe Verarbeitungsvorgänge lassen sich besser in kleinen Teilschritten mit jeweils *einfachen* Ereignismustern und *wenigen* Regeln durchführen, als die komplette Verarbeitung in einem einzigen, großen Schritt mit *komplexen* Ereignismustern und *vielen* Regeln abzuarbeiten. Um eine schrittweise Verarbeitung zu realisieren, muss jeder Schritt durch einen eigenen EPA realisiert werden. Durch die Verknüpfung mehrerer EPAs ergibt sich ein Netzwerk, in dem die Ausgabe eines EPAs als Eingabe eines anderen dient.

Event Processing Network (EPN): Ein *Event Processing Network* ist eine Menge von Event Processing Agents, die durch sogenannte *Ereigniskanäle* miteinander verbunden sind [18], über die sie während der Verarbeitung Ereignisse austauschen.

Ein einzelner EPA übernimmt hierbei nur einen oder wenige abgegrenzte Schritte im Prozess der Ereignisverarbeitung, zum Beispiel das Herausfiltern von doppelten Ereignissen oder das Aggregieren von zusammenhängenden Ereignissen zu einem komplexen Ereignis. Aufgrund der eingeschränkten Regelmenge kann ein solcher EPA auch als *leichtgewichtig* bezeichnet werden.

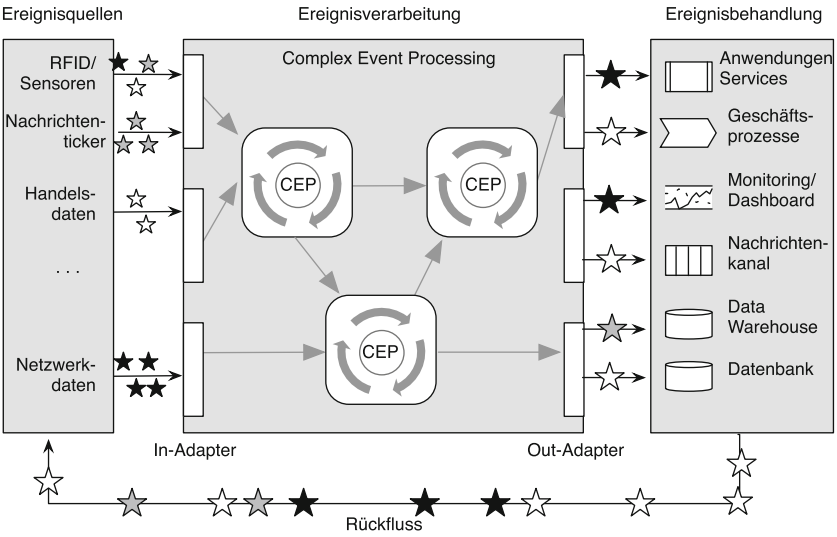

Abb. 2.2 Logische Schichten einer Event-Driven Architecture

2.4 Event-Driven Architecture

Die *Architektur* von Software ist die grundlegende Organisation eines Systems, bestehend aus ihren Komponenten sowie deren Beziehungen und Interaktionen [21]. *Event-Driven Architecture* repräsentiert einen Softwarearchitekturstil, der speziell auf die *Echtzeitverarbeitung* von massiven Ereignisströmen ausgerichtet ist. Dabei kommt Complex Event Processing als Kerntechnologie zum Einsatz.

> **Event-Driven Architecture (EDA)**: Event-Driven Architecture ist ein *Architekturstil*, in dem einige Komponenten *ereignisgesteuert* sind und die Interaktion durch den *Austausch von Ereignissen* erfolgt [18].

Abbildung 2.2 zeigt die grundlegenden *logischen Strukturierungsschichten* und Bestandteile einer Event-Driven Architecture [8, 11]:

1. *EDA-Schicht: Ereignisquellen*
   Eine *Ereignisquelle* (event source) ist eine beliebige Entität, die Ereignisse *erkennt*, entsprechende Ereignis*objekte* generiert und bekannt gibt [18]. Beispiele für Ereignisquellen sind Sensoren, RFID-Lesegeräte, Datenticker bzw. -Feeds, Softwaremodule, Web Services, Benutzerinteraktionen, E-Mail, interne Uhren und viele mehr. Auch ereignisverarbeitende Komponenten (beispielsweise EPAs) können wiederum Ereignisse generieren und somit als Quelle fungieren.
2. *EDA-Schicht: Ereignisverarbeitung*
   Die *Ereignisverarbeitung* (event processing) nimmt Ströme von Ereignisdaten aus den verschiedenen Ereignisquellen entgegen. Die Analyse und Verarbeitung der Ereignisse erfolgt durch CEP-Regeln, die in einem aus mehreren EPAs bestehenden EPN ausgeführt werden, wie in den Abschn. 2.2 und 2.3 beschrieben.
3. *EDA-Schicht: Ereignisbehandlung*
   Die eigentliche Ereignisbehandlung (event handling) als Reaktion auf Ereignismuster erfolgt in den produktiven Unternehmensanwendungen, den nachgelagerten Backend-Systemen. Typische Reaktionen zur Ereignisbehandlung sind beispielsweise:
   - *Aufruf von Diensten*: Aufruf eines Dienstes, z. B. eines Web Service oder eines anderen API, um eine fachliche Anwendungsfunktion auszuführen oder einen Geschäftsprozess zu initiieren.
   - *Aktualisierung eines Dashboards*: Direkte Benachrichtigung von Benutzern durch Aktualisierung von Kennzahlen, Visualisierung in einer graphischen Benutzungsoberfläche oder in einem Dashboard.
   - *Publizieren einer (Ereignis-)Nachricht*: Einstellen eines Ereignisobjekts in Form einer Nachricht in eine Nachrichtenwarteschlange. Registrierte Ereignisinteressenten werden über die neue Nachricht informiert und können diese zur weiteren Verarbeitung abrufen.
   - *Auslösen einer menschlichen Prozessbehandlung*: In Notfallsituationen ist es wichtig, die beteiligten Personen entlang einer vordefinierten Eskalationshierarchie frühzeitig zu informieren.

Eine Softwarekomponente aus der Ereignisbehandlungsschicht kann natürlich selbst wiederum Ereignisse generieren und damit als Quelle fungieren. Diese Ereignisse fließen erneut in die Verarbeitungsschicht zur weitergehenden Analyse ein. Dieser Rückfluss von Ereignissen ist in Abb. 2.2 durch den untersten Pfeil dargestellt.

Tab. 2.1 CEP-Eigenschaften zur komplexen Analyse von Datenströmen

Eigenschaften von Datenströmen	CEP-Eigenschaften
Aktuelle Live-Daten	Verarbeitung der Daten zum Zeitpunkt des Auftretens
Feingranular	Mustererkennung zur *Korrelation* und *Abstraktion* von Ereignissen; Hierarchie von Abstraktionsstufen
Komplexe Abhängigkeiten zwischen Daten	Mächtige EPL-Sprachkonstrukte zur Spezifikation komplexer Analysemuster
Unbegrenzt	Spezielle EPL-Operatoren für Fragmentierung des Datenstroms
Kontinuierlich und *volatil*	*In-memory*-Verarbeitung der Datenströme
Massiv und *hochfrequent*	*Massendatenverarbeitung* und *Echtzeitfähigkeit* von hochspezialisierten CEP-Engines
Implizite Beziehungen	Wissen über Datenanalyse deklarativ in Regeln formuliert

Komplexe Analyse von Datenströmen mit CEP

Die Tab. 2.1 stellt noch einmal die wesentlichen Eigenschaften von Datenströmen und den dazu passenden Lösungskonzepten von CEP gegenüber.

Springer

http://www.springer.com/978-3-658-09898-8

Complex Event Processing
Komplexe Analyse von massiven Datenströmen mit CEP
Bruns, R.; Dunkel, J.
2015, IX, 54 S. 36 Abb., Softcover
ISBN: 978-3-658-09898-8