Table of Contents

# Einleitung

Im Rahmen dieser Hausarbeit wird das Thema (Complex) Event Processing betrachtet. Um ein allgemeines Verständnis für das Thema zu erlangen, werden zunächst die Grundlagen erläutert. In den ersten Kapiteln erfolgt eine Erläuterung der Begriffe Ereignis und Event Processing. In Bezug auf Event Processing wird in Simple Event Processing und Complex Event Processing unterschieden. Da Event Processing immer mehr in den Fokus der Entwicklung rückt, das Datenvolumen immer mehr steigt und diese Daten in Echtzeit verarbeitet werden müssen werden ebenfalls Prozesse und somit auch Ereignisse immer komplexer. In dieser Arbeit wird aus diesem Grund der Bereich Complex Event Processing hervorgehoben und umfassend betrachtet.  
Es wird auf das Thema Ereignismuster im Hinblick auf Complex Event Processing eingegangen und es werden die Anforderungen an entsprechende Ereignisanfragen erörtert. Des Weiteren erfolgt ein Blick auf den Aufbau von Event Processing Networks. Auf Basis der Grundlagen zeigt sich, innerhalb welcher Bereiche Event Processing Anwendung findet. Als Anwendungsbeispiel wird das Tool Riemann herangezogen und beispielhaft eine Konfiguration von Events, Streams und dem Index erläutert.

# Ereignisse

Ereignis (event): „Anything that happens, or is contemplated as happening“ [LuSc11]

Jeder einzelne, genauer betrachtete Datensatz lässt sich als ein eigenständiges Ereignis auffassen, das mit dem Auftreten von bestimmten Geschehnissen in der Anwendungsdomäne korrespondiert. Jedes Ereignis enthält entsprechende relevante Daten zu einem betreffenden Vorkommnis oder Sachverhalt. [BrDu15] Nach [ChSc10] ist ein Ereignis alles, was geschieht oder als geschehen betrachtet wird. Ein Ereignisobjekt ist ein Objekt, das darstellt, kodiert oder aufzeichnet, was geschieht. Eine isolierte Betrachtung eines Ereignisses reicht zumeist nicht aus, um einen Sachverhalt beurteilen zu können. Zunächst müssen die Ereignisse in einen Zusammenhang gestellt werden und es muss eine Betrachtung mehrere Ereignisse über einen längeren Zeitraum erfolgen, um einen entsprechenden Informationsgehalt ableiten und Schlussfolgerungen ziehen zu können. [BrDu15] Um eine Verarbeitung eines Ereignisses zu ermöglichen, muss dieses entsprechende Daten mit sich führen. Ereignisdaten bestehen aus allgemeinen Metadaten und spezifischen Kontextdaten und beschreiben somit den Kontext, in dem das Ereignis aufgetreten ist. Jedes Ereignis muss allgemeine und unabhängige Metadaten beinhalten und benötigt eine eindeutige Ereignis-ID zur Identifikation. Ebenso benötigt ein Ereignis einen Zeitstempel, um den Zeitpunkt des Auftretens zu repräsentieren. Des Weiteren muss ein Ereignis den eingetretenen Sachverhalt beschreiben, es muss Nutzdaten mitführen. Eine linear geordnete Sequenz von kontinuierlich eintreffenden Ereignissen wird als Ereignisstrom (event stream) bezeichnet. [BrDu15]  
Mit Hilfe von Event Processing wird aus den Ereignissen höheres und wertvolleres Wissen in Form von komplexen Ereignissen abgeleitet. Durch Event Processing werden Ereignisse in den Fokus der Softwarearchitektur gesetzt.  
# Event Processing Data Science ist die Verwendung von verschiedenen analytischen Techniken und Methoden, um aus Rohdaten nützliche Erkenntnisse abzuleiten, mit dem Ziel, Prozesse zu optimieren und die Entscheidungsfindung durch verbesserte, auf Daten beruhende Informationen zu unterstützen. [Pier16] Event Processing steht für eine Verarbeitung komplexer Prozesse. Aufgrund der gehäuften Anzahl von Datenquellen ergibt sich ein stetiger Anstieg des Datenvolumens. Für den Umgang mit diesen großen Datenmengen, bedarf es unterschiedlicher Techniken, Methoden und Werkzeuge. Zumeist müssen massive Datenströme immer schneller und vor allem häufig in Echtzeit verarbeitet werden, um geschäftliche Entscheidungen situationsabhängig treffen und Prozesse optimieren zu können. Ein entscheidender Aspekt ist, dass die Verarbeitungsgeschwindigkeit dieser Datenströme mit den Datenmengen die eintreffen, Schritt halten können muss. [BrDu15] Es ergibt sich im Bereich des Event Processing eine Unterscheidung in „Simple Event Processing“ und „Complex Event Processing“. Das Simple Event Processing heißt, dass ein Ereignis nicht mit anderen Ereignissen zusammengefasst wird und somit keine Kombination aus Ereignissen darstellt. Jedes Ereignis wird unabhängig voneinander verarbeitet. Zwar als „Simple“ bezeichnet, können diese Ereignisse einen großen Wert bieten. Diese Ereignisse werden transformiert, was zum Übersetzen, Aufteilen und Zusammensetzen der Ereignisse führt. Ebenfalls umfasst das Simple Event Processing das Ändern eines Ereignisschemas von einer Form zu einer anderen. [IBM10] Complex Event Processing hingegen umfasst die Zusammenfassung von mehreren komplexen Ereignissen zu neuen komplexen Ereignissen [IBM10]. Komplexe Ereignisse sind Situationen, die sich nur durch eine Kombination mehrerer Ereignisse erkennen lassen. Complex Event Processing lässt sich in die zwei Arten „Detection oriented“ und „Computation oriented“ unterteilen. Es wird von „Detection oriented“ gesprochen, wenn das Ziel des Verfahrens ist, aus einem kontinuierlichen Strom von Ereignissen das Eintreten eines komplexen Ereignisses, das vorher bereits als Muster bekannt ist, festzustellen. Um derartige, für komplexe Ereignisse stehende Muster spezifizieren zu können, werden Ereignisanfragesprachen benötigt (Siehe Kapitel Ereignisanfragesprachen). Hingegen ist das Ziel des Verfahrens „Computation oriented“, Muster aus dem kontinuierlichen Strom von Ereignissen zu erkennen und somit neue komplexe Ereignisse zu extrahieren. [BrDu15]

# Ereignismuster

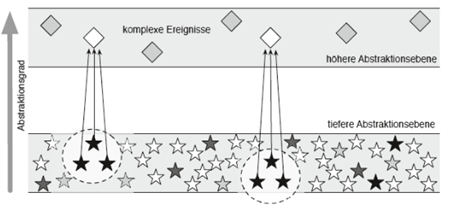
Nach [BrDu15] ergibt sich ein Informationsgehalt eines Ereignisstroms nur durch das Erkennen von Zusammenhängen, Abhängigkeiten und Korrelationen zwischen mehreren Ereignissen. Die zwischen den zusammenhängenden Ereignissen bestehenden Beziehungen, stellen Muster (Event Pattern) dar, die es gilt zu erkennen. Pattern Matching beschreibt das Ziel der Verarbeitung von Ereignisströmen. Es erfolgt eine Unterscheidung in „low level events“ und „high level events“. Low level events sind die direkt aus den Datenquellen stammenden Ereignisse, die ein konkretes Vorkommnis in der realen Welt repräsentieren. Low level events befinden sich auf einer niedrigen Abstraktionsebene. Neue, höherwertige komplexe Ereignisse (high level events) repräsentieren ein erkanntes Muster. High level events befinden sich auf einer höheren Abstraktionsebene, da sie aus der Korrelation von einfachen Ereignissen entstehen (siehe Abbildung1). 

Abbildung 1: Ereignisse auf unterschiedlichen Abstraktionsebenen [BrDu15]

Nach [BrDu15] lassen sich drei Arten der Mustererkennung differenzieren. Eine einfache Mustererkennung identifiziert Einzelereignisse oder boolesche Kombinationen von Ereignissen in einer Ereignismenge. Beispielsweise beschreibt das Muster (A˄B˄⌐C), dass zwei Ereignisse A und B auftreten, bei Abwesenheit des Ereignisses C. Falls weitere Operatoren für die Spezifikation von komplexeren Mustern nötig sind, ist die Sprache von komplexer Mustererkennung. Besondere Bedeutung haben dabei Operatoren zur Spezifikation der Reihenfolge von Ereignissen oder zur Festlegung von Zeitfenstern für das Auftreten der Ereignisse. Ein dritter Aspekt ist die Abstraktion von Ereignismustern, da bei der Erkennung eines Musters in einer Folge einfacher Ereignisse somit ein komplexeres Ereignis auf einer höheren Abstraktionsebene erzeugt werden kann, das das erkannte Muster repräsentiert. Durch eine derartige Abstraktion kann die Anzahl der relevanten Ereignisse reduziert werden und eine verständlichere Sicht auf komplexe Situationen ermöglicht werden. [BrDu15]

# Ereignisanfragen

Für die Verarbeitung von Ereignissen kommen Ereignisanfragesprachen (Event Pattern Language - EPL) zum Einsatz, mithilfe derer sich komplexe Bedingungen definieren lassen, die die Beziehungen zwischen Ereignissen darstellen [VaHo13]. Ereignisanfragen werden kontinuierlich ausgewertet während Ereignisse passieren. Zwar arbeiten Datenbanken ebenso oft mit ereignisbezogenen Daten, so sind diese jedoch einmalig und gegen eine endliche Datenmenge. Nach [EcBr09] lassen sich Anforderungen an eine Ereignisanfragesprache durch die im Folgenden genannten vier Aspekte beschreiben:

**Extraktion von Daten** Ereignisse enthalten Daten. Diese Daten sind relevant dafür, ob und wie reagiert wird und müssen für Bedingungen in Anfragen, für mögliche Reaktionen, zur Anreicherungen mit anderen Daten (z.B. Datenbanktabellen) oder zur Konstruktion neuer Ereignisse zur Verfügung stehen.

**Komposition** Es muss ermöglicht werden, mehrere einzelne Ereignisse verbinden zu können, damit ihr gemeinsames, über die Zeit verteiltes Auftreten ein komplexes Ereignis ergibt. Eine Komposition muss meist datenbezogen sein.

**Zeitliche Zusammenhänge** Ereignisanfragen enthalten meist zeitliche Bedingungen, die ausdrücken, dass Ereignisse in einer gewissen Zeitspanne oder Reihenfolge passieren müssen.

**Akkumulation**  
Sowohl Anfragen mit Negation (Fehlen eines Ereignisses), als auch die Aggregation von Ereignisdaten (eine Menge von Ereignissen wird kompakt auf ein einziges Ereignis reduziert) ergeben auf unendlichen Strömen keinen Sinn, da eine Beantwortung erst am Ende korrekt ist. Derartige Anfragen können nur gegen gewisse endliche Schritte des Stroms gestellt werden, wo ihr Ergebnis wohldefiniert ist. Ein weiterer Aspekt sind die Regeln in der Ereignisverarbeitung. Regeln bilden eine adäquate Form der Repräsentation von Fachwissen und können zum Beispiel Ereignisse ausfiltern, die bei der Verarbeitung nicht mehr benötigt werden oder auch ein Ereignisformat in ein anderes Format transformieren. Ebenfalls wird durch entsprechende Regeln ermöglicht, mehrere Ereignisse zu einem komplexeren Ereignis zu aggregieren und ein neues abgeleitetes Ereignisobjekt zu generieren. [BrDu15]  
Nach [EcBr09] sind zwei Arten, deduktive und reaktive Regeln wichtig. Auf der Basis von Ereignisanfragen definieren deduktive Regeln neue Ereignisse. Zu erwähnen ist, dass diese auf Ereignissen operieren und nicht auf Fakten, wie deduktive Regeln in der Logikprogrammierung und in deduktiven Datenbanken. Reaktive Regeln hingegen spezifizieren, wie auf Ereignisse reagiert wird.

## Ereignisanfragesprachen

Durch Ereignisanfragesprachen lassen sich Ereignismuster und -reaktionen deklarativ und somit auf einer hohen Abstraktionsebene beschreiben. Die gesuchten Muster müssen in einer speziell dafür vorgesehenen Deklarationssprache definiert bzw. modelliert werden. [EcBr09] Derzeit sind drei Arten unterschiedlicher Sprachen, für die Formulierung von Ereignisanfragen in Gebrauch, die im Folgenden vorgestellt werden.

### Kompositionsoperatoren

Mit Hilfe von Operatoren werden Anfragen gegen einzelne Ereignisse zu komplexen Ereignisanfragen zusammengesetzt. Operatoren die Verwendung finden sind beispielsweise die Konjunktion von Ereignissen, oder aber auch die Negation einer Sequenz. Durch Schachtelung lassen sich ebenfalls komplexere Anfragen ausdrücken. Nach [EcBr09] ermöglichen Kompositionsoperatoren es, komplexe Ereignisse zu spezifizieren und können nach [Frey10] als Funktionen interpretiert werden, deren Parameter und Rückgabewerte Ströme von Ereignissen sind. Kern aller Kompositionsoperatoren basierter Sprachen sind die vier Operatoren Sequenz, Konjunktion, Disjunktion und Negation. Eine Konjunktion (˄) zweier Ereignisse A und B bedeutet, dass Ereignis A und b auftreten. Zu erwähnen ist an dieser Stelle, dass die Konjunktion zeitlich ist und somit die Ereignisse zu unterschiedlichen Zeitpunkten passieren können. Die Disjunktion (˅) besagt, dass Ereignis A oder B auftritt. Die Negation (⌐) bedeutet, dass das entsprechende Ereignis nicht auftritt. Das Erkennen der Abwesenheit von Ereignissen in vielen Event Processing Anwendungen wichtig, um eine Fehlfunktion, Nichtverfügbarkeit oder eine verzögerte Verarbeitung zu erkennen. Aus diesem Grund unterstützen Kompositionsoperator-basierte Sprachen den Negations-Operator. Die Sequenz steht dafür, dass ein Ereignis A auftritt und ein Ereignis B folgt. Mithilfe dieser Operatoren lassen sich Schachtelungen erstellen und somit können komplexe Bedingungen modelliert werden. [Ecke08] Bisher verfolgen beispielsweise IBM ActiveMiddleware Technology (Amit) und ruleCore den auf Kompositionsoperatoren basierenden Ansatz [EcBr09].

### Datenstrom-Anfragesprachen

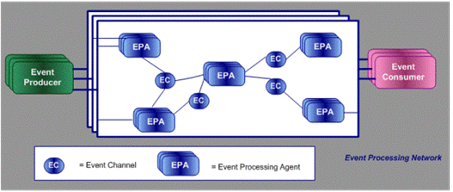
Ein zweiter Sprachstil wurde im Kontext des relationalen Datenstrommanagements entwickelt. Datenstrom-Anfragesprachen basieren auf der Datenbankanfragesprache SQL (Structured Query Language) und haben ein entsprechendes Grundprinzip. Die Datenströme, die die Ereignisse als Tupel enthalten, werden in Relationen umgewandelt und auf diesen Relationen wir dann eine reguläre SQL-Anfrage ausgewertet und das Ergebnis, bei dem es sich um eine Relation handelt, wird wieder in einen Datenstrom gewandelt. [EcBr09] Durch verschiedene Fensteroperationen erfolgt die Umwandlung von Datenströmen in Relationen. Für den umgekehrten Weg, der Umwandlung von Relationen in einen Datenstrom gibt es drei unterschiedliche Möglichkeiten: Nach [EcBr09] geben nur Tupel, die im Vergleich zum vorherigen Ergebnis hinzugekommen sind, ein neues Ereignis, nur Tupel, die wegfallen sind, oder einfach jedes Tupel im (jetzigen) Ergebnis. Continous Query language (CQL) ist beispielsweise ein Vertreter von Datenstrom-Anfragesprachen.

### Produktionsregeln

Bei Produktionsregeln handelt es sich im eigentlichen Sinne nicht um Ereignisanfragesprachen, jedoch eignen sie sich aufgrund ihrer inkrementellen Auswertung. Passiert ein Ereignis, so muss ein entsprechendes Faktum erzeugt werden und Ereignisanfragen werden dann als Bedingungen für diese Fakten ausgedrückt. Der Einsatz von Produktionsregeln ist sehr flexibel und gut an bestehende Programmiersprachen angebunden. Es wird zustandsorientiert und nicht ereignisorientiert gearbeitet. Insgesamt gelten Produktionsregeln weniger effizient, als Datenstrom-Anfragesprachen. [EcBr09]

# Event Processing Network

Ein Event Processing Network zeigt eine Architektur, bei der das Event Processing mithilfe eines Netzwerks aus Event Processing Agents erfolgt. Ein Event Processing Network besteht aus den Komponenten Event, Event Producer, Event Consumer und Event Processing Agent (siehe Abbildung 2). Der Begriff Ereignis (Event) wurde bereits definiert.



Event Processing Network

Abbildung 2: Event Processing Network [IBM10]

Im Folgenden werden die weiteren Komponenten eines Event Processing Networks erläutert.

## Event Producer

Bei einem Event Producer handelt es sich um eine Entität am Rande eines Event Processing Systems. Event Producer generieren ausschließlich Events, ohne diese empfangen oder weiterverarbeiten zu können. [EtNi10] Ein Beispiel für einen Event Producer ist ein GPS-Sensor in einem LKW, der anhand empfangener GPS-Daten regelmäßig Position-Events produziert und sie in das System einspeist. Durch die Produktion dieser Events ist dem Event Processing System immer die genaue Position des LKWs bekannt und andere Komponenten, die die Position des LKW benötigen, können die Daten vom Event Processing System erhalten. [Lack12]

## Event Consumer

Ein Event Consumer ist eine Entität am Rande eines Event Processing Systems, die jedoch Events vom System empfängt [EtNi10] Event Consumer können nicht, wie die Event Producer Events produzieren. Sie empfangen ausschließlich Events von dem Event Processing System, somit liegt ihre Aufgabe in der finalen Verarbeitung von Ereignissen. Die vom System empfangenen Events können für eine spätere Analyse gespeichert oder direkt in einem Programm zur Anzeige gebracht werden. Beziehen wir uns erneut auf das zuvor genannte Beispiel, so könnte der Event Consumer ein Programm zur Verwaltung der Empfangenen GPS-Daten von mehreren LKWs sein. Event Producer und Event Consumer sind völlig unabhängig voneinander. Entsprechend lassen sich Komponenten austauschen oder auch erweitern.

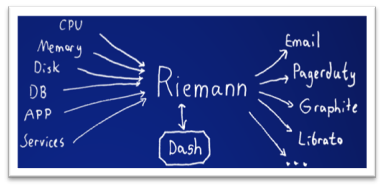
## Event Processing Agents

Event Processing Agents können, im Gegensatz zu Event Producer und Event Consumer, Events sowohl senden, als auch Empfangen und sind somit Produzenten und Konsumenten. Die Verarbeitung von Events wird in einem Netzwerk von Event Processing Agents übernommen. Zu den Aufgaben von Event Processing Agents gehören einfache Routing-Aufgaben, bis hin zur Erkennung von Mustern. Betrachtet man den einfachsten Fall, so nehmen Event Processing Agents Events entgegen und leiten diese weiter. Es erfolgt eine Unterscheidung in Filter Agents, die Events entgegennehmen und Events, die nicht benötigt werden rausfiltern, Transformation Events, die den Inhalt der eingehenden Events modifizieren und Pattern Detect Events, die als Reaktion auf ein gefundenes Muster ein neues komplexes Event erstellen. Event Processing Agents können ebenfalls mehrere Funktionen gleichzeitig erfüllen. [EtNi10]

# Anwendungsgebiete

Event Processing findet in den unterschiedlichsten Bereichen Anwendung. Im Folgenden werden beispielhaft Einsatzgebiete für Event Processing erläutert. Eine Einsatzmöglichkeit zeigt sich im Business Activity Monitoring (BAM). Das Business Activity Monitoring überwacht Geschäftsprozess und unternehmenskritische Ressourcen, indem es Ereignisse in Key Performance Indikatoren zusammenfasst, mit dem Ziel, Probleme und Chancen frühzeitig zu erkennen. [EcBr09] Neben der Aggregation von Standardindikatoren können benutzerdefinierte Auswertungen durchgeführt werden. [SeBr14] Auch Aktien- und Rohstoffpreise können als Ereignisse aufgefasst werden, die frühzeitig und kontinuierlich analysiert werden müssen, um entsprechende „Trends“ zu erkennen und auf diese zu reagieren. Somit ergibt sich eine Anwendung von Event Processing im Bereich der Marktdaten. Nach [SeBr14] kann Event Processing ebenfalls zur Sicherung von B2B und B2C Schnittstellen, anhand der Erkennung gewisser Nutzungsmuster und entsprechender Alarmierung, beitragen. Ebenso ermöglicht Event Processing eine zeitnahe und individuell gestaltbare Analyse von Kundenverhalten, wie beispielsweise die Analyse von Warenkörben in Online-Shops.

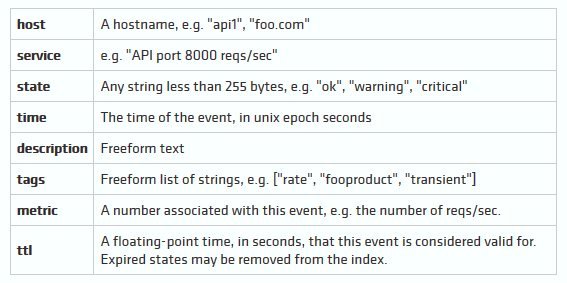
## Tool Riemann



Riemann

Abbildung 3: Riemann

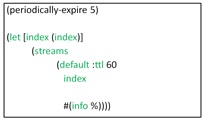
Ein Tool für Event Processing bietet Riemann. Anwendung findet Riemann beispielsweise, um die Latenzverteilung einer Web-App zu testen, für das Senden einer E-Mail für jede Ausnahme in einer App oder auch für das Anzeigen der Top-Prozesse auf jedem Host (siehe Abbildung 3) [Riem17]. Das Tool Riemann aggregiert Ereignisse von Servern und Anwendungen durch eine leistungsfähige Stream-Verarbeitungssprache. Die Idee hinter Riemann ist, das Überwachen und Messen von Ereignissen zu einem einfachen Standard zu machen. Der Zustand von eingehenden Ereignissen kann verfolgt werden. [Turn16] Ereignisse (events) sind nach Riemann nur Strukturen, die über „Protocol Buffer“ versendet werden und in Riemann werden diese als unabänderlich behandelt. Jedes Ereignis hat bestimmte, optionale Felder (siehe Abbildung 4) und wird eindeutig durch seinen „Host“ und „Service“ identifiziert. [Turn26]



Event Felder

Abbildung 4: Event-Felder

Der Index ist in Riemann die Zustandstabelle und verfolgt immer nur das jüngste Ereignis für ein gegebenes (Host-, Service-) Paar. Sowohl das Dashboard, als auch Netzwerkclients und Streams können den Index abfragen, um feststellen zu können, wie das System aussieht. Ereignisse innerhalb des Index haben ein „ttl“ Feld, das anhand einer Gleitkommazahl angibt, wie lange das jeweilige Ergebnis gültig ist. Nach Ablauf dieser Zeit, wird das Ereignis aus dem Index entfernt und mit einem entsprechenden Status („abgelaufen“) wieder in die Ereignisströme (Streams) aufgenommen. [Riem17] Jedes ankommende Ereignis wird einem oder mehreren Streams hinzugefügt. Streams sind Funktionen, denen Ereignisse für Aggregation, Modifizierung oder auch Eskalation zugeführt werden können. Anhand von „child streams“ wird das Filtern oder Partitionieren eines Ereignisstroms, beispielsweise dadurch, dass nur bestimmte Hosts oder Services ausgesucht werden, ermöglicht. [Turn16] Im Folgenden wird nach [Turn16] beispielhaft die Konfiguration von Events, Streams und dem Index gezeigt (siehe Abbildung 5).



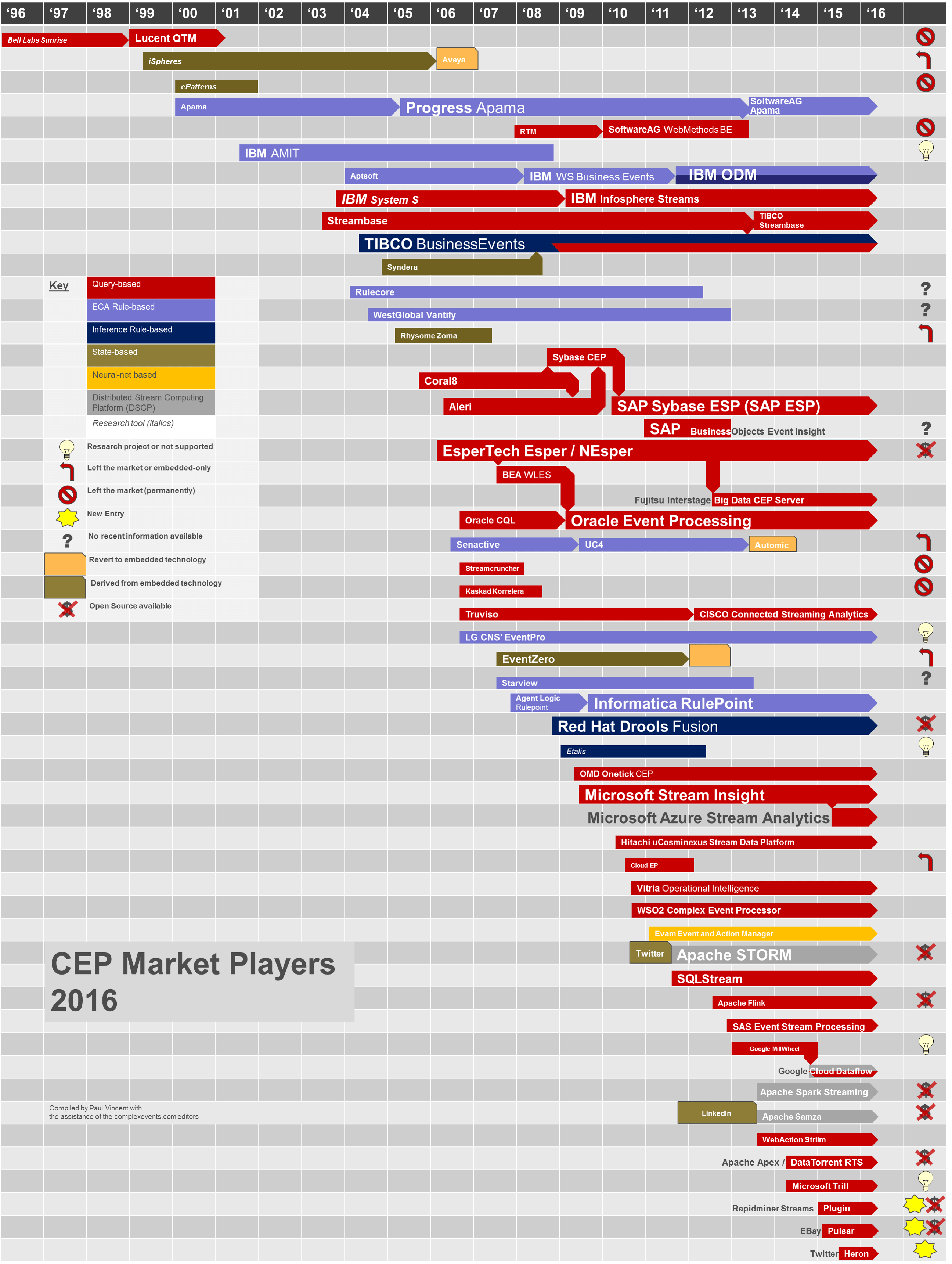
Konfiguration Riemann

Abbildung 5: Konfigurationsbeispiel [Turn16]

Die Funktion (periodically-expire 5) wird genutzt, um alle Events vom Index, die abgelaufen sind, zu entfernen. Hierbei handelt es sich um einen Eventrepeater. Die Zahl 5 steht dafür, dass er alle 5 Sekunden läuft und die abgelaufen Ereignisse (abgelaufene ttl) aus dem Index löscht. Für jedes dieser gelöschten Ereignisse, wird ein neues Ereignis für den indizierten Host und Service erstellt (die Host und Service Felder des abgelaufenen Ereignisses werden kopiert) und dem Stream hinzugefügt. Es folgt der Ausdruck (let [index (index)], der ein neues Symbol „index“ mit dem Wert index darstellt. Diese Funktion sendet Ereignisse zu Riemanns Index. Innerhalb des let-Ausdrucks folgt als nächstes die Streams-Funktion. [Turn16] Bei jeder Stream-Funktion handelt es sich um eine Clojure-Funktion [Riem17]. Die Liste mit Funktionen wird aufgerufen, wenn neue Ereignisse ankommen. Mit (default :ttl 60) wird zunächst eine Standard-ttl von 60 Sekunden gesetzt und somit bestimmt, wie lange ein Ereignis im Index gültig ist. Der Aufruf des folgenden Index Symbols bedeutet, dass alle ankommenden Ereignisse automatisch zum Riemann Index hinzugefügt werden. Durch #(info %) werden alle Ereignisse in die Protoko#lldatei geschrieben. [Turn16]

# Fazit

Das Datenvolumen steigt und mit ihm das Interesse an dem Bereich Event Processing auch wenn Event Processing schon auf mehr als 50 Jahre zurückgeht. In Unternehmen werden immer mehr Daten verarbeitet und auch das WWW trägt zu einer Fülle von Daten bei. Die Nutzung von Event Processing nimmt rapide zu, da es die Möglichkeit bietet, Daten in Echtzeit zu verarbeiten und vor allem auf diese zu reagieren. Zumeist ist Complex Event Processing in vorhandenen Produkten enthalten. Große Banken und Finanzdienstleister nutzen jedoch zumeist Front-Office-Systeme für den Kapitalmarkt, in die sie eine eigene Complex Event Processing Logik einbetten. Die folgende Abbildung 6 aus dem Jahr 2016, voraussichtlich nicht mal ganz vollständig, zeigt die enorme Anzahl der Markt Player im Complex Event Processing Bereich.

 Abbildung 6: Überblick CEP Market Player 2016 [Vinc16]

Complex Event Processing führt, hingegen zu herkömmlichen Datenbankmanagementsystemen Daten auf einer gespeicherten Abfrage aus und nicht eine Abfrage auf gespeicherten Daten. Daten die für einen Anfrage nicht relevant sind, können somit sofort verworfen werden. Aufgrund dieses Unterschieds kann Complex Event Processing auf einen unendlichen Datenstrom angewendet werden und Eingaben können direkt verarbeitet werden. Aufgrund dieses Aspektes, führt Complex Event Processing zur Echtzeitanalyse von Daten und bietet somit einen entscheidenden Vorteil im Hinblick auf die Zunft mit einem Umgang des weiter stetig wachsenden Datenvolumens.



Data Science

# (Complex) Event Processing

## Hausarbeit im Modul Datenbanktechnologien

**von Hanna Looks**

Hochschule Emden/Leer Matrikelnummer 7001523

# Inhaltsverzeichnis

1. Einleitung
2. Ereignisse
3. Event Processing
4. Ereignismuster
5. Ereignisanfragen 5.1 Ereignisanfragesprachen 5.1.1 Kompositionsoperatoren 5.1.2 Datenstrom-Anfragesprachen 5.1.3 Produktionsregeln
6. Event Processing Network 6.1 Event Producer 6.2 Event Consumer 6.3 Event Processing Agents
7. Anwendungsgebiete
8. Fazit Literaturverzeichnis

# Literaturverzeichnis

**[BrDu15]** Bruns, Ralf; Dunkel, Jürgen: Complex Event Processing Komplexe - Analyse von massiven Datenströmen mit CEP, Springer Vieweg, Springer Fachmedien Wiesbaden 2015.

**[ChSc10]** Chandy, K. Mani; Schulte, W. Roy: Designing IT Systems for Agile Companies, Oktober 2010.

**[EcBr09]** Eckert, Michael; Bry, Francois: Complex Event Processing Institut für Informatik, Ludwig-Maximilians-Universität München, http://www.pms.ifi.lmu.de, Springer Verlag, 2009.

**[Ecke08]** Eckert, Michael: Complex Event Processing with XChange: Language Design, Formal Semantics, and Incremental Evaluation for Querying Events, Dissertation, München, 22. Oktober 2008, https://edoc.ub.uni-muenchen.de/9405/1/Eckert\_Michael.pdf

**[EtNi10]** Etzion, Opher; Niblett, Peter:Event Processing in Action. Manning Publications Co., 2010.

**[Frey10]** Frey, Michael: Ein Ansatz für eine Spezifikationssprache zur Generierung von Verarbeitungsvorschriften in heterogenen Complex-Event-Processing-Systemen, 12. Oktober 2010, https://www2.informatik.hu-berlin.de/~frey/publications/frey10.pdf#subsection.2.1.3

**[IBM10]** IBM: A Conceptual Model for Event Processing Systems, 9.2.2010, https://www.ibm.com/developerworks/library/ws-eventprocessing/

**[Lack12]** Lackaw, Maximilian: Konzeption und Realisierung eines generischen Event-Mechanismus für Prozess-Management-Systeme, Diplomarbeit, Universität Ulm, 2012, http://dbis.eprints.uni-ulm.de/827/1/DA\_Thesis\_Lackaw\_Event-Mechanismus.pdf

**[LuSc11]** Luckham,David; Schulte,Roy:Event Processing Glossary–Version2.0. http://www.complexevents.com/wp-content/uploads/2011/08/EPTS\_Event\_Processing\_Glossary\_v2.pdf, 2011.

**[Pier16]** Pierson, Lillian: Data Science für Dummies, WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 1. Auflage 2016.

**[Riem17]** http://riemann.io/concepts.html, 2017.

**[SeBr14]** Serban, Dr. Floarea; Brise, Dr. Yves:Complex Event Processing – Eine strategische Big Data Entscheidung, http://ipt.ch/complex-event-processing-eine-strategische-big-data-entscheidung/, 11. August 2014.

**[Turn16]** Turnbull, James: The Art of Monitoring, https://artofmonitoring.com/TheArtOfMonitoring\_sample.pdf, 2016.

**[VaHo13]** Vandenhouten, Ralph; Holland-Moritz, Ralph: Erkennung von Bewegungsprofilen im Gebäudemanagement mithilfe von Complex Event Processing, THWildau Wissenschaftliche Beiträge 2013.

**[Vinc16]** Vincent, Paul: CEP Tooling Market Survey 2016, http://www.complexevents.com/2016/05/12/cep-tooling-market-survey-2016/, 12.05.2016.