Distributed Event Based Systems – Complex Event Processing

Netzprogrammierung
(Algorithmen und Programmierung V)
Our topics last week

Descriptive models for distributed system design

Physical model

Architectural model

Architectural elements

- Communicating entities
  - Processes
  - Objects
  - Components
  - Web Services

- Communication paradigm
  - Inter-process communication
    - UDP sockets
    - TCP sockets
    - Multicast
  - Indirect communication
  - Remote invocation

- Roles and responsibilities
  - Architectural styles
    - Client-server
    - Peer-to-peer

- Placement
  - Multiple server
  - Proxy/Cache
  - Mobile code

Architectural patterns

- Vertical distribution
  - Multi-tier
  - Thin/Fat Client

- Horizontal distribution

Interaction model

Interaction model

Failure model

Security model

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Our topics today

• Distributed Event Based Systems
• Complex Event Processing
• CEP Languages
• CEP Reference Architecture and CEP Functions
Event Processing vs. Databases

**Ex-Post Data Queries**

- Database Queries
  - Query Processor
  - Database

**Real Time Data Processing**

- Event Subscriptions
  - Incoming Events
    - Processing moment of events
    - Time
      - Future
      - Past
Knowledge Value of Events

- Knowledge of the event at the time of occurrence (Real-Time)
- Knowledge of the event within a certain time frame after occurrence (Some time after event)
- Knowledge of the event after a long time (Post-Processing)

- Proactive actions (Before the event)
- Reactive actions (At event)

Predictive Analysis
Complex Event Processing
Late reaction or Long term report
Historical Event

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Complex Events – What are they?

Complex Events are aggregates, derivations, etc. of Simple Events

- Complex Event Processing (CEP) will enable, e.g.
  - Detection of state changes based on observations
  - Prediction of future states based on past behaviours
Complex Event Processing – What is it?

CEP is about complex event detection and reaction to complex events

- Efficient (near real-time) **processing** of large numbers of events
- **Detection, prediction** and **exploitation** of relevant complex events
- Supports **situation awareness**, track & trace, sense & respond

**Event Cloud**

(orderless events)

**Event Streams**

(order events)

**Continuous Event Queries**

```sql
INSERT INTO OrdersMatch_s
SELECT B.ID, S.ID, B.Price
FROM BuyOrders_s B, Trades_s T, SellOrders_s S,
MATCHING [30 SECONDS: B, !T, S]
ON B.Price = S.Price = T.Price;
```

**Patterns, Rules**

A → B → C

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Complex Event Processing – What is it?

Complex Event Processing (CEP) is a **discipline** that deals with event-driven behavior. Selection, aggregation, and event abstraction for generating higher level complex events of interest.
CEP – Example Application Domains

Quick decisions, and reactions to threats and opportunities according to events in business transactions

Monitoring, BAM, ITSM,

Monitor and detect exceptional IT service and business behavior from occurred events

RTE

Valuable Information at the Right Time to the Right Recipient

Information Dissemination

Expert Decision Management

CEP Media

Detect
Decide
Respond

Enterprise Decision Management

Expert Systems

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Core CEP Life Cycle

1. Event Production
2. Event Definition
3. Event Selection
4. Event Aggregation
5. Event Handling
6. Event Consumption
Basic Definition (1)

Atomic Event (also raw event or primitive event)
- An atomic event (also raw event or primitive event) is defined as an instantaneous (at a specific point in time), significant (relates to a context), indivisible (cannot be further decomposed and happens completely or not at all) occurrence of a happening.

Complex Event
- composed (composite event) or derived (derived event) from occurred atomic or other complex event instances, e.g. according to the operators of an event algebra or as a result of applying an algorithmic function or process to one or more other events
- included events are called components while the resulting complex event is the parent event
- first event instance contributing to the detection of a complex event is called initiator, where the last is the terminator; all others are called interiors

Event Pattern
- An event pattern (also event class, event definition, event schema, or event type) describes the structure and properties of an (atomic or complex) event
- It describes on an abstract level the essential factors that uniquely identify the occurrence of an event of that type, i.e. its detection condition(s) and its properties with respect to instantiation, selection and consumption.
Basic Definition (2)

Event Instance
- A concrete instantiation of an event pattern is a specific event instance (also event object).

Situation
- A situation is initiated or terminated by one or more (complex) events, i.e. the effect of a complex event (complex event + conditional context)

Situation Individuals
- Situation individuals are discrete individuals of situations with a fixed context allocation of time, date, location, participant, etc. They are not repeatable and are temporally connected.

Complex Action
- Compound action from occurred atomic or other complex action instances
Example Snoop Event Algebra Operators:

- **Sequence Operator (•):** $(E_1;E_2)$
- **Disjunction Operator (∨):** $(E_1 ∨ E_2)$, at least one
- **Conjunction Operator (∧):** $(E_1 ∧ E_2)$
- **Simultaneous Operator (=):** $(E_1 = E_2)$
- **Negation Operator (¬):** $(E_1 ∧ ¬ E_2)$
- **Quantification (Any):** Any$(n)$ $E_1$, when $n$ events of type $E_1$ occurs
- **Aperiodic Operator (Ap):** $Ap(E_2, E_1, E_3)$, $E_2$ Within $E_1$ & $E_3$
- **Periodic Operator (Per):** $Per(t, E_1, E_2)$, every $t$ time-steps in between $E_1$ and $E_2$
Example – Interval-based Sequence

Example: $D = A; (B; C)$
### Event Detection Operators & Windowing

**Window** \([t-5 \leftrightarrow t]\)

<table>
<thead>
<tr>
<th></th>
<th>(t-8)</th>
<th>(t-7)</th>
<th>(t-6)</th>
<th>(t-5)</th>
<th>(t-4)</th>
<th>(t-3)</th>
<th>(t-2)</th>
<th>(t-1)</th>
<th>(t)</th>
<th>(t+1)</th>
<th>(t+2)</th>
<th>(t+3)</th>
<th>(t+4)</th>
<th>(t+5)</th>
<th>(t+6)</th>
<th>(t+7)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Events</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t-1)</td>
<td>(A_1)</td>
<td>(B_1)</td>
<td>(C_1)</td>
<td>(A_2)</td>
<td>(B_2)</td>
<td>(D_1)</td>
<td>(A_3)</td>
<td>(B_3)</td>
<td>(C_3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t-2)</td>
<td>(A_1)</td>
<td>(B_1)</td>
<td>(C_1)</td>
<td>(A_2)</td>
<td>(B_2)</td>
<td>(D_1)</td>
<td>(A_3)</td>
<td>(B_3)</td>
<td>(C_3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(t-3)</td>
<td>(A_1)</td>
<td>(B_1)</td>
<td>(C_1)</td>
<td>(A_2)</td>
<td>(B_2)</td>
<td>(D_1)</td>
<td>(A_3)</td>
<td>(B_3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Event Detection Pattern** \([A ; B]\)

**Matches**
- **Time** \(t\) \(\rightarrow\) 2 or 3 (based on event consumption policy)
- **Time** \(t-1\) \(\rightarrow\) 1
- **Time** \(t-2\) \(\rightarrow\) 1
- **Time** \(t-3\) \(\rightarrow\) 2 or 3

---

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Event Processing Network is a collection of Event Processing Agents.

The EPN describes the “programming in the large”, while each individual agent describes the “programming in the small”.

Event Processing Agent and Event Processing Network

EPN Example

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
What is the idea?

The EPN (event processing network) represents the event processing application as a directed graph:

- The nodes represent event processing agents (and states in some models)
- The edges represent either individual events or event streams (depends on the processing type). A generalized EPN may support both.

The coverage of EPN varies

- It may cover some language operator (e.g. SQL like language, rule language, …)
- It may take a broader approach and cover also routing decisions, pub/sub etc…
- EPN can have a specific programming model, or an hybrid programming model, where one or more languages represent the agents.
**EPN example**

- Blood pressure reading
- Fever C
- Fever F
- Aggregator: Max.
- Filter: Diabetes
- Pattern detection: Alert physician
- Pattern detection: Alert Nurse
- Pattern Detection: Continuous Fever
- Enrich: Is diabetic?
- Translate: C2F fever

---

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Example Event Processing Languages

- Reaction RuleML
- WBE
- Prova
- ECA / Reaction Rules
- TIBCO
- XChangeEQ
- AMiT
- RuleCore
- Agent Logic
- Spade
- Streambase
- Agent Oriented
- SQL extension
- Oracle
- State oriented
- Imperative/Script Based
- Netcool Impact
- Coral8
- EventZero
- Aleri
- Esper
- Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Some EPN, event flow and data flow oriented languages

Agent Logic

Event Zero

Oracle/BEA

Spade (IBM System S)

Streambase

A SPADE program builds a data-flow network out of operators
Example Research Prototypes

• **Finite State Automata, e.g.:**
  – ADAM, ACOOD, ODE, SAMOS, COMPOSE, SNOOP (active database research in 80s & 90s)
  – SASE, Cayuga, Esper

• **Rule-Based CEP Engines, e.g.:**
  – Prolog, **Prova**, Drools, ETALIS

• **Data Stream Engines, e.g.:**
  – Stream CQL, PIPE, TelegraphCQ, Gigascope, Aurora Borealis, SPADE

• **Stream Reasoning, e.g.:**
  – C-SPARQL, EP-SPARQL, SCEPter, SPARQLStream, CQELS
Implementations - Example Systems: TIBCO BusinessEvents

TIBCO BusinessEvents “event server” / event processing engines
Multiple languages / engines in various packaging options
- UML-based event and concept (object) class hierarchical models
- Java-based language
- Eclipse-based IDE

Distributed execution model:
- Inference agents, Query agents, Datagrid agents
- In-memory, memory+grid, explicit grid operational modes
- “Grid Options” of TIBCO ActiveSpaces / Oracle Coherence
- eXtreme Event Processing use cases: Fedex, US Govt, …

Transactional model option:
- TIBCO AS Transactions
- eXtreme Event Processing use cases: Telco

Extensible: eg OWL import, RDF import, etc
Implementations - Example Systems: TIBCO BusinessEvents (ctd)

Events

- Event and Data Structures
- Inference Rules
- States and Transitions
- Sets and Queries
- Temporal Patterns
- Decision Tables

via JMS, RV, MQ, TCP/IP, etc...

Information

- Stored Events and Data

Continuously process events using procedural and declarative event processing elements

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Implementations - Example Systems: Oracle Event Processing (formerly Oracle CEP)

Development platform for event processing applications
Application model based on EPN (event processing network) abstraction running on top of OSGi-based Java container.

Language is an extension of SQL with stream and pattern matching extensions
Implementations - Example Systems:
Oracle Event Processing (formerly Oracle CEP)

CQL: Continuous Query Language
Leverages SQL, extended with Stream, Pattern-matching, and Java.
Continuous: driven by time, and events
Highly-sophisticated push-down technology to RDBMS, Cache, Hadoop
Implementations - Example Systems: Prova (http://prova.ws)

- Java JVM based, open source rule language for reactive agents and event processing
- Leverages declarative ISO Prolog standard extended with (event,message) reaction logic, type systems (Java, Ontologies), query built-ins, dynamic Java integration.
- Combines declarative, imperative (object-oriented) and functional programming styles
- Designed to work in distributed Enterprise Service Bus and OSGi environments
- Supports strong, loose and decoupled interaction
- Compatible with rule interchange standards such as Reaction RuleML
Implementations - Example Systems: IBM WODM Decision Server Events

**Decision Server Events** component of IBM WebSphere Operational Decision Management (WODM)
Manages business events flowing across systems and people to provide timely insight and responses
Detect, evaluate, and respond to events
Discover event patterns and initiate actions
Implementations - Example Systems: IBM CICS Transaction Server for z/OS
Implementations - Example Systems: IBM InfoSphere Streams – Stream Processing Language

IBM InfoSphere Streams: platform for analyzing big data in motion i.e. high-volume, continuous data streams SPL

IBM Streams Processing Language (SPL)
- Programming language for InfoSphere Streams
- Exposes a simple graph-of-operators view
- Provides a powerful code-generation interface to C++ and Java
Functional View source: definitions of EP

Event Processing, Complex Event Processing, Event Stream Processing

Event Abstraction, Event Pattern Detection, Event Composition etc. etc.

Event Producer
(Event Source, Event Emitter)

Event Consumer
(Event sink, event handler, event listener)

Design time  Run time  Administration
EPTS Reference Architecture - Functional View

Architect and Developer perspective
- includes the 3 main functions (development, run-time and administration),
- targets primarily the automated event processing operations

Run-time functions in 2 main groups:
- the event infrastructure (sources and consumers) external to the event processor under consideration,
- the event processor.
Reference Architecture: Functional View

Event Reaction
Assessment, Routing, Prediction, Discovery, Learning

Complex Event Detection
Consolidation, Composition, Aggregation

Event Preparation
Identification, Selection, Filtering, Monitoring, Enrichment

Event Analysis
Analytics, Transforms, Tracking, Scoring, Rating, Classification

Event Production
Publication, Retrieval

Event Consumption
Dashboard, Apps, External Reaction

State Management

Event Process Monitoring, Control

Event Actions
Event Correlations and patterns
Event Computations
Event Selections
Event Production/Consumption

Process Updates
Resource Utilization
High Availability
Security
Start/Stop

Design time
Run time
Administration

Barry Linnert, linnert@inf.fu-berlin.de, Netzprogrammierung WS 2015/16
Reference Architecture: Functional View / Runtime

- **Event Reaction**
  - Assessment, Routing, Prediction, Discovery, Learning

- **Complex Event Detection**
  - Consolidation, Composition, Aggregation

- **Event Analysis**
  - Analytics, Transforms, Tracking, Scoring, Rating, Classification

- **Event Preparation**
  - Identification, Selection, Filtering, Monitoring, Enrichment

- **Event Production**
  - Publication, Retrieval

- **Event Consumption**
  - Dashboard, Apps, External Reaction
Reference Architecture: Functional View / Runtime

**Event Production**: the source of events for event processing.

- **Event Publication**: As a part of event production, events may be published onto a communication mechanism (e.g., event bus) for use by event consumers (including participants in event processing). This is analogous to a "push" system for obtaining events.

- **Event Retrieval**: As a part of event production, events may be explicitly retrieved from some detection system. This is analogous to a "pull" system for obtaining events.
Reference Architecture: Functional View / Runtime

**Event Consumption**: the process of using events from event publication and processing. Event processing itself can be an event consumer, although for the purposes of the reference architecture, event consumers are meant to indicate downstream consumers of events generated in event processing.

- **Dashboard**: a type of event consumer that displays events as they occur to some user community.
- **Applications**: a type of event consumer if it consumes events for its own processes.
- **External Reaction**: caused through some event consumption, as the result of some hardware or software process.

<table>
<thead>
<tr>
<th>Event Production</th>
<th>Event Consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Publication, Retrieval</td>
<td>Dashboard, Apps, External Reaction</td>
</tr>
</tbody>
</table>
Event Preparation: the process of preparing the event and associated payload and metadata for further stages of event processing.

- **Entity Identification**: incoming events will need to be identified relative to prior events, such as associating events with particular sources or sensors.

- **Event Selection**: particular events may be selected for further analysis. Different parts of event processing may require different selections of events. See also event filtering.

- **Event Filtering**: a stream or list of events may be filtered on some payload or metadata information such that some subset is selected for further processing.

- **Event Monitoring**: particular types of events may be monitored for selection for further processing. This may utilise specific mechanisms external to the event processing such as exploiting event production features.

- **Event Enrichment**: events may be "enriched" through knowledge gained through previous events or data.
Event Analysis: the process of analysing suitably prepared events and their payloads and metadata for useful information.

- **Event Analytics**: the use of statistical methods to derive additional information about an event or set of events.
- **Event Transforms**: processes carried out on event payloads or data, either related to event preparation, analysis or processing.
- **Event Tracking**: where events related to some entity are used to identify state changes in that entity.
- **Event Scoring**: the process by which events are ranked using a score, usually as a part of a statistical analysis of a set of events. See also *Event Analytics*.
- **Event Rating**: where events are compared to others to associate some importance or other, possibly relative, measurement to the event.
- **Event Classification**: where events are associated with some classification scheme for use in downstream processing.
Complex Event Detection: the process by which event analysis results in the creation of new event information, or the update of existing complex events.

- **Event Consolidation:** combining disparate events together into a "main" or "primary" event. See also event aggregation.

- **Event Composition:** composing new, complex events from existing, possibly source, events.

- **Event Aggregation:** combining events to provide new or useful information, such as trend information and event statistics. Similar to event consolidation.
Reference Architecture: Functional View / Runtime

**Event Reaction:** the process subsequent to event analysis and complex event detection to handle the results of analysis and detection.

- **Event Assessment:** the process by which an event is assessed for inclusion in some process, incorporation in some other event, etc.
- **Event Routing:** the process by which an event is redirected to some process, computation element, or other event sink.
- **Event Prediction:** where the reaction to some event processing is that some new event is predicted to occur.
- **Event Discovery:** where the reaction to some event processing is the disclosure of a new, typically complex, event type.
  - Note that event prediction is predicting some future event, usually of a known type, whereas event discovery is the uncovering of a new event type. See also event-based learning.
- **Event-based Learning:** the reaction to some event processing that uses new event information to add to some, typically statistical-based, understanding of events.
  - Note that event-based learning is a specialisation of general machine learning and predictive analytics.

---

**Event Reaction**

Assessment, Routing, Prediction, Discovery, Learning
Reference Architecture: Functional View / Design time

Covers the definition, modeling, improvement / maintenance of the artefacts used in event processing:

• **event definitions**, including event metadata and payloads,

• event and event object organisations and structures,

• event processing transformations / queries / rules / procedures / flows / states / decisions / expressions (although these can sometimes be considered as administrative updates in some situations).
Reference Architecture: Functional View / Administration

Administrative concepts of monitoring and control. This may involve

• starting and stopping the application and event processing elements, including application monitors

• providing and updating security levels to event inputs and outputs (also can design-time)

• management of high availability and reliability resources, such as hot standby processes

• **resource utilisation monitoring** of the event processing components

• process updates, such as how-swapping of event processing definitions to newer versions.
Summary - Event Processing Patterns

Functions from Reference Architecture are a guide to possible event processing patterns

<table>
<thead>
<tr>
<th>Event Preparation</th>
<th>Identification, Selection, Filtering, Monitoring, Enrichment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex Event Detection</td>
<td>Consolidation, Composition, Aggregation</td>
</tr>
<tr>
<td>Event Analysis</td>
<td>Analytics, Transforms, Tracking, Scoring, Rating, Classification</td>
</tr>
<tr>
<td>Event Reaction</td>
<td>Assessment, Routing, Prediction, Discovery, Learning</td>
</tr>
</tbody>
</table>
So, what have we learned today?

- Basic Definitions and Approach of Complex Event Processing in Distributed Event Based Systems
  - event algebra operators for event pattern definitions
  - windowing techniques for real-time event pattern matching

- Overview CEP Languages
  - Rule-based, agent-based, SQL extensions, state-based, imperative, scripting

- CEP Reference Architecture with typical CEP functions
  - production, consumption, preparation, analysis, detection, reaction functions
Questions

• What is the purpose of complex event processing?
• How does complex event processing in distributed event based systems work? What is a complex event pattern? Name typical event algebra operators.
• Name examples of CEP language approaches.
• Describe the main roles in the CEP reference architecture and name the five functional groups in the runtime view. Name a function example for each group (production, consumption, preparation, analysis, detection, reaction functions).
Our topics next week

Descriptive models for distributed system design

Physical model

Architectural model

Architectural elements

- Communicating entities
  - Processes
  - Objects
  - Components
  - Web Services

- Communication paradigm
  - Inter-process communication
    - UDP sockets
    - TCP sockets
    - Multicast
  - Indirect communication
  - Remote invocation

Roles and responsibilities

- Architectural styles
  - Client-server
  - Peer-to-peer

Placement

- Multiple server
- Proxy/Cache
- Mobile code

Architectural patterns

- Vertical distribution
  - Multi-tier
  - Thin/Fat Client

- Horizontal distribution

Interaction model

Failure model

Security model
Next class

Distributed Objects and Components
References


Opher Etzion and Peter Niblett. 2010. Event Processing in Action (1st ed.). Manning Publications Co., Greenwich, CT, USA.

Adrian Paschke, Paul Vincent, Alexandre Alves, Catherine Moxey: Tutorial on advanced design patterns in event processing. DEBS 2012: 324-334; www.slideshare.net/isvana/epts-debs2012-event-processing-reference-architecture-design-patterns-v204b


Prova Rule Engine http://www.prova.ws/