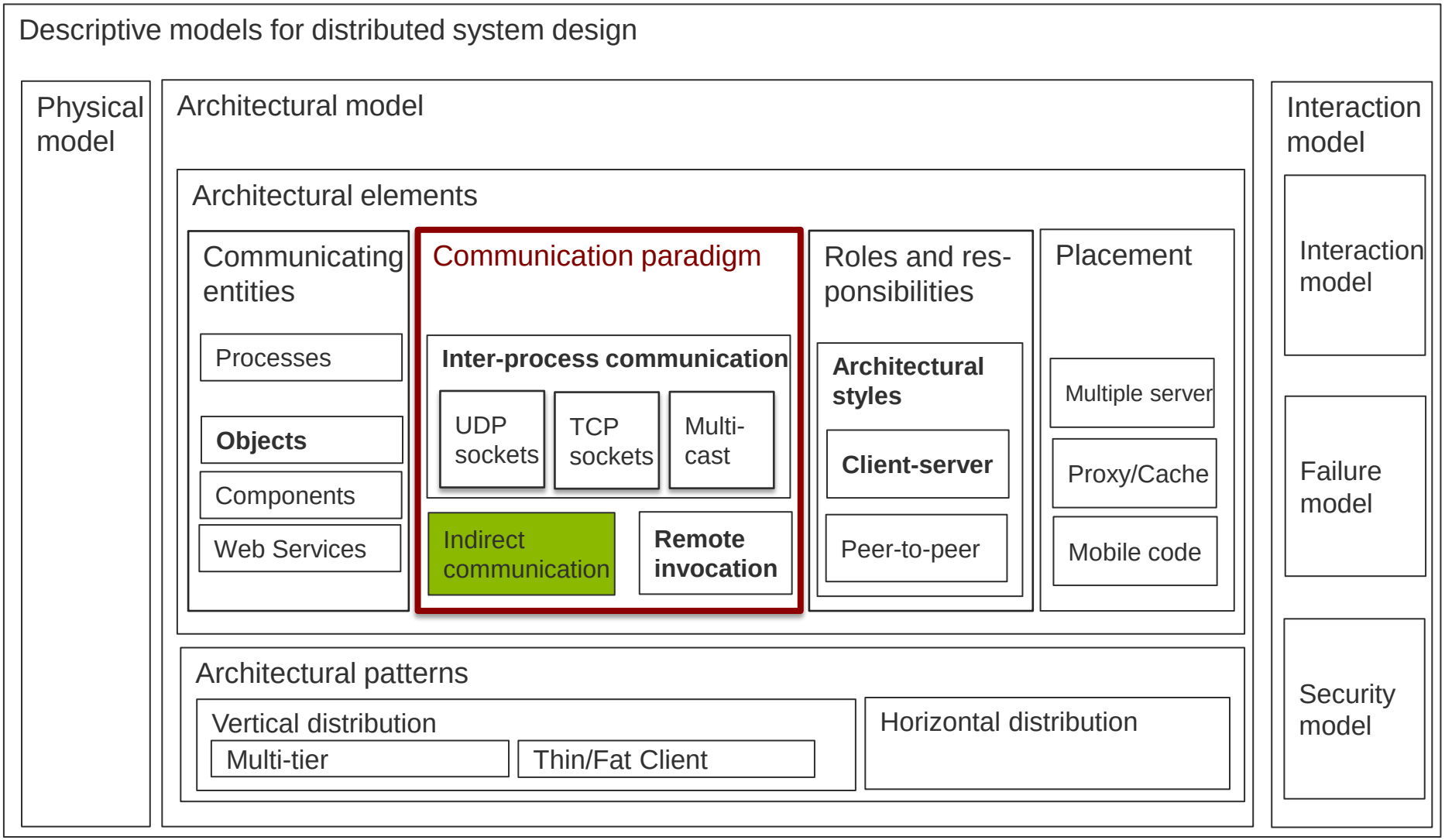
Freie Universität Berlin

Distributed Event Based Systems – Complex Event Processing

Netzprogrammierung

(Algorithmen und Programmierung V)

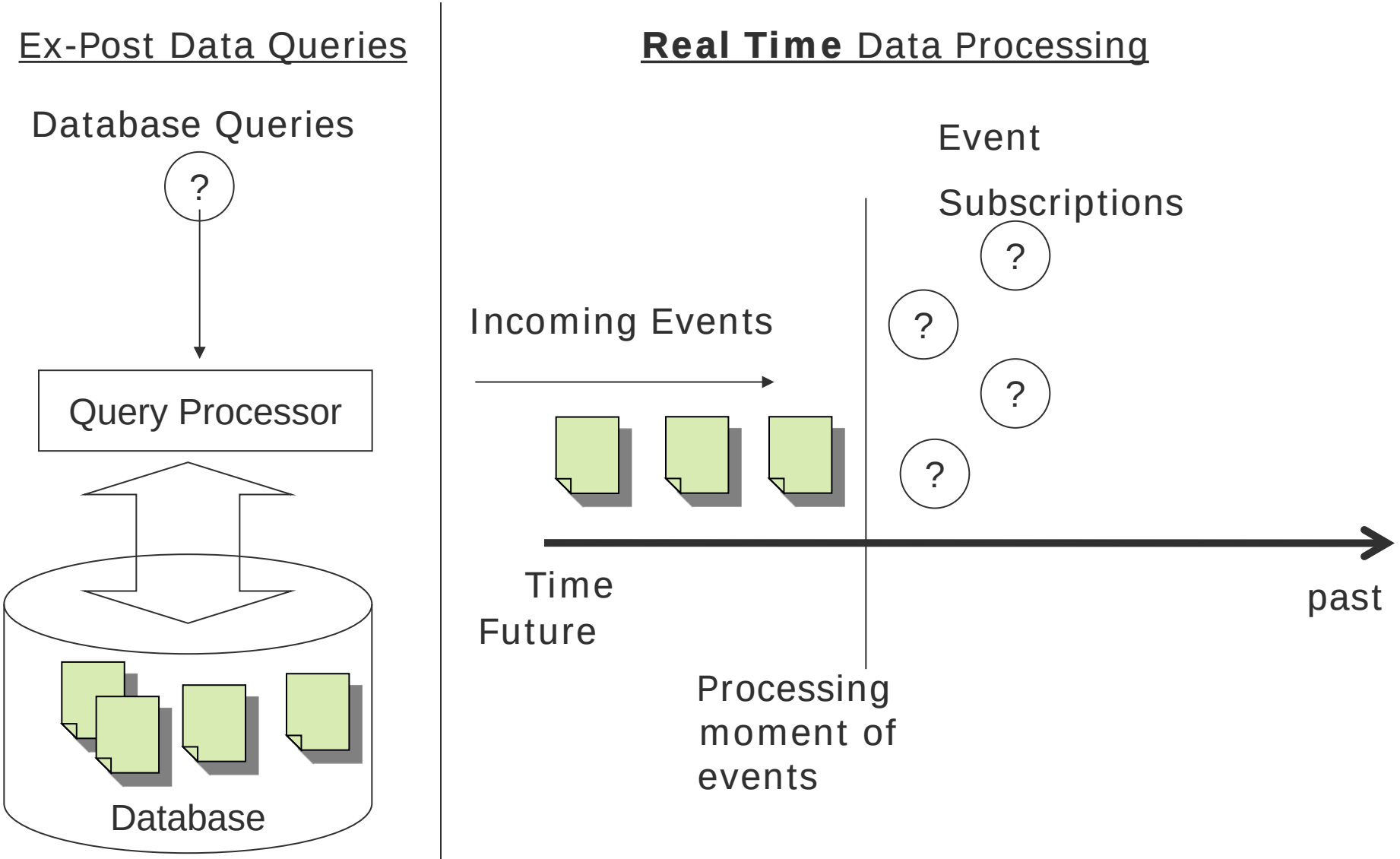
Our topics last week



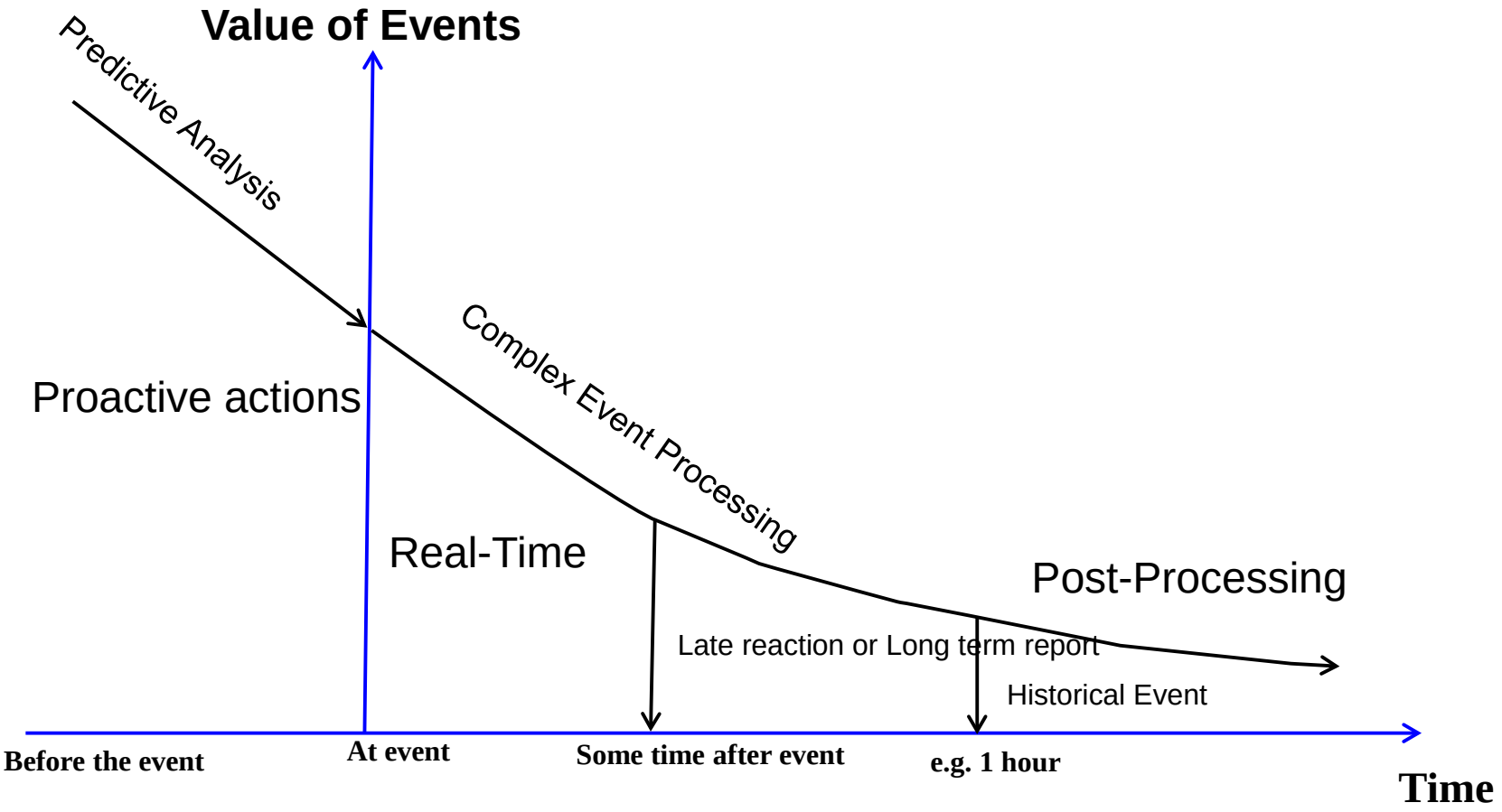
Our topics today

- Distributed Event Based Systems
- Complex Event Processing
- CEP Languages
- CEP Reference Architecture and CEP Functions

Event Processing vs. Databases

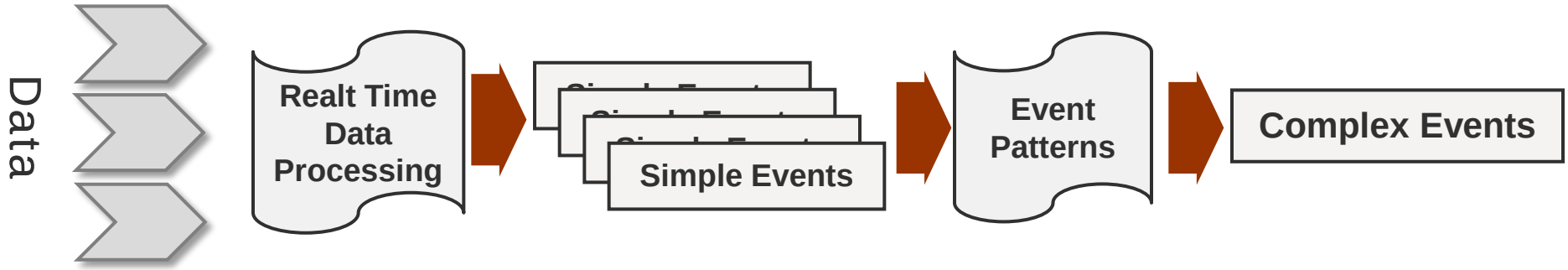


Knowledge Value of Events



Complex Events – What are they?

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

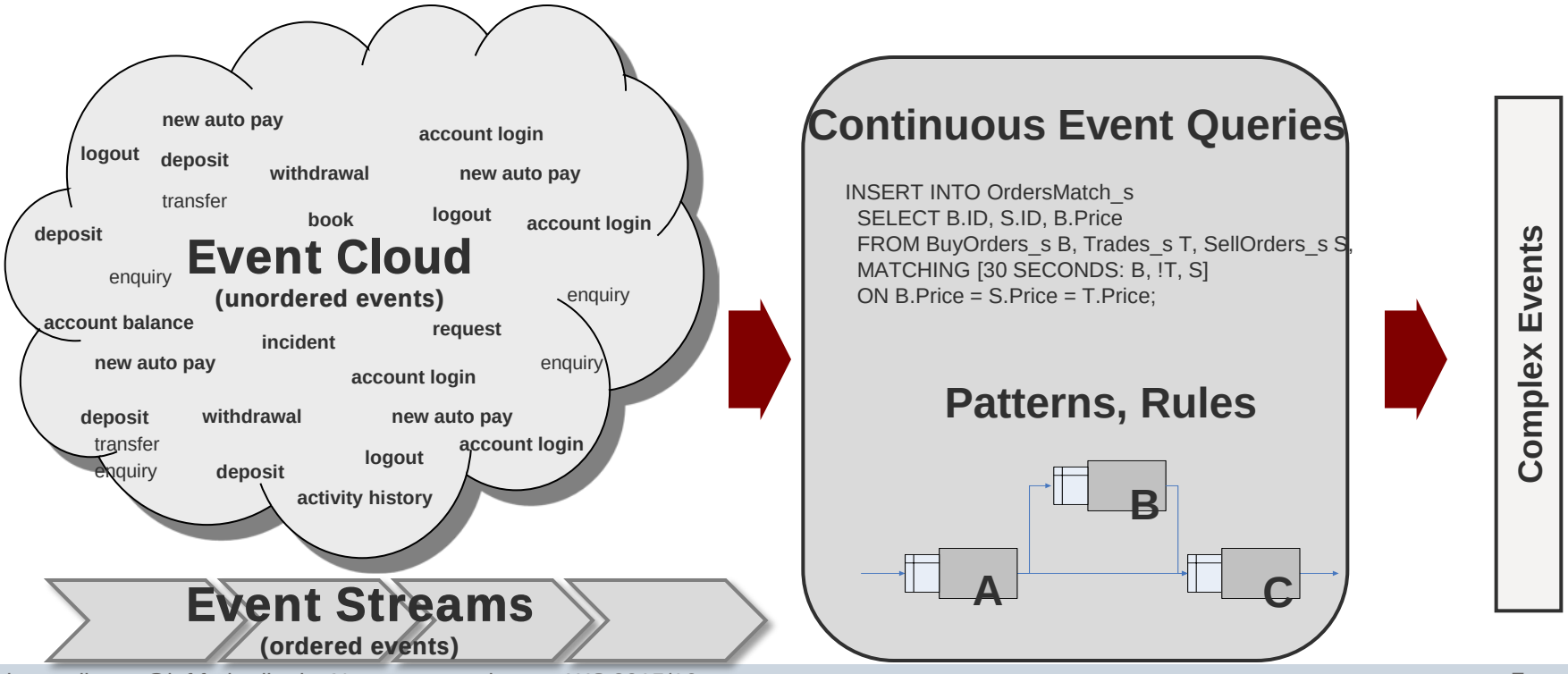


- Complex Event Processing (CEP) will enable, e.g.
  - **Detection** of state changes based on observations
  - **Prediction** of future states based on past behaviours

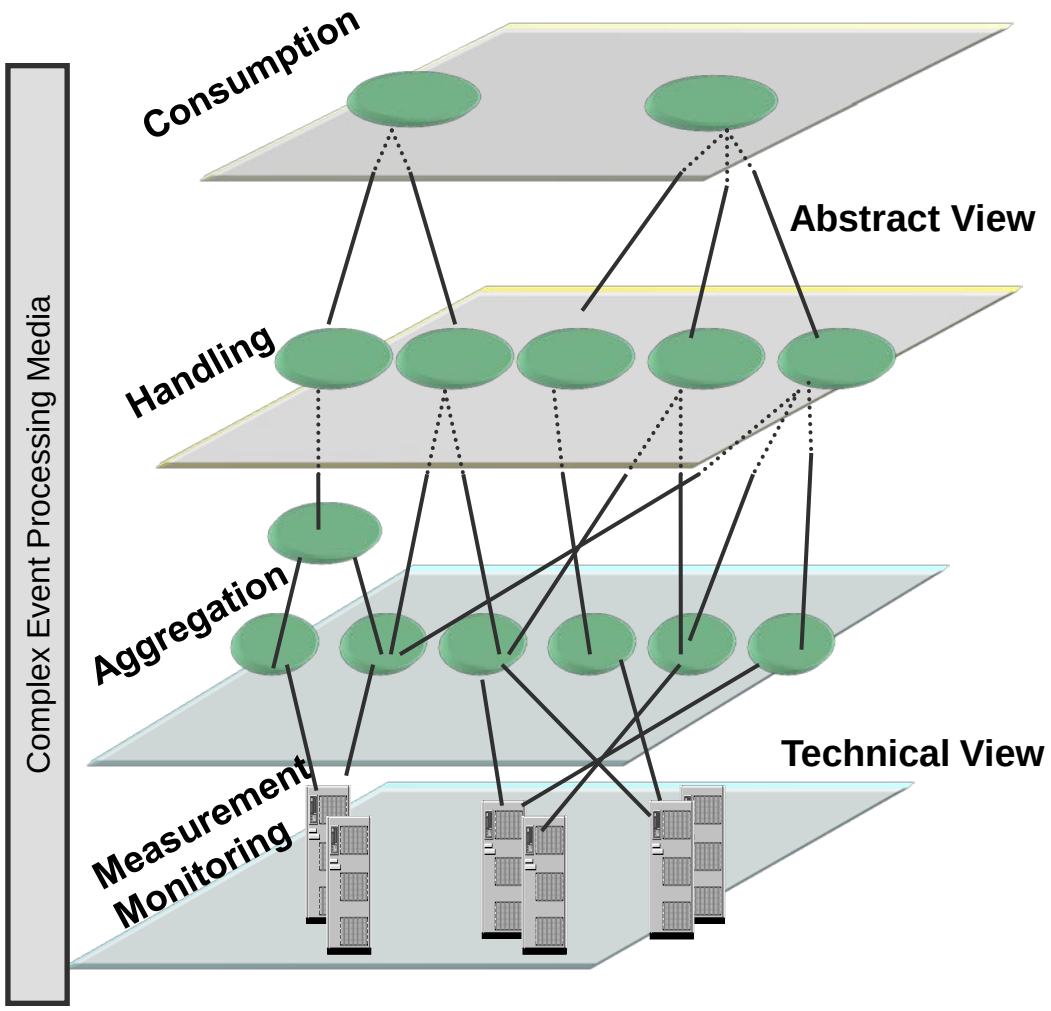
Complex Event Processing – What is it?

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**



Complex Event Processing – What is it?



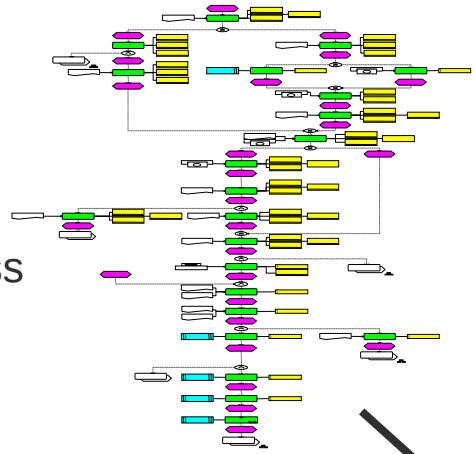
Complex Event Processing (CEP) is a discipline that deals with event-driven behavior

Selection, aggregation, and event abstraction for generating higher level complex events of interest

CEP – Example Application Domains

RTE

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

Valuable Information at the Right Time to the Right Recipient

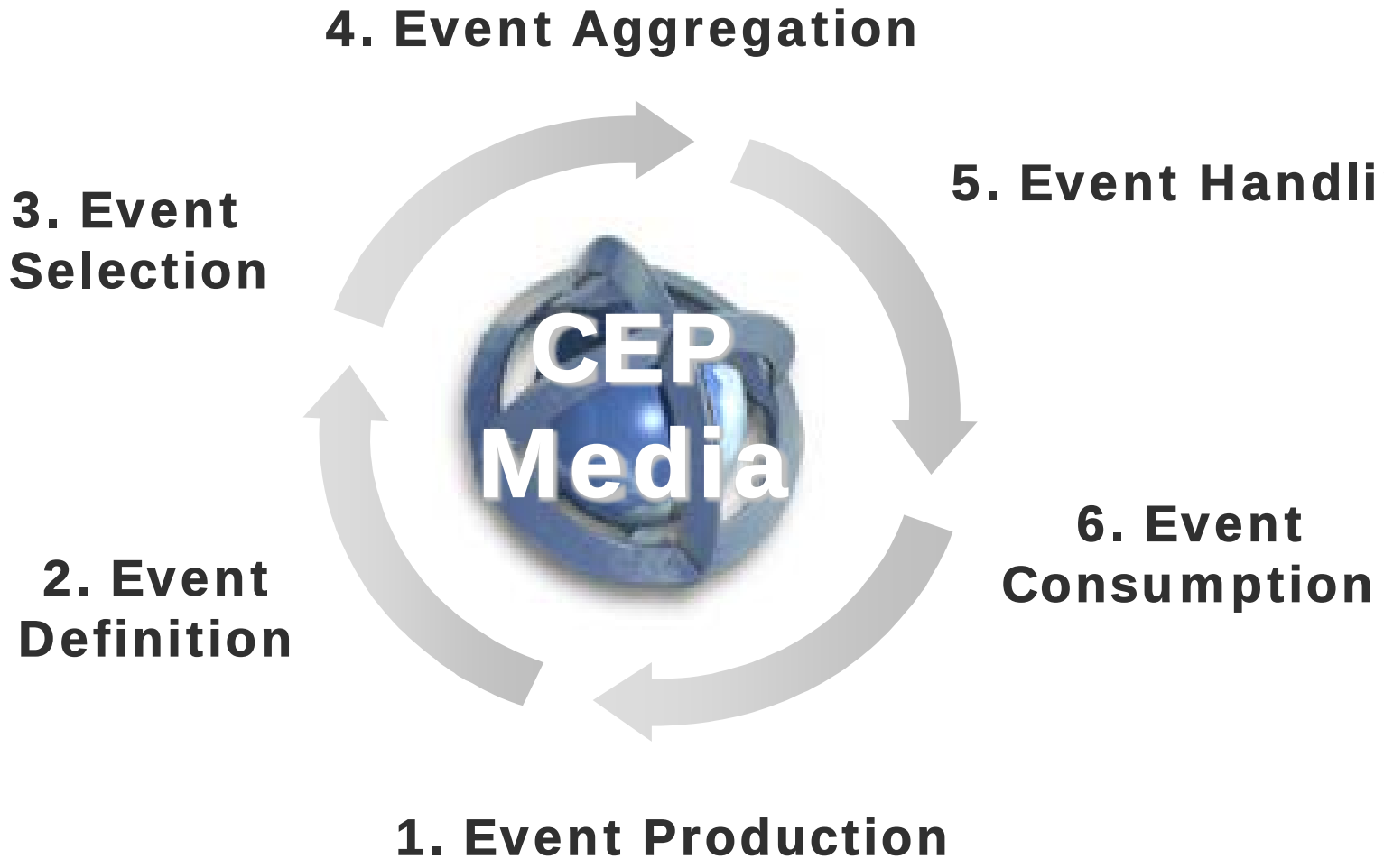
Information Dissemination

Enterprise Decision Management

Expert Systems

Expert Decision Management

Core CEP Life Cycle



Basic Definition (1)

Freie Universität Berlin

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)

Freie Universität Berlin

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

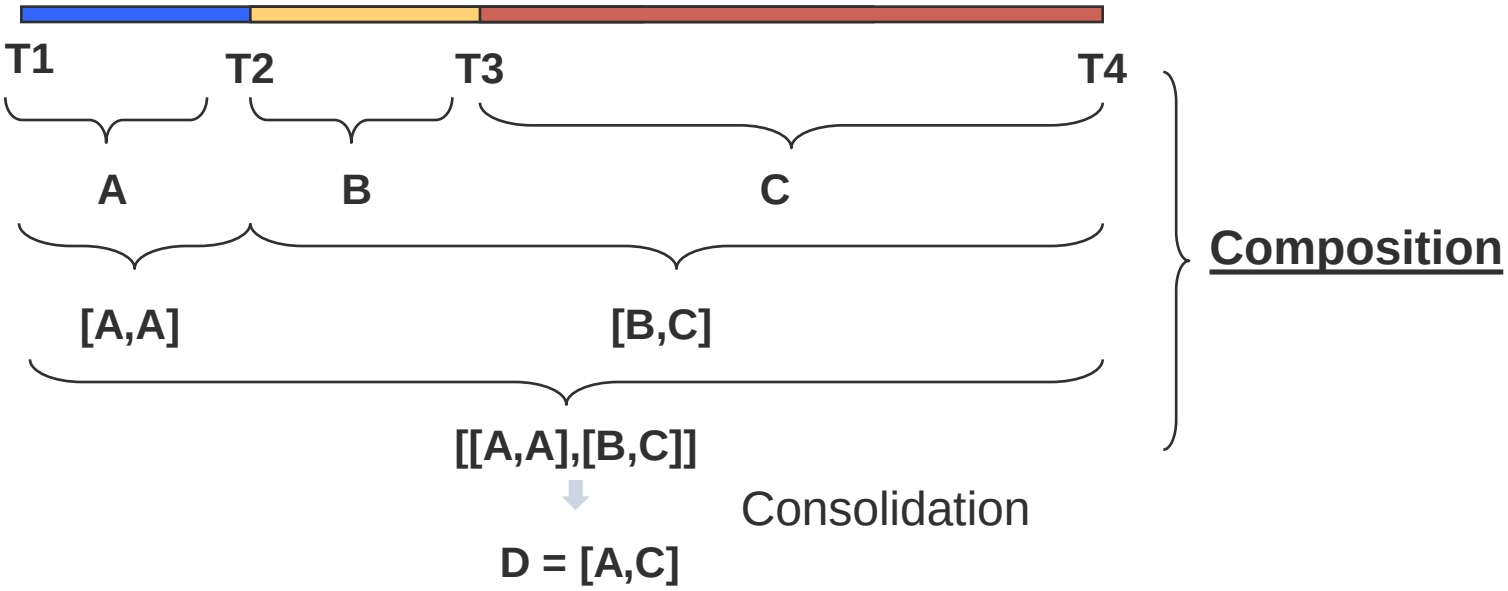
Complex Event Patterns – How?

Example Snoop Event Algebra Operators:

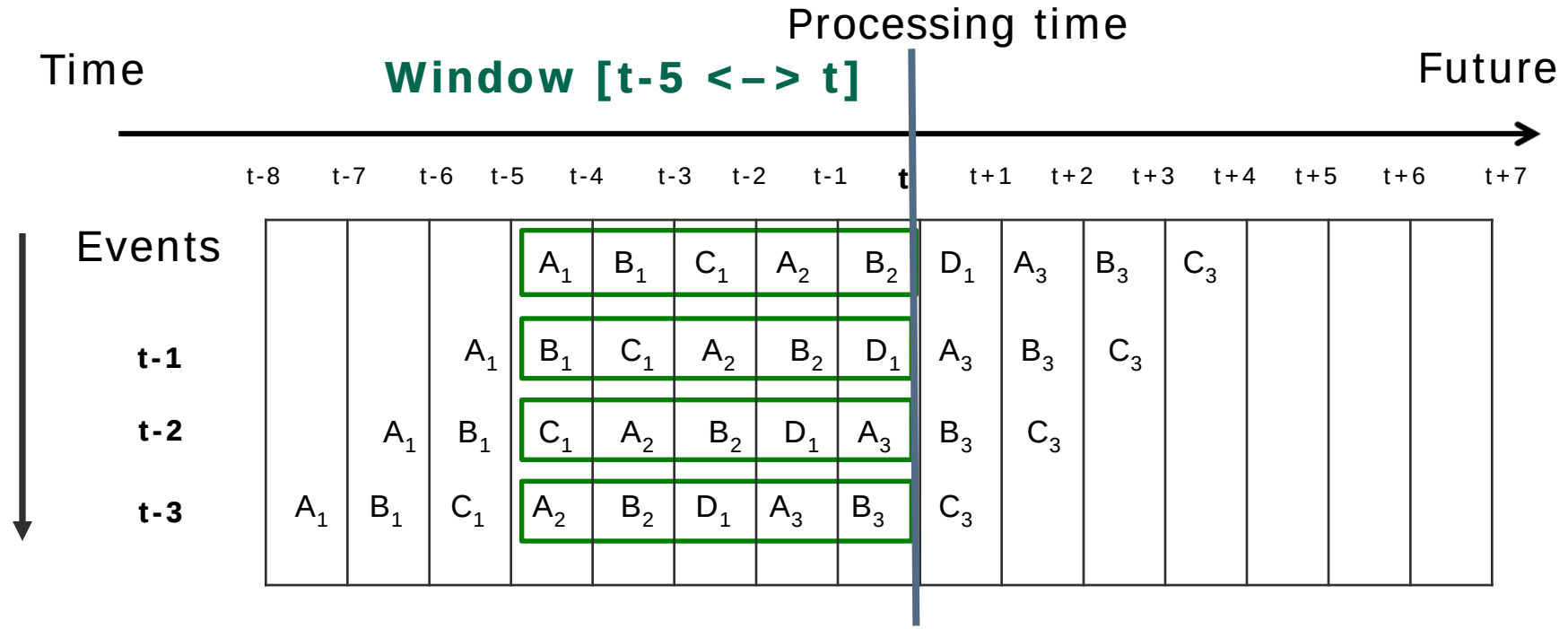
- **Sequence Operator (;):** *(E1;E2)*
- **Disjunction Operator (∨):** *(E1 ∨ E2)* , at least one
- **Conjunction Operator (∧):** (E1 ∧ E2)
- **Simultaneous Operator (=):** *(E1 = E2)*
- **Negation Operator (¬):** *(E1 ∧ ¬ E2)*
- **Quantification (Any):** Any(n) E1, when n events of type E1 occurs
- **Aperiodic Operator (Ap):** *Ap(E2, E1, E3)* , E2 Within E1 & E3
- **Periodic Operator (Per):** Per(t, E1, E2) , every t time-steps in between E1 and E2

Example – Interval-based Sequence

– Example: **D** = *A;(B;C)*



Event Detection Operators & Windowing



Event Detection Pattern [A ; B]

Matches

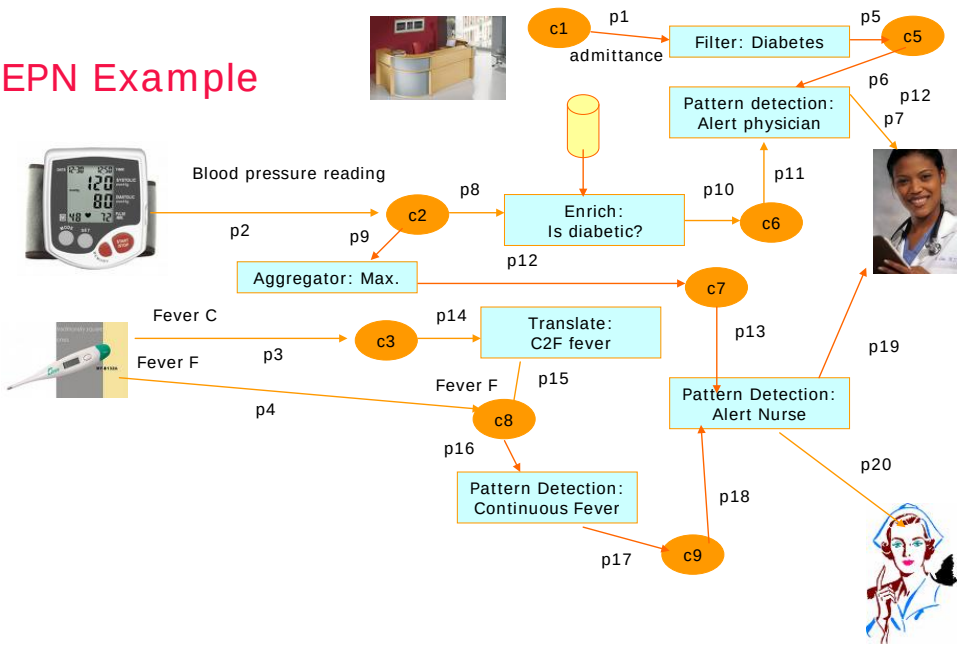
Time t → 2 or 3 (based on event consumption policy)
Time t-1 → 1
Time t-2 → 1
Time t-3 → 2 or 3

Event Processing Agent and Event Processing Network

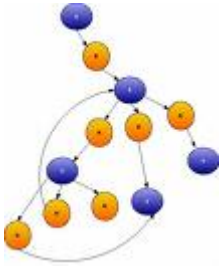
Event Processing Network is a collection of Event Processing Agents.

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

EPN Example



What is the idea ?



Freie Universität Berlin

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

c1

p1

admittance

Filter: Diabetes

p5

c5

p6

p12

p7

Pattern detection:
Alert physician

p11

p10

Enrich:
Is diabetic?

p8

Blood pressure reading

p2

c2

p9

c6

p12

Aggregator: Max.

c7

Fever C

p14

Translate:
C2F fever

p3

c3

p13

p19

Fever F

p15

Fever F

p4

c8

Pattern Detection:
Alert Nurse

p16

p20

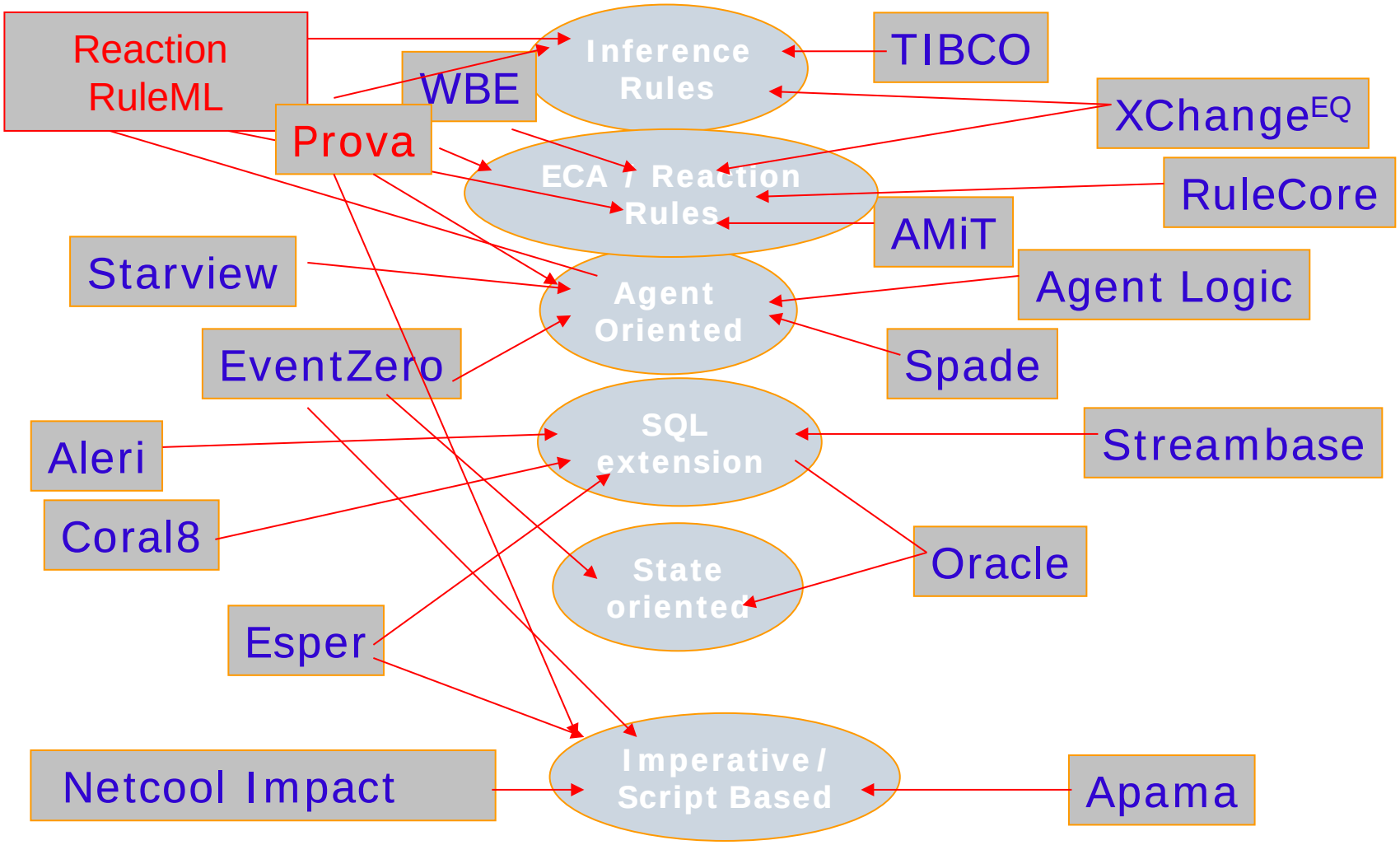
Pattern Detection:
Continuous Fever

p18

p17

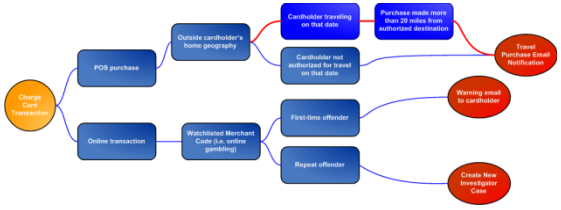
c9

Example Event Processing Languages

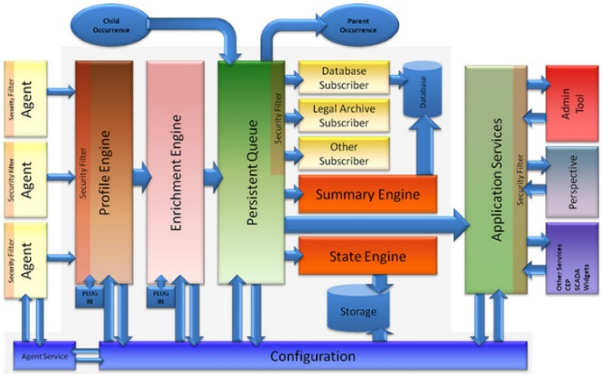


Some EPN, event flow and data flow oriented languages

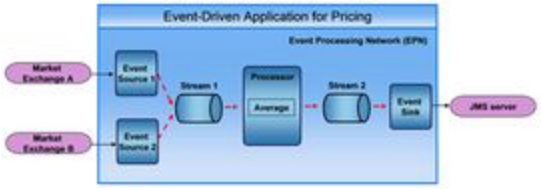
Agent Logic



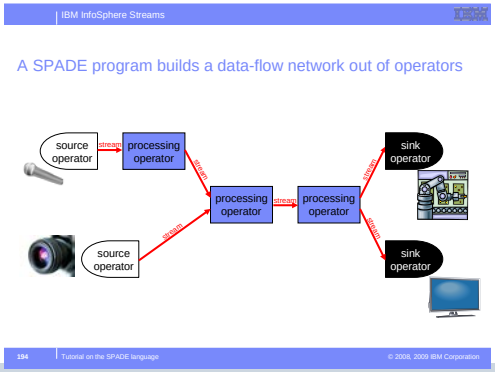
Event Zero



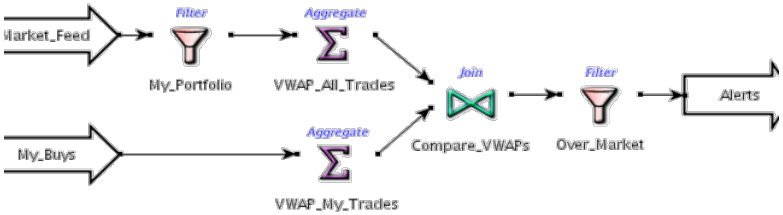
Oracle/BEA



Spade (IBM System S)



Streambase



Example Research Prototypes

Freie Universität Berlin

- ***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

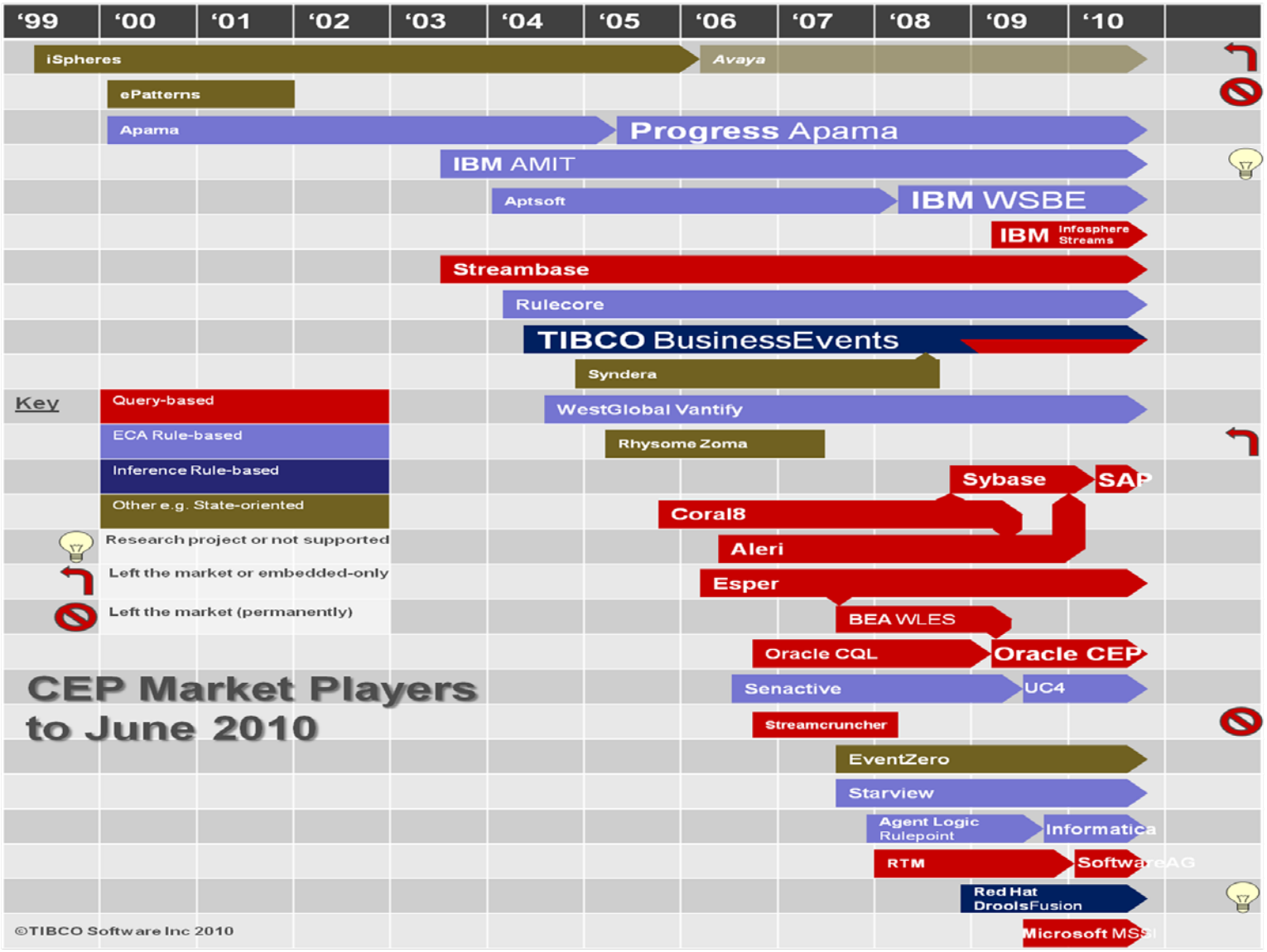
- ***Data Stream Engines, e.g.:***

–Stream CQL, PIPE, TelegraphCQ, Gigascope, Aurora Borealis, SPADE

- ***Stream Reasoning, e.g.:***

–C-SPARQL, EP-SPARQL, SCEPter, SPARQLStream, CQELS

Commercial CEP Market



Source: Tibco Software Inc. 2010

Implementations - Example Systems: TIBCO BusinessEvents

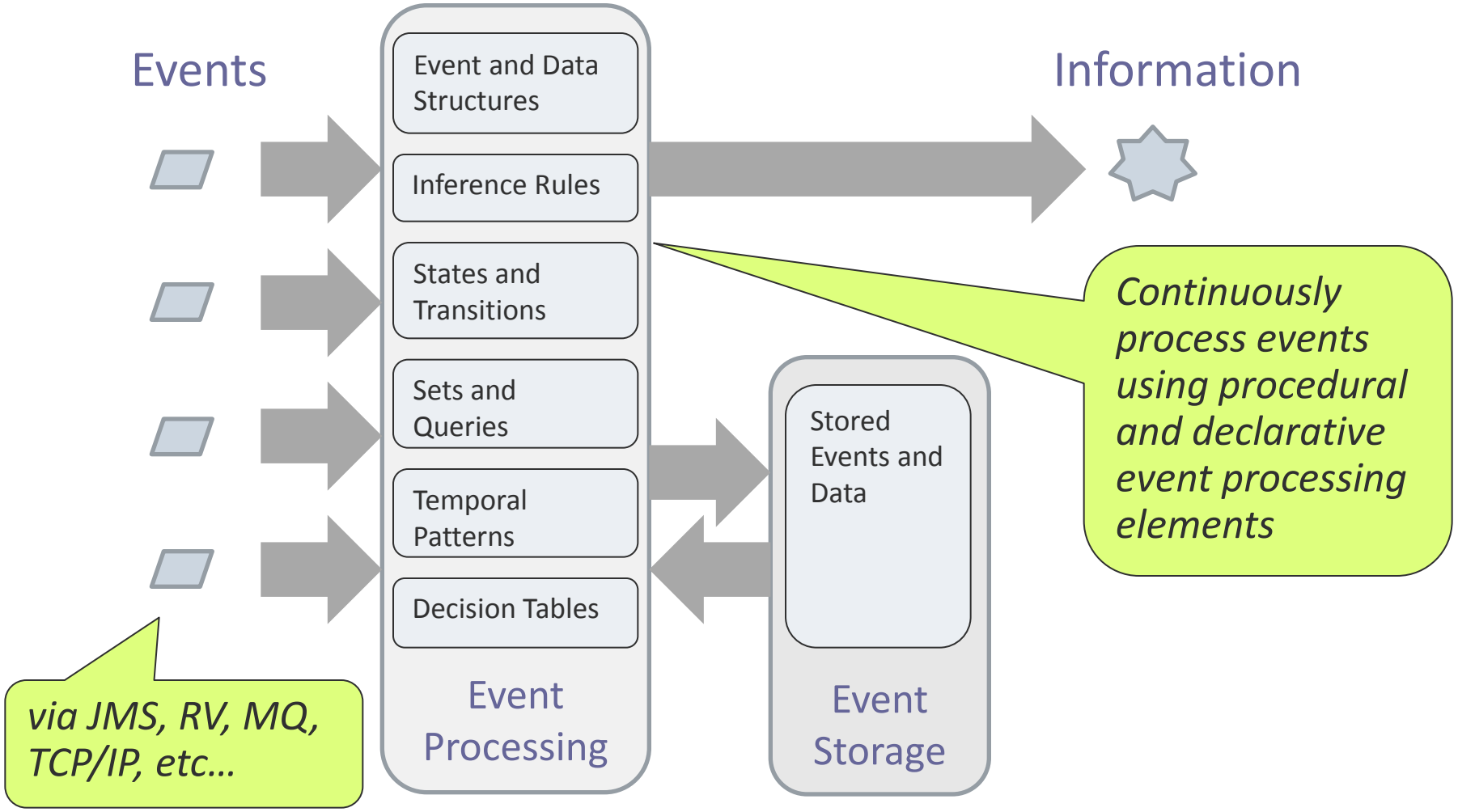
TIBCO BusinessEvents "event server" / event processing engines
Multiple languages / engines in various packaging options
- UML-based event and concept (object) class hierarchical models
- Java-based language
- Eclipse-based IDE

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)

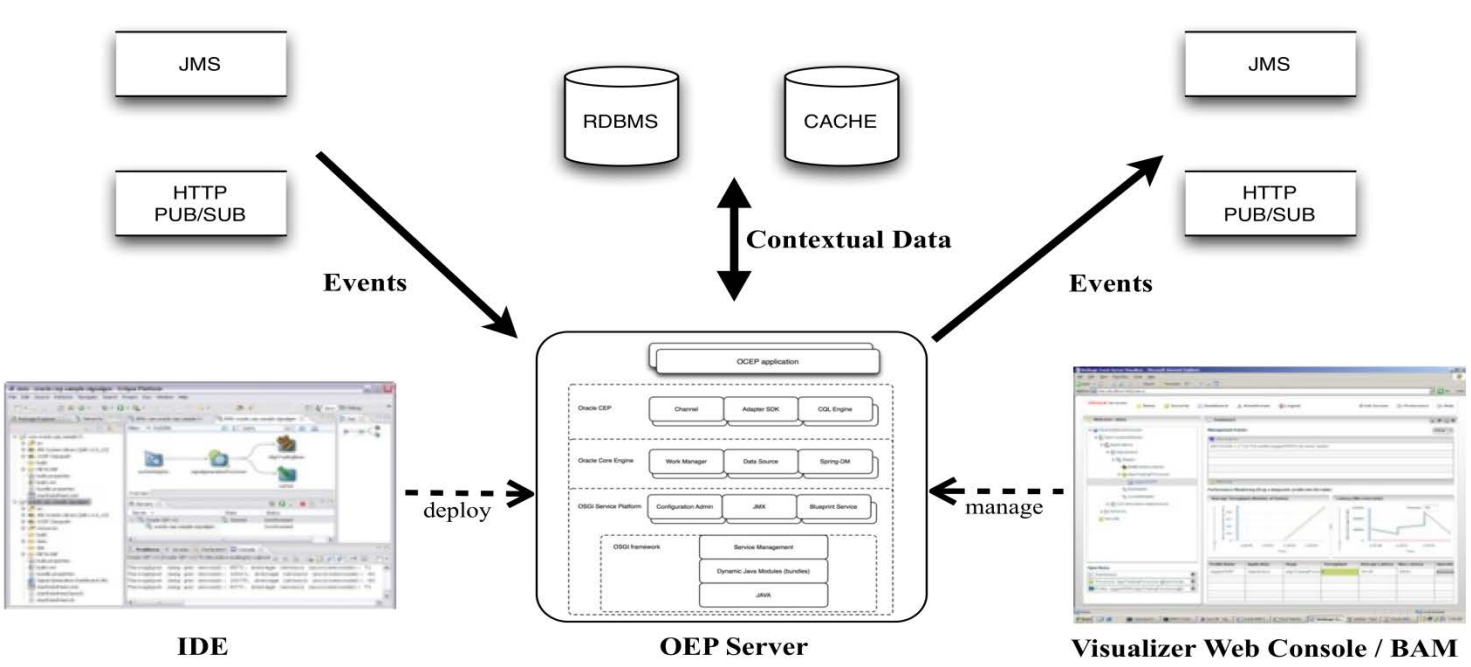


Implementations - Example Systems: Oracle Event Processing (formerly Oracle CEP)

Development platform for event processing applications

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

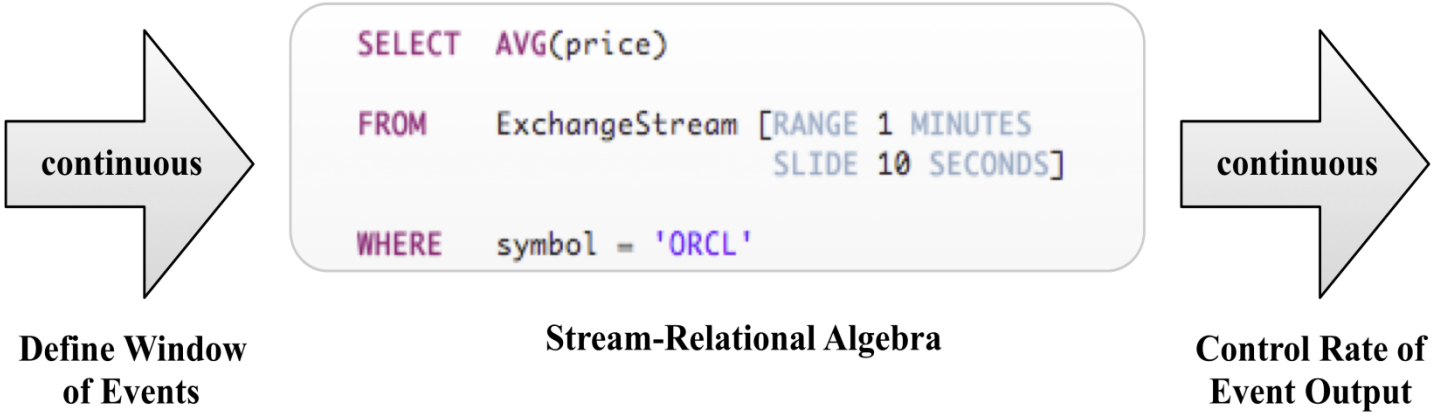
Language is an extension of SQL with stream and pattern matching extensions



Implementations - Example Systems: Oracle Event Processing (formerly Oracle CEP)

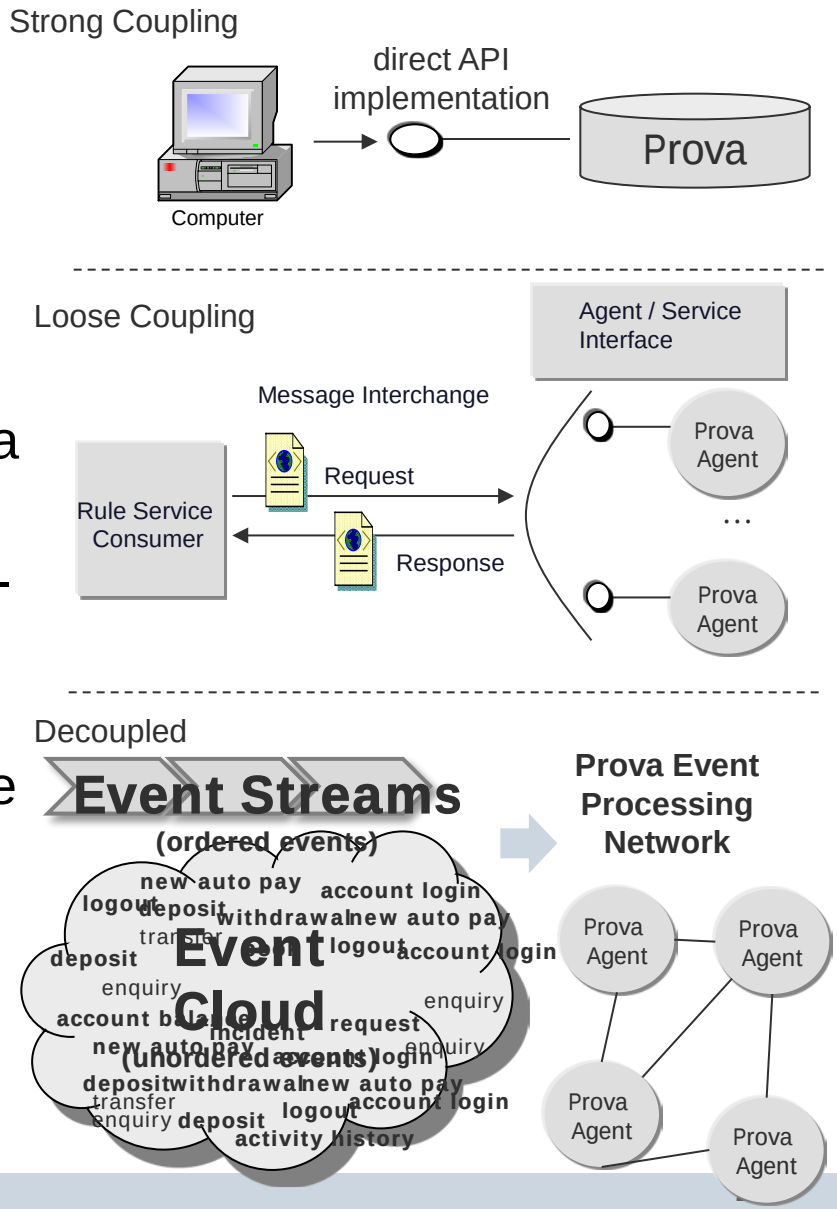
CQL: Continuous Query Language

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



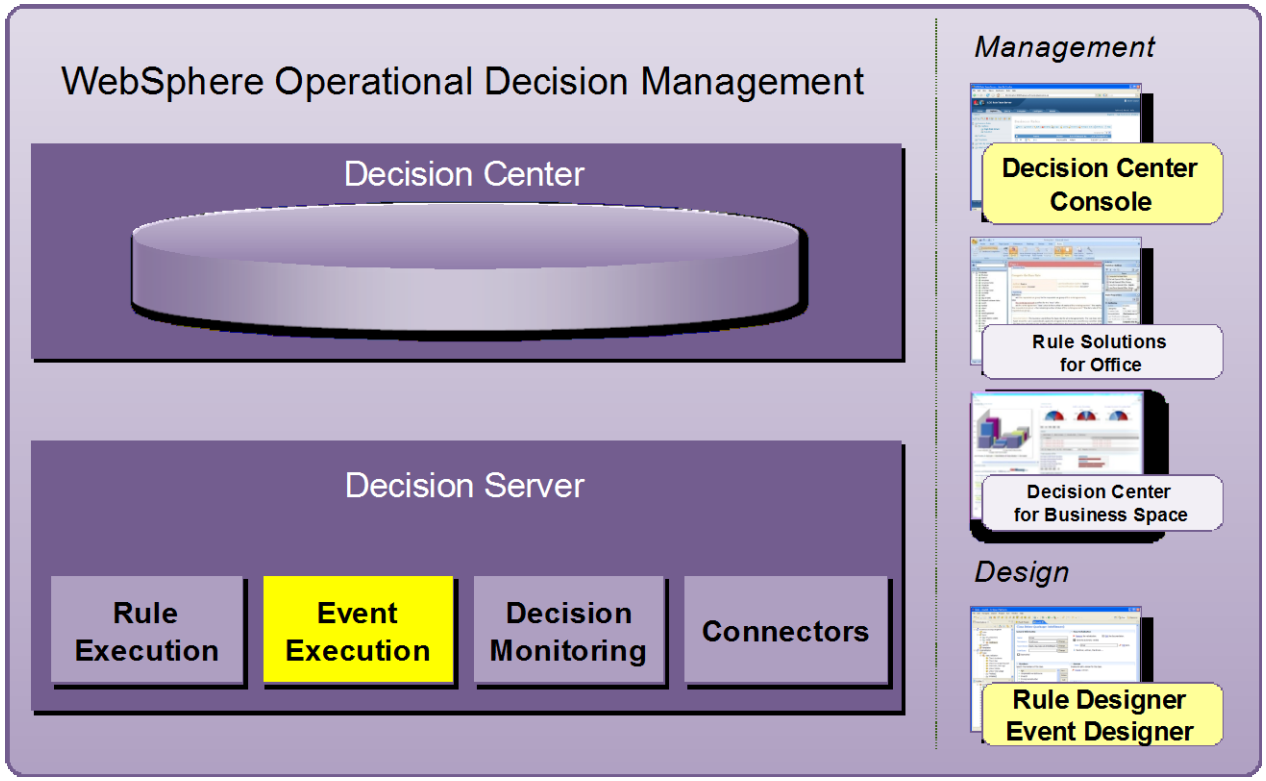
Implementations - Example Systems: IBM WODM Decision Server Events

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

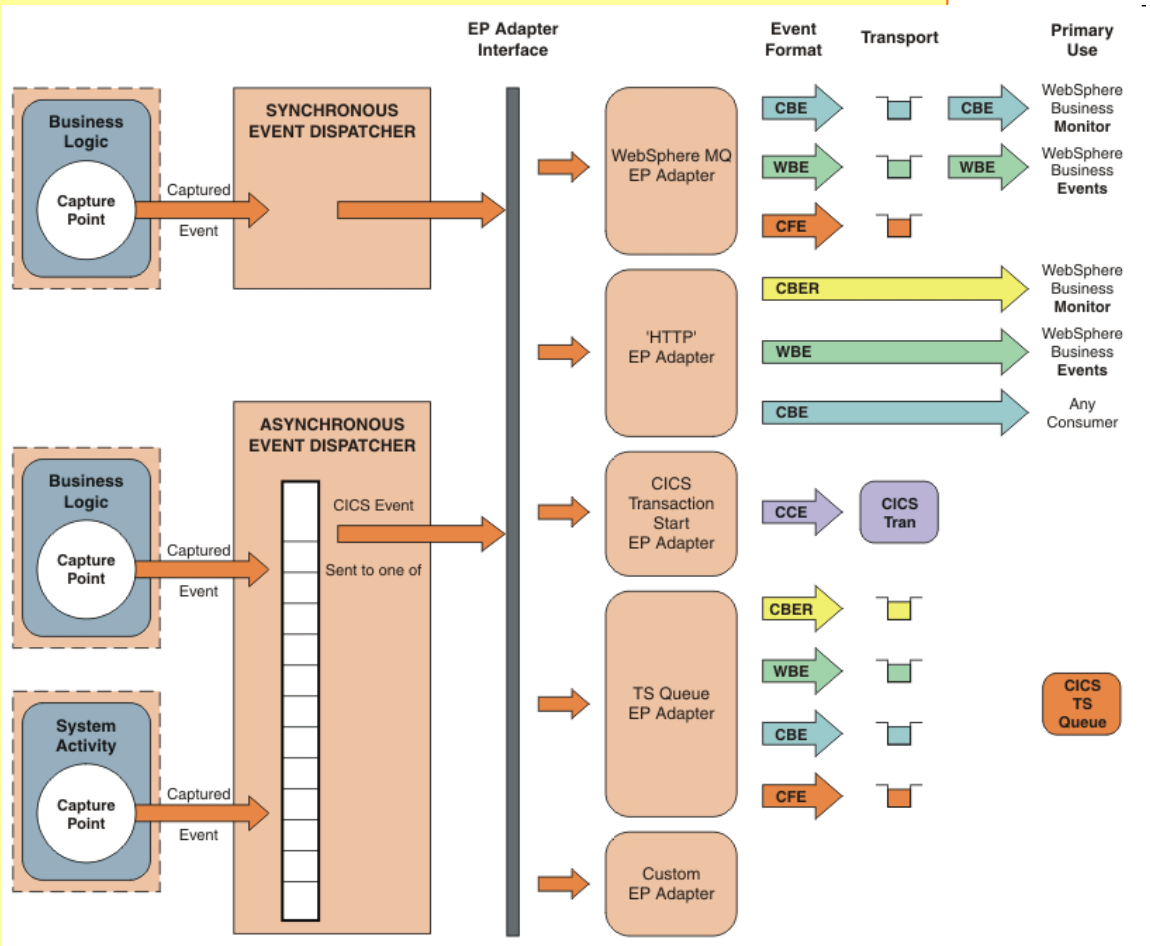
Detect, evaluate, and respond to events

Discover event patterns and initiate actions



Implementations - Example Systems: IBM CICS Transaction Server for z/OS

CICS TS



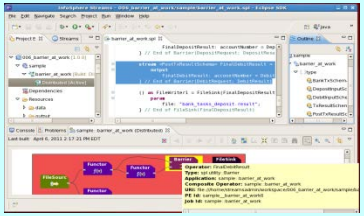
Implementations - Example Systems: IBM InfoSphere Streams – Stream Processing Language

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

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

Streams Processing Language and IDE



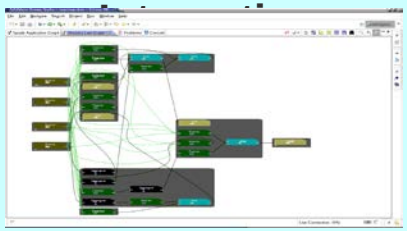
eclipse

Streams Studio
Eclipse IDE for SPL

Runtime Environment

Highly Scalable stream processing runtime

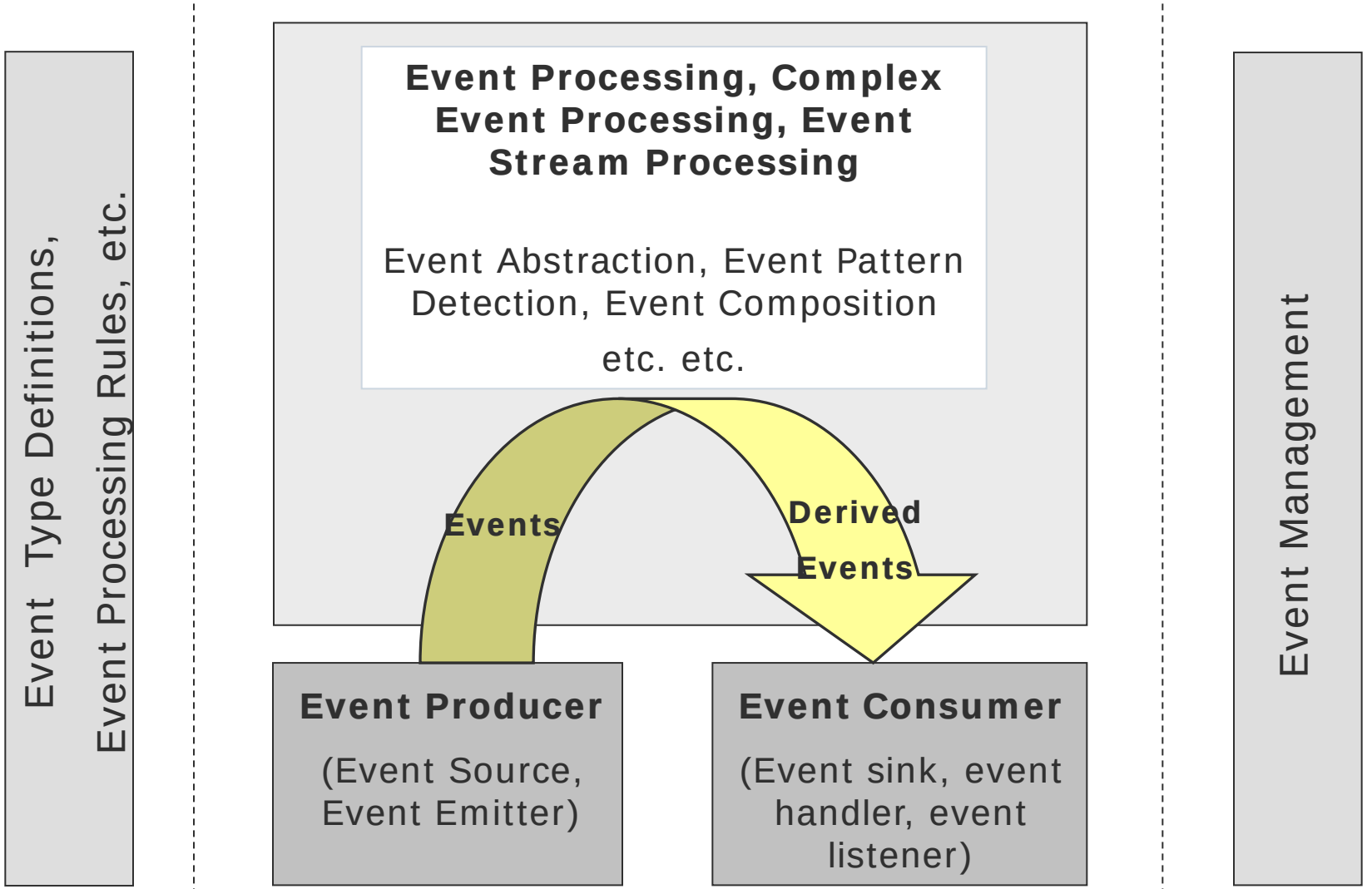
Tools and Technology



DB2 IDS SolidDB WebSphere Front Office 3.0 ORACLE FIX PROTOCOL INDUSTRY-DRIVEN MESSAGING STANDARD Microsoft SQL Server MySQL

Streams Console & Monitoring, Built-in Stream Relational Analytics, Adapters, Toolkits

Functional View source: definitions of EP



EPTS Reference Architecture - Functional View

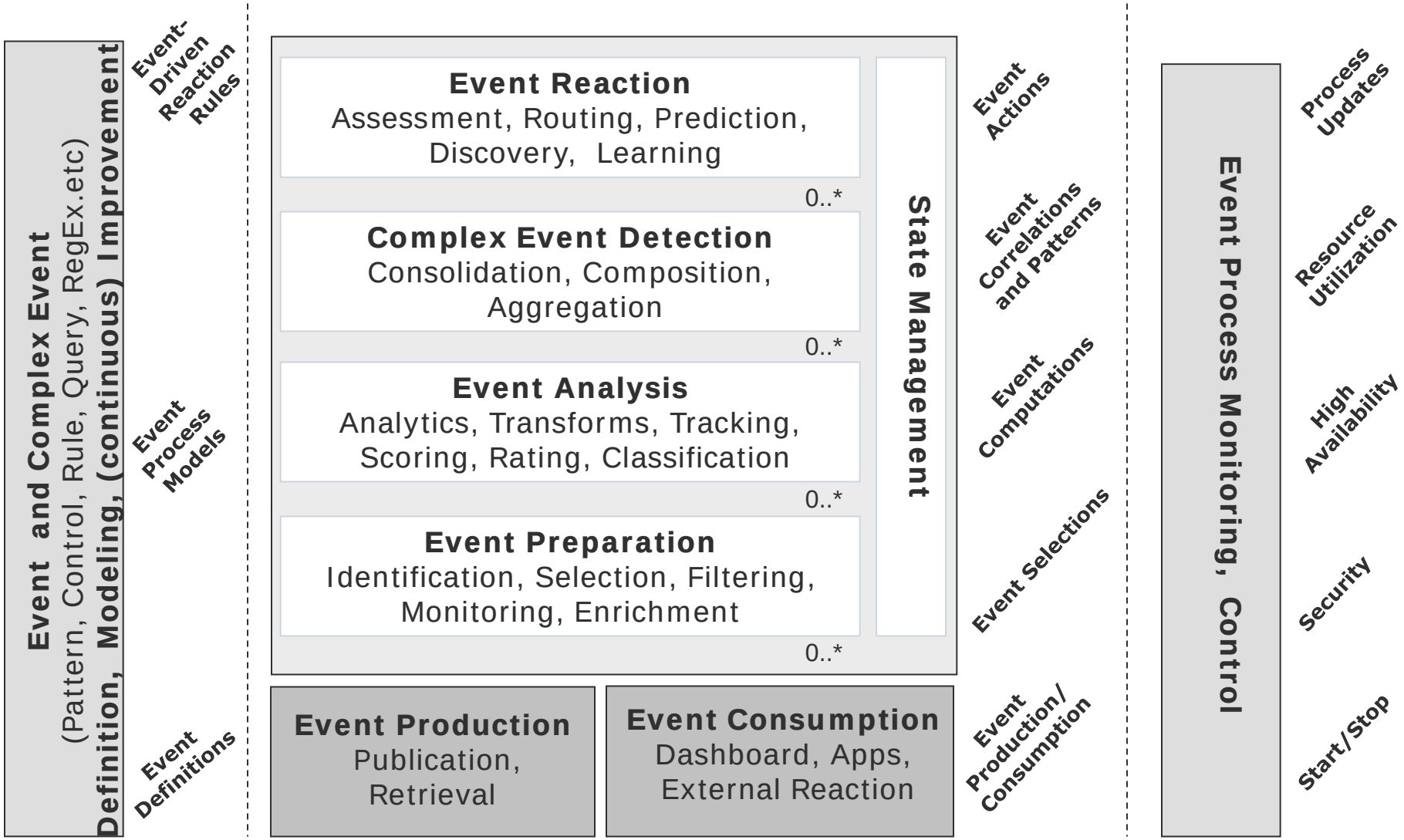
Architect and Developer perspective

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

Run-time functions in 2 main groups:

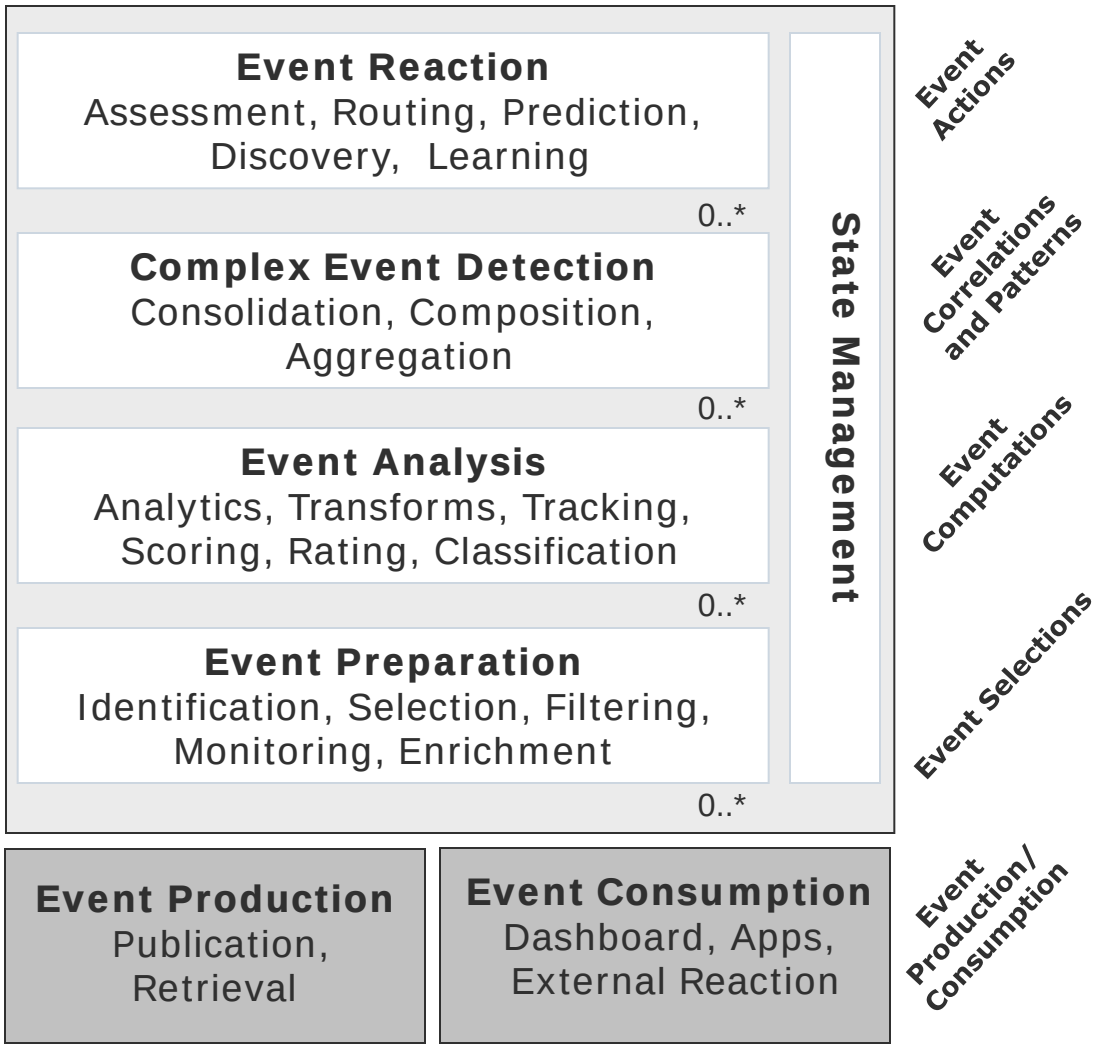
- the event infrastructure (sources and consumers) external to the event processor under consideration,
- the event processor.

Reference Architecture: Functional View



Design time **Run time** **Administration**

Reference Architecture: Functional View / Runtime



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.

Reference Architecture: Functional View / Runtime

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 Preparation
Identification, Selection, Filtering, Monitoring, Enrichment

Reference Architecture: Functional View / Runtime

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.

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

Reference Architecture: Functional View / Runtime

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.

Complex Event Detection
Consolidation, Composition, Aggregation

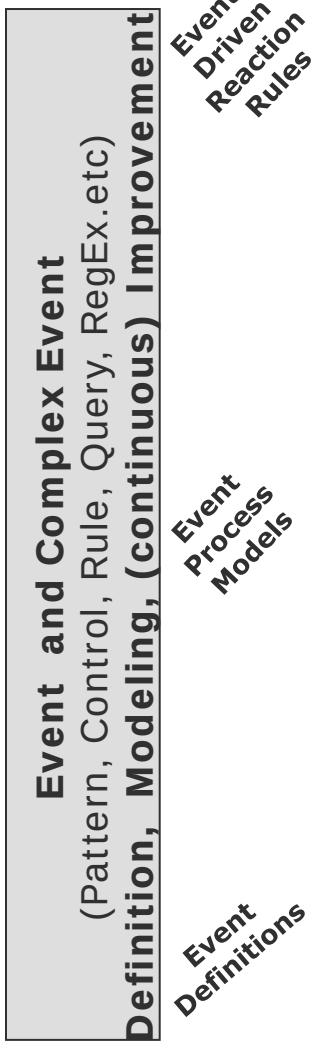
Reference Architecture: Functional View / Runtime

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)

Design time

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.

Administration

Summary - Event Processing Patterns

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

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

So, what have we learned today?

Freie Universität Berlin

- **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**

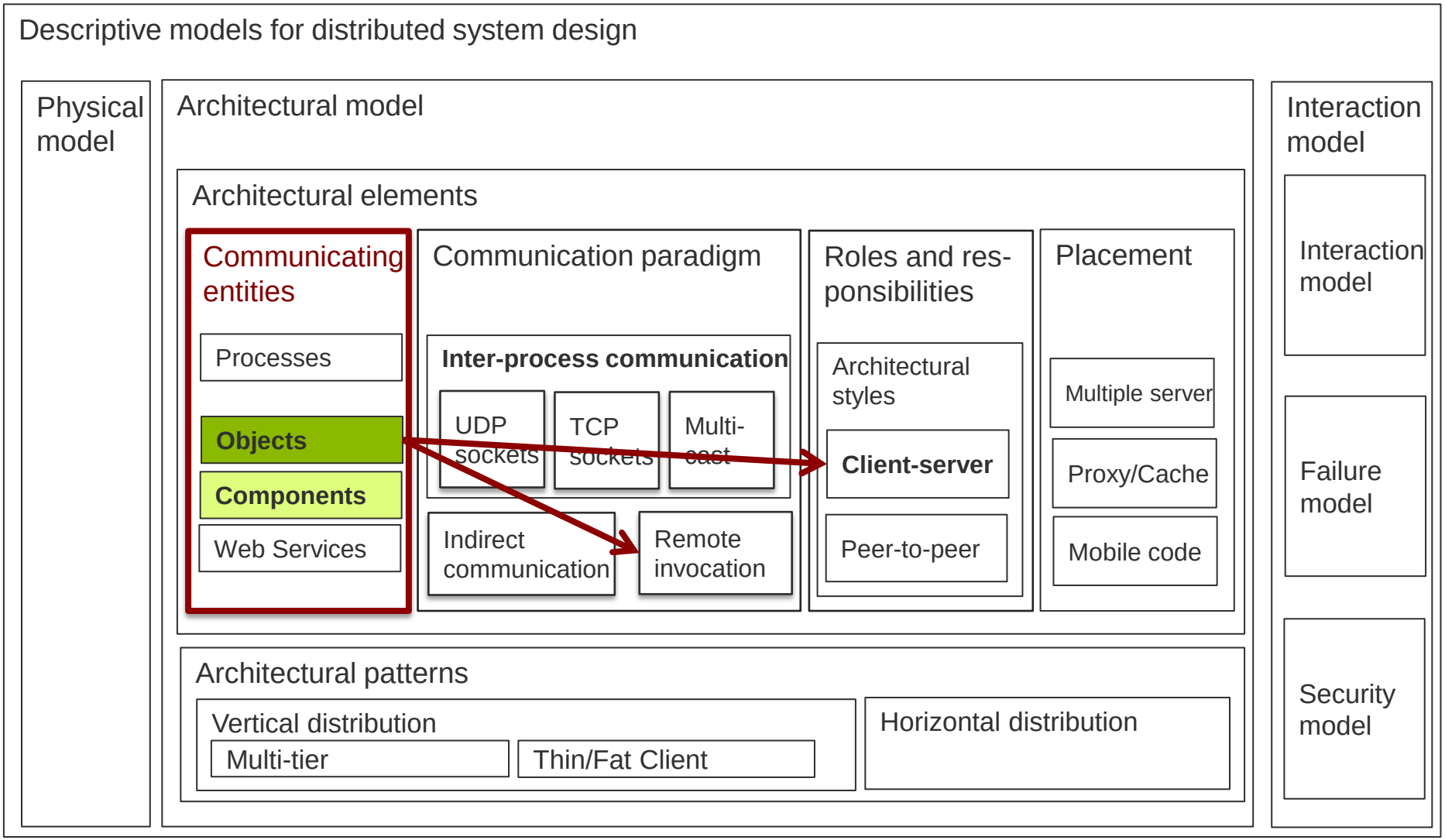
- **Overview CEP Languages**
  - **Rule-based, agent-based, SQL extensions, state-based, imperative, scripting**

- **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



Freie Universität Berlin

Next class

Distributed Objects and Components

References

David C. Luckham. 2001. *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

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

Adrian Paschke, Paul Vincent, Florian Springer: Standards for Complex Event Processing and Reaction Rules. RuleML America 2011: 128-139
http://link.springer.com/chapter/10.1007%2F978-3-642-24908-2_17
http://www.slideshare.net/isvana/ruleml2011-cep-standards-reference-model

Francois Bry, Michael Eckert, Opher Etzion, Adrian Paschke, Jon Ricke: Event Processing Language Tutorial, DEBS 2009, ACM, Nashville, 2009
http://www.slideshare.net/opher.etzion/debs2009-event-processing-languages-tutorial

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