

DemAAL Summer School 2013

September 16- 20th, Chania, Crete

Semantic CEP

Prof. Dr. Adrian Paschke

Corporate Semantic Web (AG-CSW)
Institut für Informatik, Freie Universität Berlin
paschke@inf.fu-berlin
<http://www.inf.fu-berlin/groups/ag-csw/>



Agenda

- **Introduction to Event Processing**
- **Semantic Complex Event Processing (SCEP)**
- **Event Processing Technical Society (EPTS)**
 - Event Processing Standards Reference Model
 - Event Processing Reference Architecture
- **Event Processing Function Patterns - Examples**
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- **Reaction RuleML Standard**
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- **Summary**

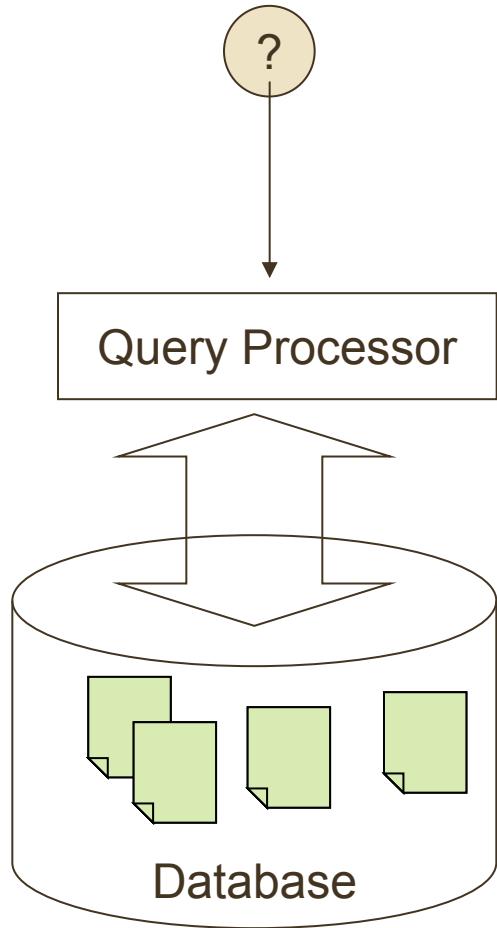
Agenda

- **Introduction to Event Processing**
- **Semantic Complex Event Processing (SCEP)**
- **Event Processing Technical Society (EPTS)**
 - Event Processing Standards Reference Model
 - Event Processing Reference Architecture
- **Event Processing Function Patterns - Examples**
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- **Reaction RuleML Standard**
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- **Summary**

Event Processing vs. Databases

Ex-Post Data Queries

Database Queries



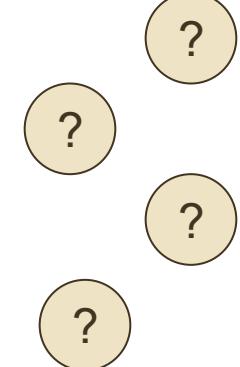
Real Time Data Processing

Incoming Events

Time
Future

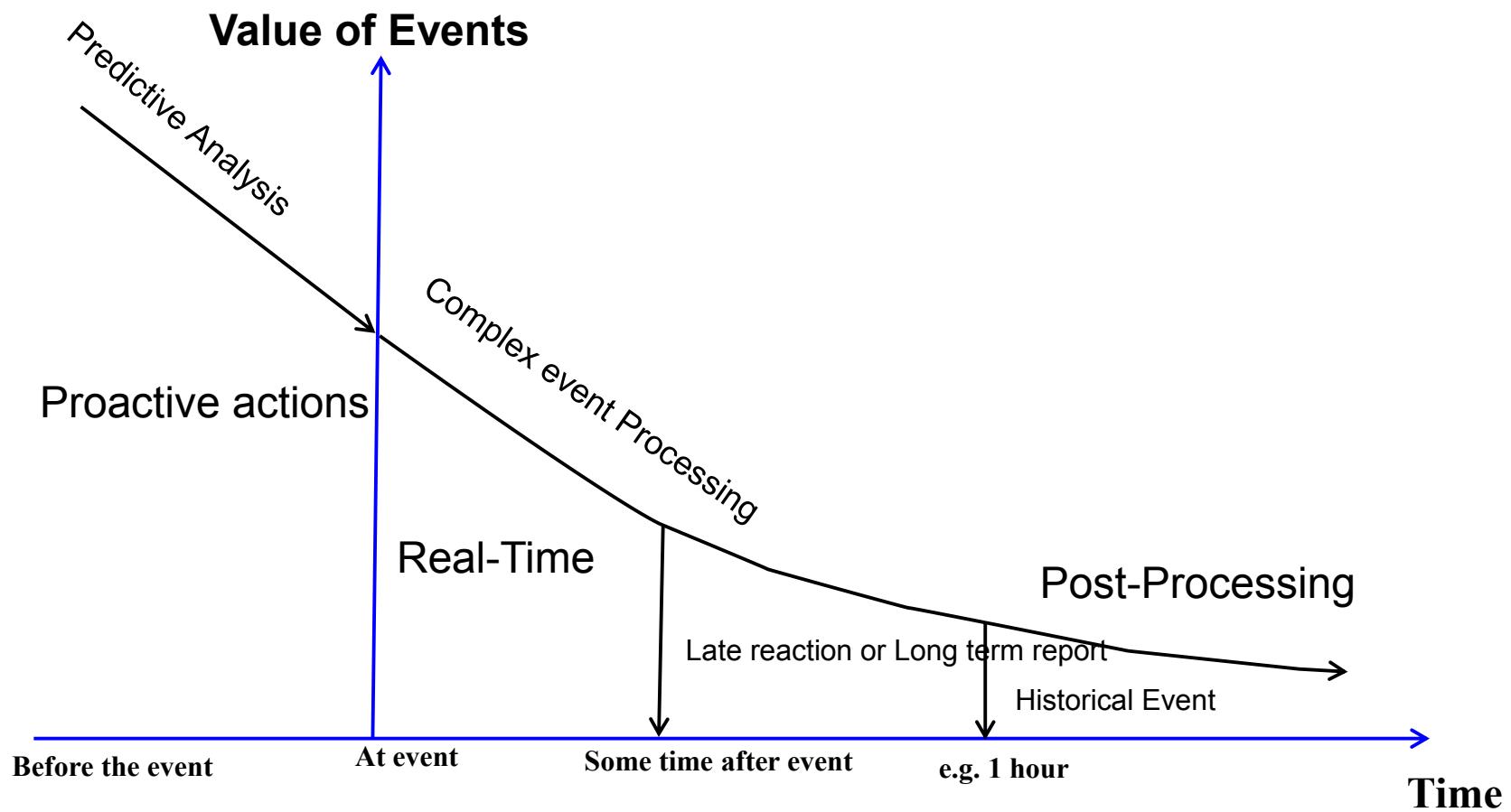
Processing
moment of
events

Event
Subscriptions



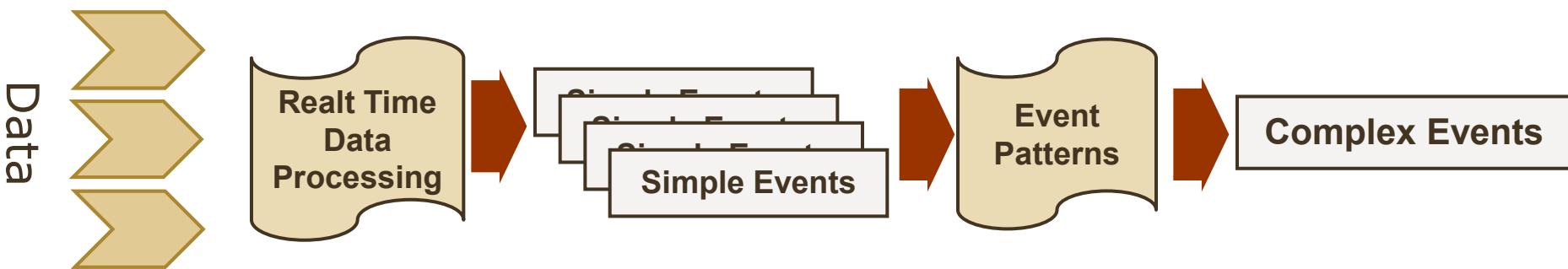
past

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 behaviors

Complex Event Processing – What is it?

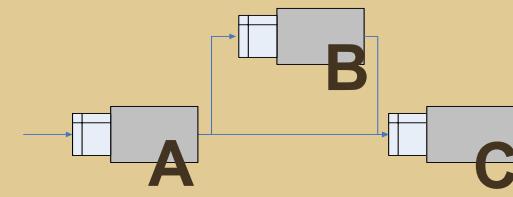
- CEP is about complex event detection and reaction to complex events
 - Efficient (near real-time) processing of large numbers of events
 - Detection, prediction and exploitation of relevant complex events
 - Supports situation awareness, track & trace, sense & respond



Continuous Event Queries

```
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;
```

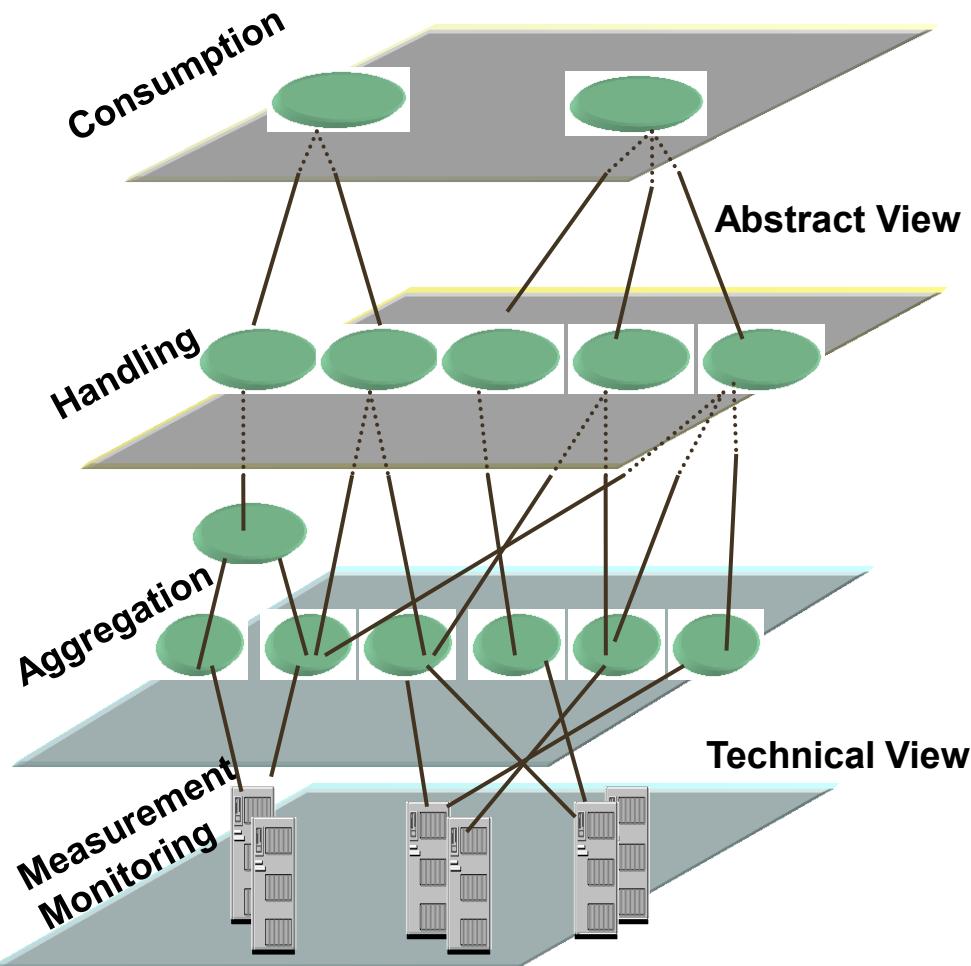
Patterns, Rules



Complex Events

Complex Event Processing – What is it?

Complex Event Processing Media



- 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

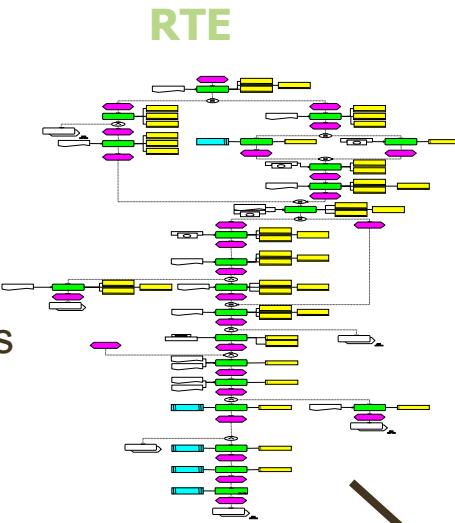
Complex Events – How?

Example Event Algebra Operators:

- **Sequence Operator (\cdot):** $(E1;E2)$
- **Disjunction Operator (\vee):** $(E1 \vee E2)$, at least one
- **Conjunction Operator (\wedge):** $(E1 \wedge E2)$
- **Simultaneous Operator ($=$):** $(E1 = E2)$
- **Negation Operator (\neg):** $(E1 \wedge \neg E2)$
- **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

CEP – Why do we need it? Example Application Domains

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



Valuable Information at the Right Time to the Right Recipient



Information Dissemination

CEP
Media
Detect Decide Respond

BAM, ITSM



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

Enterprise Decision Management



Expert Systems

Expert Decision Management

Basic Definitions* (1)

* Paschke, A., Boley, H.: Rules Capturing Event and Reactivity, in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, ISBN:1-60566-402-2, 2009
<http://www.igi-global.com/book/handbook-research-emerging-rule-based/465>
<http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=16435934&article=0&fd=pdf>

- **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 Definitions* (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

Classification of the Event Space* (1)

* Reaction RuleML Classification of the
Event/Action/State Processing and Reasoning Space
<http://arxiv.org/ftp/cs/papers/0611/0611047.pdf>

- **Event Processing**

- **Atomic vs. complex event processing:** only support for atomic events or computation of complex events, i.e. combinations of several atomic (and complex) events
- **Short term vs. long term:**
 - *Short term- immediate reaction*: transient, non-persistent, real-time selection and consumption of events (e.g. triggers, ECA rules)
 - *Long term- retrospective, deferred, or prospective*: Persistent events, typically processed in retrospective e.g. via KR event calculi reasoning or event algebra computations on a event instance history; but also prospective planning / proactive, e.g. KR abductive planning
- **Separation of event definition, selection and consumption:**
 - Event definition: algebraic, temporal logic, event/action calculus, ...
 - Event selection (from instance sequence): first, last, all, ...
 - Event consumption: consume once, do not consume, ...
- **Deterministic vs. non-deterministic:** simultaneous occurred events trigger more than one rule and give rise to only one model (deterministic, e.g. by priority-based selection of rules) or two or more models (non-deterministic)
- **Active vs. passive:** actively query / compute / reason / detect events (e.g. via monitoring, querying / sensing akin to periodic pull model or on-demand retrieve queries) vs. passively listen / wait for incoming events or internal changes (akin to push models e.g. publish-subscribe)
- **Local vs. global:** events are processed locally within a context (e.g. a process, conversation, branch) or globally (apply global on all rules)

Classification of the Event Space* (2)

• Event Type

- **Flat vs. semi-structured compound data structure/type**, e.g. simple flat representations (e.g. propositional) or complex objects with or without attributes, functions and variables
- **Primitive vs. complex**, e.g. atomic, raw event types or complex event types
- **Homogenous vs. heterogeneous** Subevents are different from the complex event
- **Temporal:**
 - absolute (e.g. calendar dates, clock times),
 - relative/delayed (e.g. 5 minutes after ...),
 - durable (occurs within an interval),
 - durable with continuous, gradual change (e.g. clocks, countdowns, flows)
- **State or Situation:**
 - without explicit boundaries
 - State (e.g. “the fire alarm stopped”): selective or interval-oriented, homogenous (the state holds in all subintervals and points), without dynamic change
 - Process (e.g. “he runs”): interval-oriented, homogenous (the process is executed in sub-intervals), continuous dynamic change
 - Iterative event (e.g. “the alarm rings again and again”): selective or interval-oriented, homogenous
 - with explicit boundaries
 - dynamic change (e.g. the light changes the colour)
 - interval-based (within the next 5 minutes)
 - frequency (e.g. the telephone rung three times)
- **Spatio / Location:** durable with continuous, gradual change (approaching an object, e.g. 5 meters before wall, “bottle half empty”)
- **Knowledge Producing:** changes agents/systems knowledge/belief and not the state of the external world, e.g. look at the schedule → internal effect, e.g. **belief update and revision**

Classification of the Event Space* (3)

- **Event Source**

- **Implicit** (changing conditions according to self-updates) vs. **explicit events** (e.g. production rules vs. ECA rules)
- **By request** (query on database/knowledge base or call to external system) vs. **by trigger** (e.g. incoming event message, publish-subscribe, agent protocol / coordination)
- **Internal database, KB update events** (e.g. add, remove, update, retrieve) or **external explicit events** (inbound event messages, events detected at external entities)
- **Generated, produced** (e.g. phenomenon, derived action effects) vs. **occurred** (e.g. detected or received event)

Classification of the Condition Space*

- **Logical conditions vs. pattern-based test constraints:**
 - logical conditions act as goals on complex conditional logic represented in terms of (derivation) rules as in e.g. in backward-reasoning logic programming; might support e.g. variables (new free and bound event variables) + backtracking over different variable bindings, logical connectives (conjunction, disjunction and negations)
 - ground pattern tests e.g. on changed conditions (~ implicit events in production rules) and pattern matching tests
- **Conditions on the state before/after/intermediate the action change**
- **Scoped conditions: a scoped condition only applies on an explicitly defined scope (e.g. part of the knowledge base (e.g. a module) or working memory**
- **External data integration: conditions can define constructive views (queries) over external data sources and bind values/objects to variables**

Classification of the Situation Space* (1)

- ***Situation***
 - A situation is initiated or terminated by one or more (complex) events, i.e. complex event + conditional context
- **Situations as a narrow and closed whole without addressing the elapsed time**
 - heterogeneous: interior situations differ from the overall situation
 - Sub-types
 - Dynamic change
 - e.g. the traffic light changes the color
 - With time frame
 - e.g. within 5 minutes
 - Frequency
 - e.g. it rings three times
- **Situation in the flow/process, without addressing the narrowness / closeness**
 - homogeneous: interior situations are similar to the general situation
 - Sub-types
 - State
 - e.g. *he lays on the floor*
 - Process
 - e.g. *he runs*
 - Iterative process
 - e.g. *he coughs (again and again)*
 - Habitual process
 - e.g. *he smokes*

Classification of the Action Space* (1)

- **Action Processing**

- **Atomic vs. complex action processing:** only support for atomic action or execution of complex actions (e.g. complex workflows, execution paths)
- **Transactional vs. non-transactional processing:** Transactional complex actions possibly safeguarded by post-conditional integrity constraints / test case tests might be rolled-back in case of failures or committed
- **Hypothetical vs. real execution:** actions have a real effect on the internal (add new fact) or external world, or are preformed hypothetically, i.e. the effect is derive or computed implicitly
- **Concurrent vs. sequential execution:** actions can be processed concurrently (e.g. in parallel branches / threads)
- **Local vs. global execution:** actions are executed in a global context or locally in a (workflow) process, scope or conversation station (e.g. outbound action messages in conversations).
- **Variable iteration:** actions iterate over variable bindings of event and conditional variables including possible backtracking to binding variants

Classification of the Action Space* (2)

- **Action Type**
 - **Internal knowledge self-update actions** with internal effects on extensional KB or working memory (update facts / data) and intensional KB / rule base (update rules)
 - **State actions** with effects on changeable properties / states / fluents, often actions directly relate to events and are treated similar (as e.g. in KR event and fluent calculi)
 - **External actions** with side effects on external systems via calls (procedural attachments), outbound messages, triggering/effecting
 - **Messaging actions**: outbound action messages, usually in a conversational context
 - **Process actions**: process calculi and workflow pattern style actions such as join, merge, split, etc.
 - **Complex actions** (e.g. delayed reactions, actions with duration, sequences of bulk updates, concurrent actions, sequences of actions) modelled by e.g. action algebras (~event algebras) or process calculi

Classification of the Event / Action / State Processing* (1)

- **1. Event/Action Definition Phase**
 - Definition of event/action pattern by event algebra
 - Based on declarative formalization or procedural implementation
 - Defined over an atomic instant or an interval of time or situation context.
- **2. Event/Action Selection Phase**
 - Defines selection function to select, e.g. “*first*”, “*last*”, event from several occurred events (stored in an event instance sequence e.g. in memory, database/KB, queue) of a particular type
 - Crucial for the outcome of a reaction rule, since the events may contain different (context) information, e.g. different message payloads or sensing information
 - KR view: Derivation over event/action history of happened or future planned events/actions

Classification of the Event / Action / State Processing* (2)

- **3. Event/Action Consumption / Execution Phase**

- Defines which events are consumed after the detection of a complex event
- An event may contribute to the detection of several complex events, if it is not consumed
- Distinction in event messaging between “multiple receive” and “single receive”
- Events which can no longer contribute, e.g. are outdated, should be removed
- KR view: events/actions are not consumed but persist in the fact base

- **4. State / Transition Processing**

- Actions might have an internal effect i.e. change the knowledge state leading to state transition from (pre)-condition state to post-condition state.
- The effect might be hypothetical (e.g. a hypothetical state via a computation) or persistent (update of the knowledge base),
- Actions might have an external side effect
- Inactive (complex) actions should be removed

Core CEP Operations



The Many Roots of CEP...

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

Active Databases

Agents

Distributed Event-based Computing

Event-driven Reaction Rules

Discrete event simulation

High performance databases

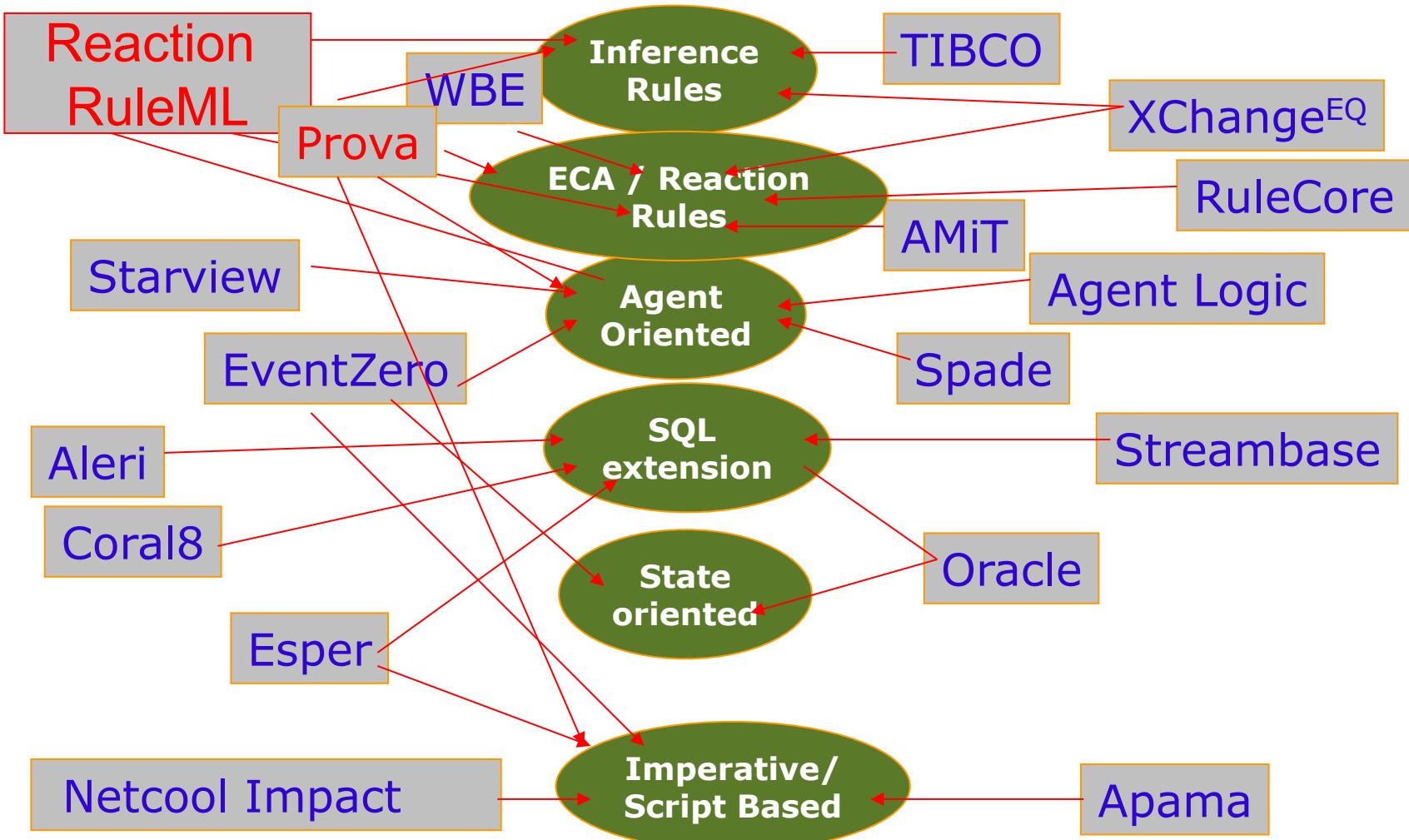
**Detect
Decide
Respond**

Event-based Workflows / Business Processes

Messaging / Middleware



Example Event Processing Languages



Commercial CEP Market



**CEP Market Players
to June 2010**

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

Agenda

- Introduction to Event Processing
- **Semantic Complex Event Processing (SCEP)**
- Event Processing Technical Society (EPTS)
 - Event Processing Standards Reference Model
 - Event Processing Reference Architecture
- Event Processing Function Patterns - Examples
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- Reaction RuleML Standard
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- Summary

Semantic CEP: The Combination

**(Complex) Event Processing: events,
complex events, event patterns, ...**

+

**Semantic technologies: rules &
ontologies**

Motivation for Semantic CEP

- **Today's environment:**

- High *complex processes, events, ...*
- Existing domain background knowledge
- Missing systems for detection of events based on available background knowledge.

- **Many event processing use cases require:**

- High expressiveness
- High flexibility
- Simplicity

Semantic CEP

- The use of **Background Knowledge** in Complex Event Processing (**CEP**) offers **declarative and expressive description** of complex **events/situations and reactions**.
- It enhances CEP systems to more ***agile*** systems and improves ***flexibility*** to dynamic changes.

Benefits of Semantics in CEP

- What are **benefits** of using Semantic Technology in Complex Event Processing?
 - **Expressiveness:** It can precisely express complex events, patterns and reactions which can be directly translated into operations.
 - **Flexibility:** Changes can be integrated into CEP systems in a fraction of time.
 - Complex Event Patterns are **declaratively represented** and are defined based on abstract strategies.

Semantic Technologies for Declarative Knowledge Representation

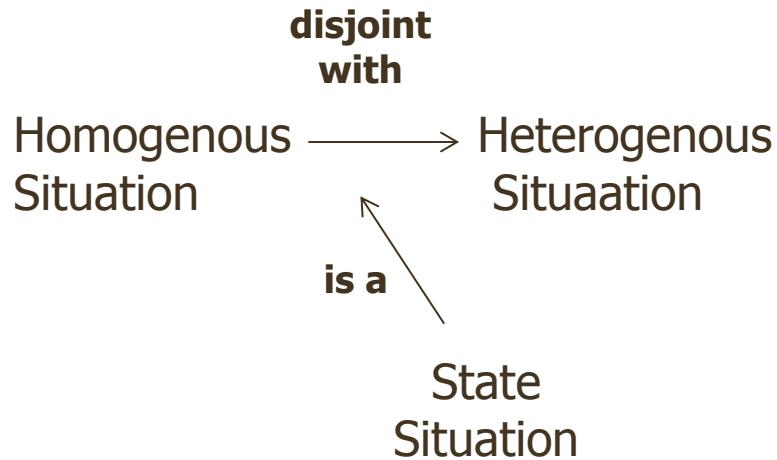
1. Rules

- Describe derived conclusions and reactions from given information (inference)

`takeoff(x) → fly(x)`
`takoff(flight123)`

2. Ontologies

- Ontologies described the conceptual knowledge of a domain (concept semantics):



Usage of Rules

- 1. Rules** that influence the operational processing & decision processes of an event processing agent:
 - ***Derivation rules*** (deduction rules): establish / derive new information that is used, e.g. in a decision process (e.g. routing).
 - ***Reaction rules*** that establish when certain activities should take place :
 - ***Condition-Action rules (production rules)***
 - ***Event-Condition-Action (ECA) rules + variants*** (e.g. ECAP).
 - ***Messaging Reaction Rules*** (event message reaction rules)
- 2. Constraints** on event processing agent's structure, behavior and information:
 - ***Structural constraints*** (e.g. deontic assignments).
 - ***Integrity constraints and state constraints***
 - ***Process and flow constraints***

Reaction Rules* (1)

- **1. (Temporal) KR event/action logics**

- Members e.g. Event Calculus, Situation Calculus, Fluent Calculus, TAL
 - Actions with effects on changeable properties / states / fluents, i.e. actions ~ events
 - Focus: reasoning on effects of events/actions on knowledge states and properties

- **2. KR evolutionary transaction, update, state transition logics**

- Members e.g. transaction logics, dynamic LPs, LP update logics, transition logics,
 - Knowledge self-updates of extensional KB (facts / data) and intensional KB (rules)
 - Transactional updates possibly safeguarded by post-conditional integrity constraints / tests
 - Complex actions (sequences of actions)
 - Focus: declarative semantics for internal transactional knowledge self-update sequences (dynamic programs)

- **3. Condition-Action / Production rules**

- Members, e.g. OPS5, Clips, Jess, JBoss Rules/Drools, Fair Isaac Blaze Advisor, ILog Rules, CA Aion, Haley, ESI Logist, Reaction RuleML
 - Mostly forward-directed non-deterministic operational semantics for Condition-Action rules
 - Focus: primitive update actions (assert, retract); update actions (interpreted as implicit events) lead to changing conditions which trigger further actions, leading to sequences of triggering production rules

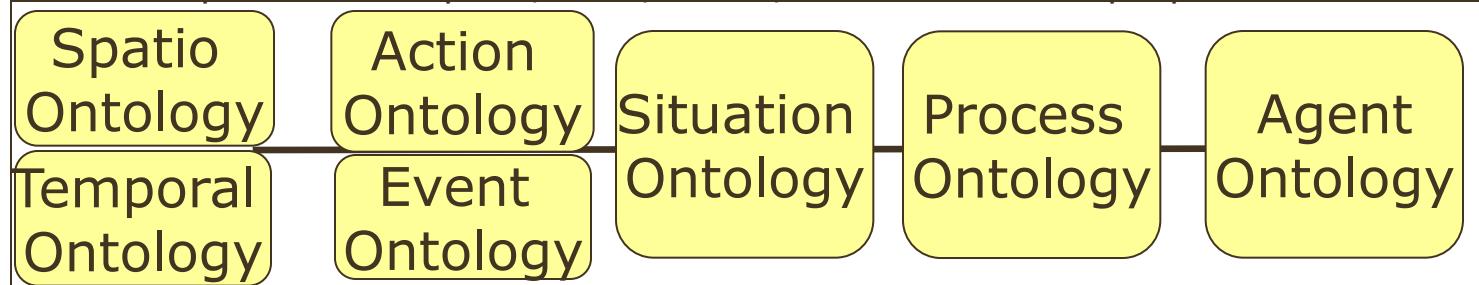
Reaction Rules* (2)

- **4. Active Database ECA rules**
 - Members, e.g. ACCOOD, Chimera, ADL, COMPOSE, NAOS, HiPac, Reaction RuleML, Prova, XChange
 - ECA paradigm: “*on Event when Condition do Action*”; mostly operational semantics
 - Instantaneous, transient events and actions detected according to their detection time
 - Focus: Complex events: event algebra (e.g. Snoop, SAMOS, COMPOSE) and active rules (sequences of self-triggering ECA rules)
- **5. Process Calculi, Event Messaging and distributed rule-based Complex Event Processing**
 - Members, e.g. process calculi (CSP, CSS, pi-calculus, join-calculus), event/action messaging reaction rules (inbound / outbound messages), rule-based intelligent CEP with rule-based Event Processing Languages (EPLs, e.g. Prova, Reaction RuleML, AMIT, Rule Core)
 - Focus: process calculi focus on the actions, event messaging and CEP on the detection of complex events; often follow some workflow pattern, protocol (negotiation and coordination protocols) or CEP pattern

Usage of Ontologies

Top Level Ontologies

General concepts such as space, time, event, action and their properties and relations



Vocabularies **related to specific domains** by specializing the concepts introduced in the top-level ontology

Domain
Ontologies

Task
Activities
Ontologies

Specific
user/application
ontologies

Application
Ontologies

Vocabularies **related to generic tasks or activities** by specializing the concepts introduced in the top-level ontology

E.g. ontologies describing roles played by domain entities while performing application activities

See – **Reaction RuleML 1.0 Metamodel**
<http://www.slideshare.net/swadpasc/reaction-ruleml-ruleml2012paschketutorial>

Ontologies used for SCEP

- Top-Level Ontologies required for SCEP (core selection)
 - Spatial
 - Temporal
 - Event
 - Situation
 - Process (can be further specialized by domain ontologies such as OWL-S, WSMO, PSL)
 - Actor/Agent (can be further specialized, by special ontologies such as FIPA, ACL, ...)
 - Action: (can be used in e.g. RIF, RuleML, ...)
- Domain Ontologies for application verticals (samples domains of CEP applications)
 - Healthcare - e.g., Hospital Activity Monitoring
 - Finance - e.g., Fraud Detection
 - Logistics and Cargo
 - Supply Chain Management
 - Insurance
 - Mortgage

Examples of Ontologies which include Events

- CIDOC CRM: museums and libraries
- ABC Ontology: digital libraries
- Event Ontology: digital music
- DOLCE+DnS Ultralite: event aspects in social reality
- Event-Model-F: event-based systems
- VUevent Model: An extension of DOLCE and other event conceptualizations
- IPTC. EventML: structured event information
- GEM: geospatial events
- Event MultiMedia: multimedia
- LODE: events as Linked Data
- CultureSampo: Publication System of Cultural Heritage
- OpenCyC Ontology: human consensus reality, upper ontology with lots of terms and assertions
- Super BPEL: ontology for the business process execution language
- Semantic Sensor Net Ontology: ontology for sensor networks
- ...

Example: Semantic CEP - Filter Pattern

Filter Pattern:

Stocks of companies, which have production facilities in Europe *and*
produce products out of metal *and*
Have more than 10,000 employees.

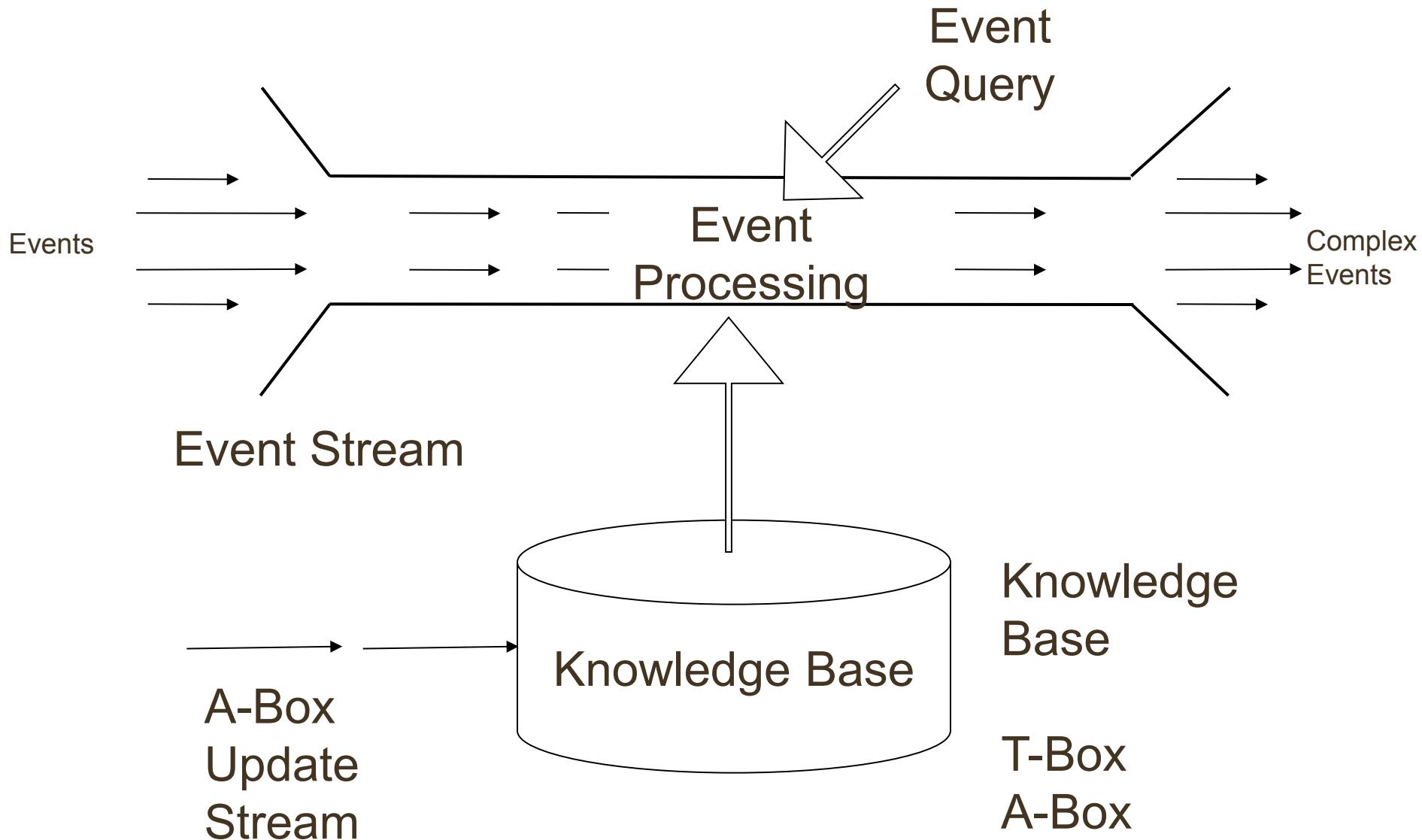
Event Stream – stock quotes

```
{(Name, "OPEL")(Price, 45)(Volume, 2000)(Time, 1) }  
{(Name, "SAP")(Price, 65)(Volume, 1000) (Time, 2)}
```

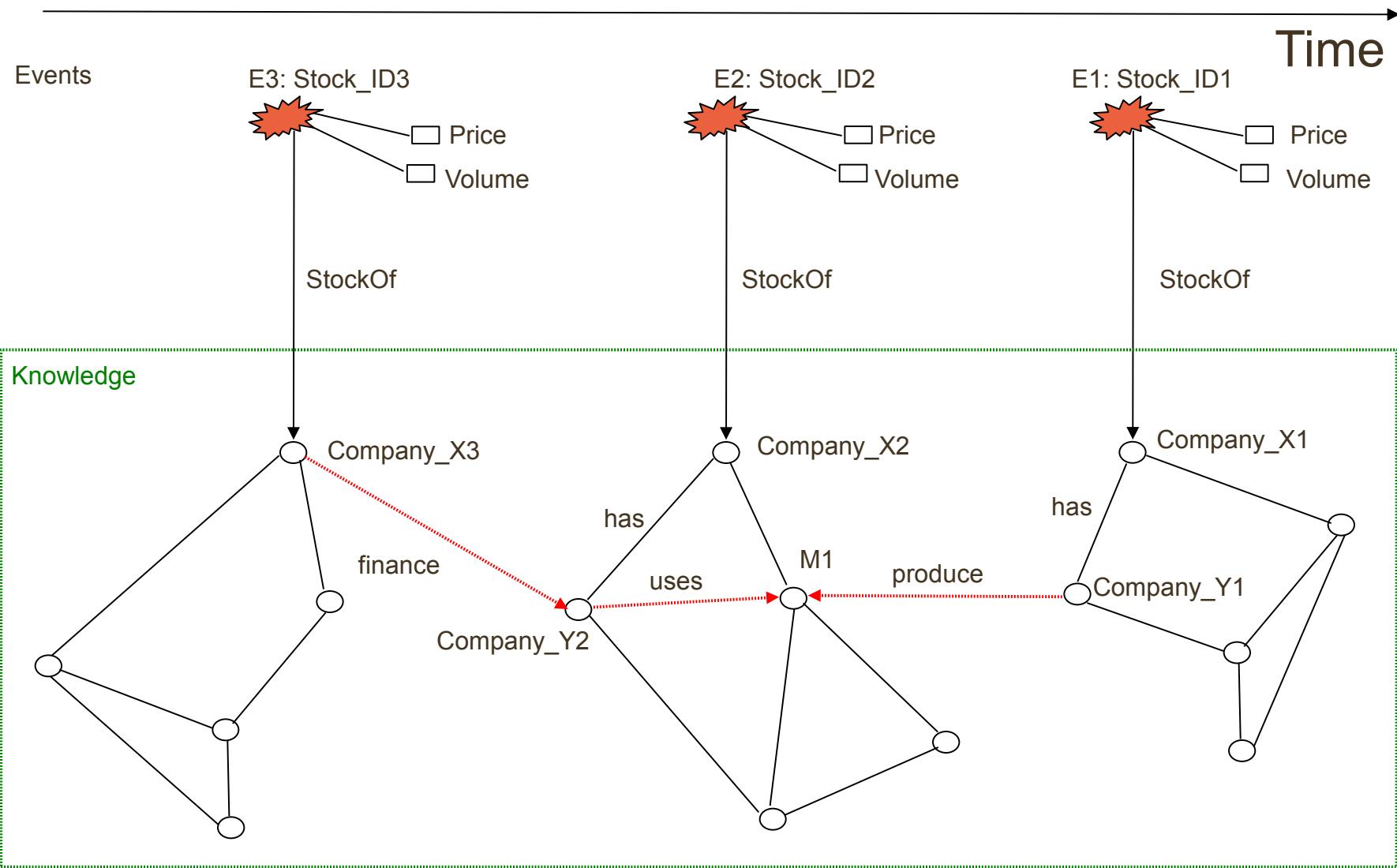
Semantic
Knowledge Base

```
{(OPEL, is_a, car_manufacturer),  
 (car_manufacturer, build, Cars),  
 (Cars, are_build_from, Metall),  
 (OPEL, hat_production_facilities_in, Germany),  
 (Germany, is_in, Europe)  
 (OPEL, is_a, Major_corporation),  
 (Major_corporation, have, over_10000_employees)}
```

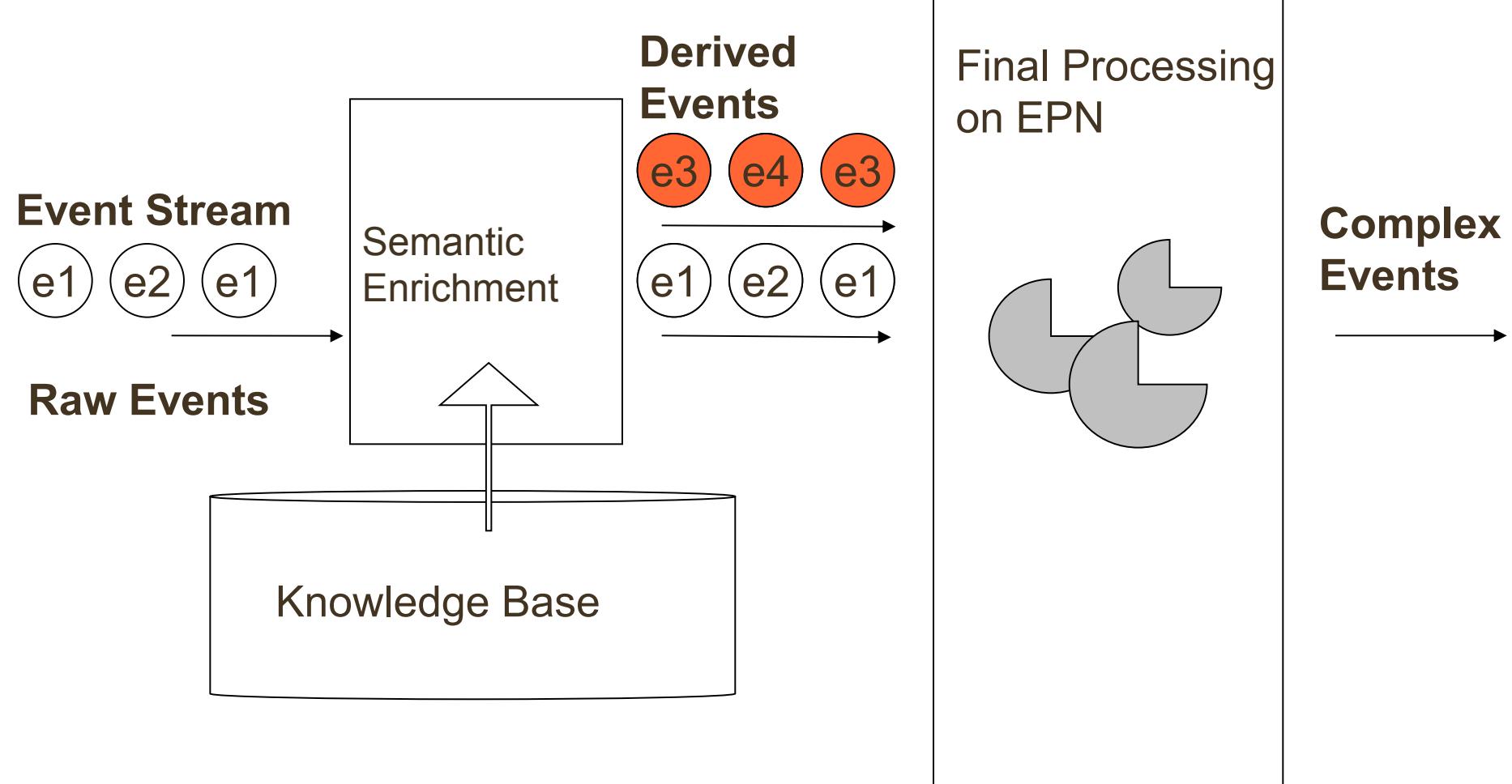
Knowledge-based Event Processing



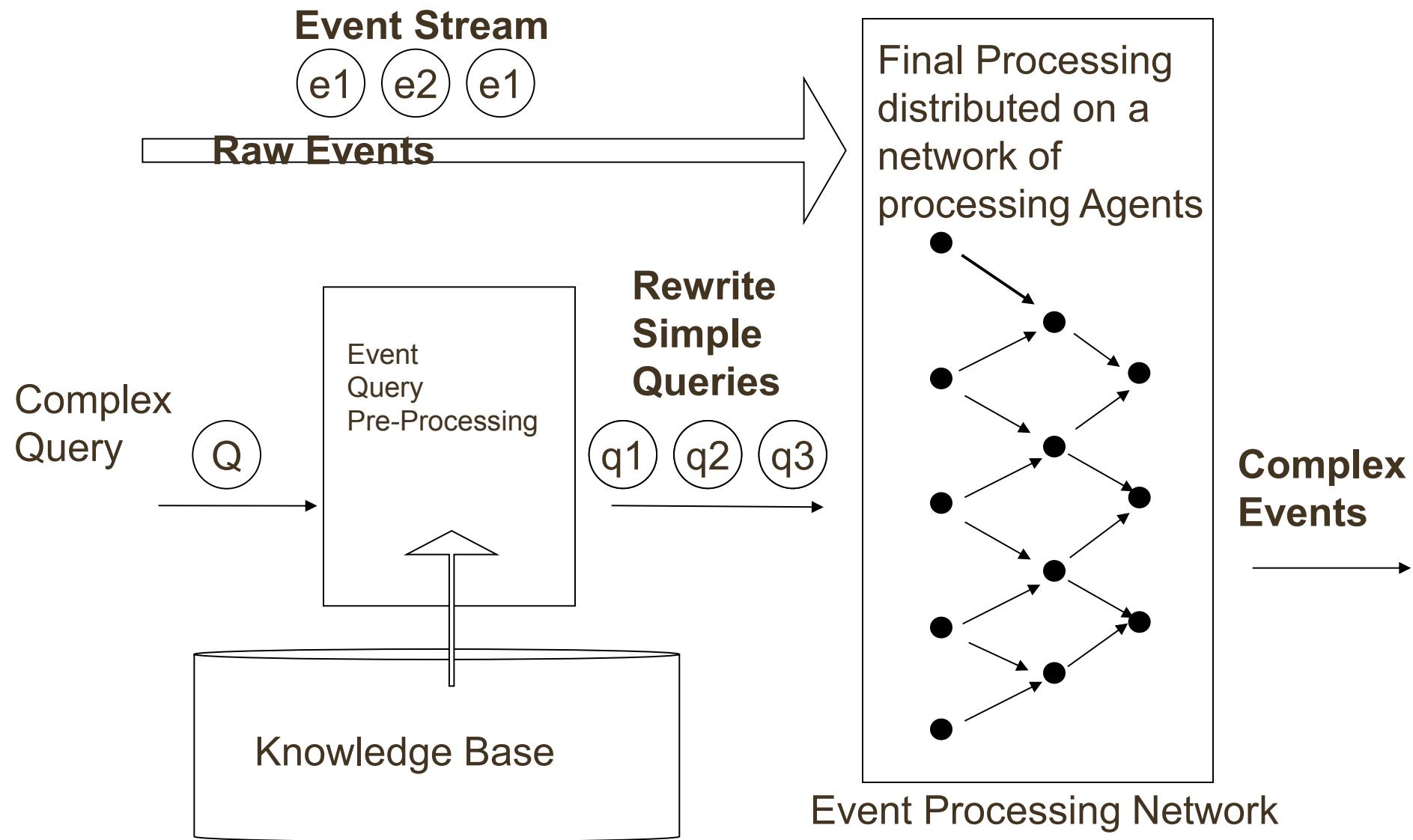
Example of Semantic Event Queries



Example: Semantic Enrichment of Event Stream



Example: Semantic Event Query Pre-Processing



Summary Semantic CEP: Selected Benefits

- Event data becomes **declarative knowledge** while conforming to an underlying **formal semantics**
 - e.g., supports automated semantic enrichment and mediation between different heterogeneous domains and abstraction levels
- Reasoning over **situations and states** by event processing agents
 - e.g., *a process is executing when it has been started and not ended*
 - e.g. *a plane begins flying when it takes off and it is no longer flying after it lands*
- Better understanding of the **relationships between events**
e.g., temporal, spatial, causal, ..., relations between events, states, activities, processes
 - e.g., *a service is unavailable when the service response time is longer than X seconds and the service is not in maintenance state*
 - e.g. *a landing starts when a plane approaches. During landing mobile phones must be switched off*
- **Declarative rule-based processing** of events and reactions to situations
 - Semantically grounded reaction rules

Agenda

- Introduction to Event Processing
- Semantic Complex Event Processing (SCEP)
- **Event Processing Technical Society (EPTS)**
 - Event Processing Standards **Reference Model**
 - Event Processing **Reference Architecture**
- Event Processing Function Patterns - Examples
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- Reaction RuleML Standard
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- Summary

About the Event Processing Technical Society

- **EPTS founded in June 2008**
 - IBM, Oracle, Tibco, Gartner, RuleML, ...
- **Steering Committee**
 - consists of founding members of the organization, representatives of major vendors and scientists
- **Working Groups**
 - Glossary Working Group
 - **Architecture and Meta-Model Working Group**
 - Language Working Group
 - Interoperability Working Group
 - Use Case Working Group
 - Business Value Working Group

About the EPTS Architecture and Metamodel Working Group

- **Started March, 2009**
 - 18 members, co-chairs are Adrian Paschke (RuleML) and Paul Vincent (TIBCO)
 - July 09 added responsibilities from Metamodel Working Group
- **Scope**
 - Define **reference architecture patterns** that are compatible with EPTS members' Event Processing solutions and products.
 - Define **terminology and components** regarding Event Processing in accordance with EPTS
 - Identify and utilize **best practices and methods** for Technical Architecture descriptions and interchange
 - **Liaise with relevant standards bodies** for EP metamodels and reference architectures

Reference Architecture and Reference Model

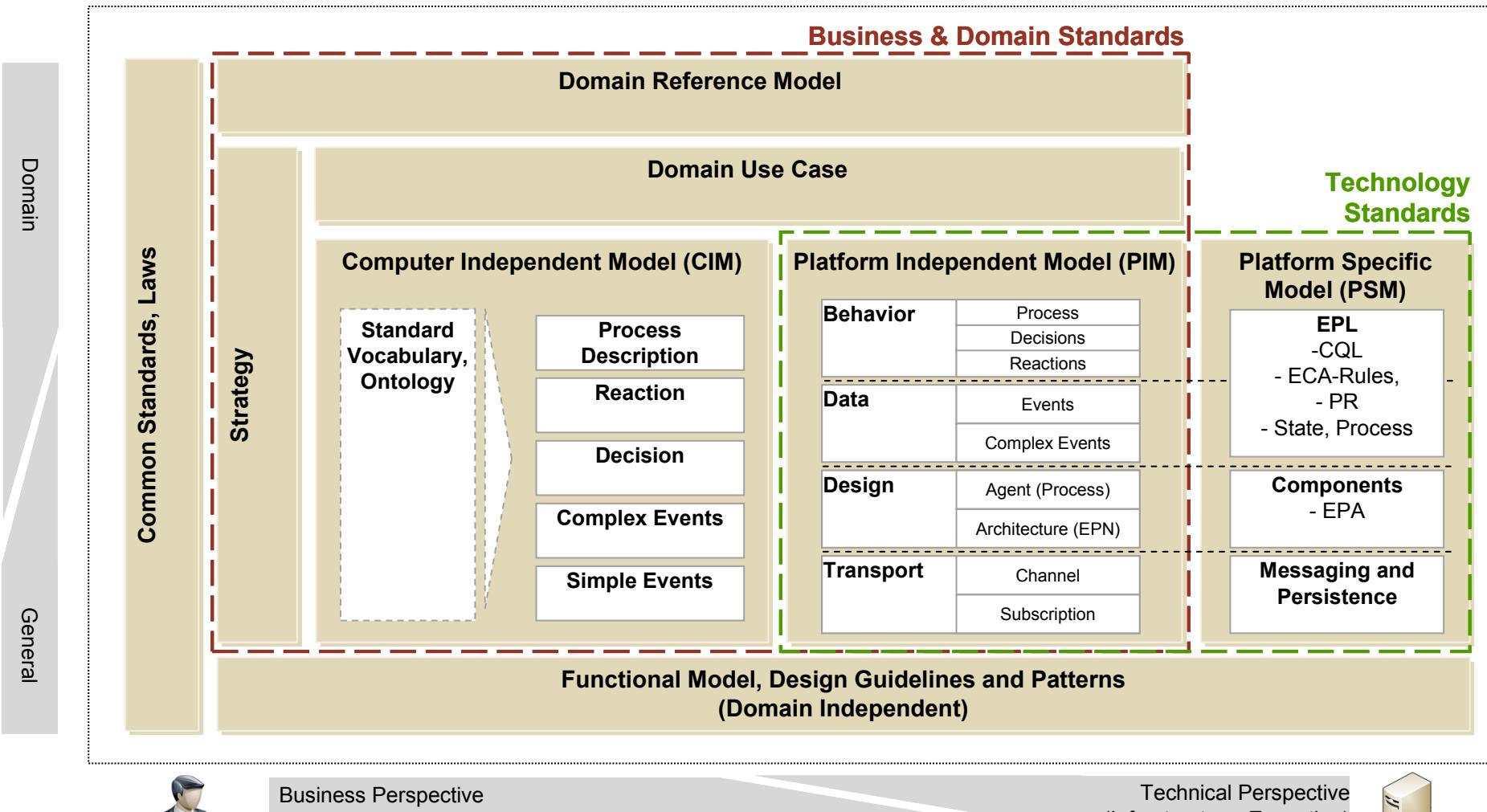
- **Reference Architecture**

A reference architecture **models the abstract architectural elements** in the domain independent of the technologies, protocols, and products that are used to implement the domain.

- **Reference Model**

A reference model **describes the important concepts and relationships** in the domain focusing on what distinguishes the elements of the domain.

EPTS CEP / Reaction RuleML Standards Reference Model



Standards Classification (1)

- **Common Standards, Laws**
 - Common standards and laws provide conditions and regulations which have to be considered in the business, e.g. laws for stock trading. Development of a software application has to follow these laws.
- **Domain Reference Model**
 - Domain Reference Models provide a subset of best practices, reference business processes, etc. for a specific business domain. These models help adaptors in defining a proper application based on well proven best practices, e.g. a well proven reference business process for the manufacturing domain.
- **Domain Use Case**
 - The use case describes how a problem can be solved, e.g. how to detect fraud in stock trading. It is often derived out of the business strategy and the reference model.

Standards Classification (2)

- **Strategy**
 - The strategy defines the individual business strategy of a company which has to be considered in developing new applications. The strategy consists of business (business motivations, goals, policies, rules) as well of technical (applications infrastructure, allowed frameworks) conditions.
- **Functional Model**
 - The functional model provides domain independent best practices and guidelines helping to design and develop a proper application, e.g. "Design Patterns - Elements of Reusable Object-Oriented Software"

Standards Classification - CIM

- **Computer Independent Model**

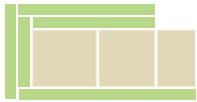
- The Computer Independent Model (CIM) describes the business functionality or system without reference to IT specifics. CIM should provide a common understanding between business users and software developers. In the CEP model it consists of following components:
- Standard Vocabulary, Ontology The ontology provides a set of definitions and terms enabling a common understanding for all stakeholders.
- Process description: Description of activities and its flow to produce a specific product or service enhanced with events occurring in it or influencing it.
- Reaction Definition of the activities which have to be initiated based on the previous taken decision.
- Decision Definition of rules, what has to be done if a relevant situation was detected by a pattern matching
- Complex Events Definition of event correlations for detection of relevant situations defined by patterns representing knowledge from source events, e.g. for detection of fraud.
- Simple Events Definition of attributes and types consisting of a simple event, e.g. the event "Stock Price" consists of the fields WKN, amount, etc.

Standards Classification - PIM

- **Platform Independent Model Standards (PIM)**
- The Platform Independent Model layer represents behavior, data, design, and messaging, independent from a particular EP platform.
- Event-Driven Behavior Effects of events lead to some (state) changes in the properties of the world which can be abstracted into situations.
- Event Data Platform-independent representation of events and their data is crucial for the interoperation between EP tools and between domain boundaries.
- Event Processing Design Platform-independent (reference) architecture and design models addressing different views for different stakeholders.
- Messaging PIM Messaging is addressing transport protocols and routing, coordination / negotiation mechanisms.

COMMON FRAMEWORK AND GUIDELINES

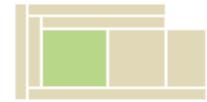
General Frameworks and Guidelines for Developing Systems



	Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Common Standards, Laws	<ul style="list-style-type: none"> ■ Industry standards, laws, etc. 	<ul style="list-style-type: none"> ■ Several laws (often country specific), e.g. BörsG for stock trading in Germany 	<ul style="list-style-type: none"> ■ A lot of existing standards are available ■ CEP Specific standards not necessary 	<ul style="list-style-type: none"> ■ Regulation of Use Cases ■ No Benefit for CEP
Strategy UseCase	<ul style="list-style-type: none"> ■ Overlying business motivations, goals, strategies, policies and business rules 	<ul style="list-style-type: none"> ■ Reduction of fraud by 10 percent, ... 	<ul style="list-style-type: none"> ■ OMG BMM ■ May need more emphasis on events and decision rules 	<ul style="list-style-type: none"> ■ Clarify the Business strategy
Reference Model	<ul style="list-style-type: none"> ■ Collection of best practices and reference business processes for a specific domain 	<ul style="list-style-type: none"> ■ Reference End-to-End process for logistics, e.g. <i>Pickup to Delivery</i> 	<ul style="list-style-type: none"> ■ Various per domain for data ■ Rarely handles events and rules 	<ul style="list-style-type: none"> ■ Helps in developing an efficient application
Use Case	<ul style="list-style-type: none"> ■ Description of how to solve a problem 	<ul style="list-style-type: none"> ■ Fraud detection for stock trading 	<ul style="list-style-type: none"> ■ UML (not CEP or rules specific) ■ EPTS UseCase Template 	<ul style="list-style-type: none"> ■ Create a common understanding between business user and software developer
Functional Model, Design Patterns ..	<ul style="list-style-type: none"> ■ Domain independent description of best practices, guidelines, etc. 	<ul style="list-style-type: none"> ■ Design Patterns for software engineering (Gamma) 	<ul style="list-style-type: none"> ■ Lots of different ones! ■ EPTS Functional Ref Architecture 	<ul style="list-style-type: none"> ■ Helps in implementing a proper application

COMPUTER INDEPENDENT MODEL 1/2

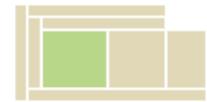
Computer Independent Model / Business Model for designing CEP Systems and Processes



	Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Standard Vocabulary, Ontology	<ul style="list-style-type: none">■ Common definitions of words enabling a common understanding for all stakeholders	<ul style="list-style-type: none">■ Stock Order: Transaction for buying a specific amount of stocks for price x	<ul style="list-style-type: none">■ Glossary, ontologies for events, time ...■ OMG SBVR, W3C OWL	<ul style="list-style-type: none">■ Common understanding for all stakeholders involved
Process Description	<ul style="list-style-type: none">■ Description of activities and its flow to produce a specific product or service enhanced with events occurring in it	<ul style="list-style-type: none">■ Buy Stocks requires “estimate risk”, “check price”, “check deposit”, “check availability”, etc.■ Events: StockPrice, ...	<ul style="list-style-type: none">■ Text descriptions of “events”■ BPMN, EPC■ Insufficient details on events and rules	<ul style="list-style-type: none">■ Create a common understanding between business user and software developer on a “big picture”
Decision	<ul style="list-style-type: none">■ Definition of rules, what has to be done if a relevant situation was detected by a pattern matching	<ul style="list-style-type: none">■ If a highly risk for the order was detected by a pattern the reaction “dropStockOrder” has to be done	<ul style="list-style-type: none">■ Decision Tree■ Decision Table (~DMN)■ Decision Rules standard missing	<ul style="list-style-type: none">■ Common understanding for all stakeholders involved
Reaction	<ul style="list-style-type: none">■ Definition of the activities which have to be done, based on the previous taken decision	<ul style="list-style-type: none">■ The stock order has to be dropped and the trader has to be informed	<ul style="list-style-type: none">■ None (Text based description)■ CIM Standard for Reaction Rules is missing	<ul style="list-style-type: none">■ Common understanding of possible outcomes all stakeholders involved

COMPUTER INDEPENDENT MODEL 2/2

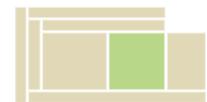
Computer Independent Model / Business Model for designing CEP Systems and Processes



Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Complex Events	<ul style="list-style-type: none">■ Definition of event correlations for detection of relevant situations defined by patterns representing knowledge from source events, e.g. for detection of fraud.	<p>Stocks of companies, where the stock price increase 3% in the last three days and which have production facilities in Europe <i>and</i> produce products out of metal <i>and</i> have more than 10,000 employees</p>	<ul style="list-style-type: none">■ None but - OMG EMP proposed■ New standard required
Simple Events	<ul style="list-style-type: none">■ Definition of attributes and types consisting an simple Event	<ul style="list-style-type: none">■ StockPrice:<ul style="list-style-type: none">- WKN (String 10)- ISIN (String 15)- Name (String 50)- Date/Time: (Time)- Price: (Decimal)- Currency: (String 3)...	<ul style="list-style-type: none">■ UML (Gaps in specifics needed for EP, ~EMP)■ Improve modelling languages eg NEAR

PLATFORM INDEPENDENT MODEL 1/4

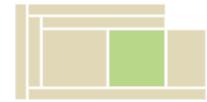
Platform Independent Model / IT Model for design of CEP Systems



Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Behavior <ul style="list-style-type: none">■ Event-Driven Behavior and Decisions:<ul style="list-style-type: none">■ Event occurrences lead to state changes in patterns, processes (= situations)■ Decisions represent the choices a system can after patterns detected or process states change■ Actions are triggered / performed as event reactions as the output of decisions and state/situation changes	<ul style="list-style-type: none">■ Event E is used in Pattern P = "E then F"■ Event E causes process R to start■ After Pattern P is detected we must decide how to respond to the situation S■ Process R must be invoked	<ul style="list-style-type: none">■ Rules: W3C RIF + Reaction RuleML;<ul style="list-style-type: none">- focused on rule-based behaviour■ OMG PRR<ul style="list-style-type: none">- lacks explicit events- specific to production rules■ OMG Decision Modeling Notation<ul style="list-style-type: none">- no explicit event■ OMG UML2 Behavioral View Diagrams<ul style="list-style-type: none">- Limited expressiveness of events and events = actions/activities■ OMG BPEL<ul style="list-style-type: none">- specific to business process execution with WS■ W3C WS Choreoigraphy<ul style="list-style-type: none">- Specific to WS■ and further EDA standards	<ul style="list-style-type: none">■ Declarative, explicit representation■ Publication and interchange of decisions and reactive behavior■ ...

PLATFORM INDEPENDENT MODEL 2/4

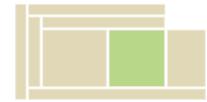
Platform Independent Model / IT Model for design of CEP Systems



Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Data <u>Some of ontologies which include events</u> CIDOC CRM: museums and libraries ABC Ontology: digital libraries Event Ontology: digital music DOLCE+DnS Ultralite: event aspects in social reality Event-Model-F: event-based systems VEvent Model: An extension of DOLCE and other event conceptualizations IPTC EventML: structured event information GEM: geospatial events Event MultiMedia: multimedia LODE: events as Linked Data CultureSampo: Publication System of Cultural Heritage OpenCyC Ontology: human consensus reality, upper ontology with lots of terms and assertions Semantic Complex Event Processing: top level meta model / ontology for semantic CEP	<ul style="list-style-type: none">■ Information carried by events and used in event patterns and processes■ May be processed as a part of event processing in the context of CEP/EP■ Archived +analysed for machine learning, BI, etc <ul style="list-style-type: none">■ Event Pattern P uses properties of event E1, E2■ Processing of event E1 requires operations on its properties■ Reaction to event E1 requires sending data D to system S	<ul style="list-style-type: none">■ Software Engineering: UML Structural View diagrams<ul style="list-style-type: none">– not specific to events■ Knowledge Representation: W3C RDFS/OWL ISO CL, ISO Topic Maps, OMG ODM UML/OWL Profiles<ul style="list-style-type: none">+ many existing event ontologies■ Rules: W3C RIF + Reaction RuleML<ul style="list-style-type: none">– specific to rule-based EP■ OASIS WS Topics<ul style="list-style-type: none">– Topic-based pubsub only■ OMG Event Meta Model<ul style="list-style-type: none">– not yet an RFP in OMG■ OASIS Common Base Event<ul style="list-style-type: none">– For business enterprise apps■ OASIS WS Notification and W3C WS Eventing<ul style="list-style-type: none">– specific to WS■ Rules: further standardization in W3CRIF/RuleML■ Standards for other domains needed, e.g. stream processing	<ul style="list-style-type: none">■ Declarative representation, translation and interchange of events■ Interchange and interoperation between different EP tools preserving the semantic interpretation of the event data■ Interchange events over domain boundaries which have different domain semantics / models■ Prerequisites Semantics: Explicit domain semantics Expressiveness Extensibility Reusability, specialisation and mapping

PLATFORM INDEPENDENT MODEL 3/4

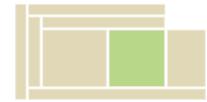
Platform Independent Model / IT Model for design of CEP Systems



Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Design <ul style="list-style-type: none"> Platform-Independent (Reference) Architecture and Model Descriptions addressing different views for different stakeholders 	<p>Mission fulfills 1..*</p> <p>Environment influences System</p> <p>System has an Architecture provides Rationale</p> <p>Architecture has Stakeholder 1..*</p> <p>Stakeholder identifies Architectural description 1..*</p> <p>Architectural description describes by Stakeholder 1..*</p> <p>Architectural description participates in View</p> <p>Concern has 1..* Stakeholder 1..*</p> <p>Concern identifies Viewpoint 1..*</p> <p>Concern used to cover Viewpoint 1..*</p> <p>Viewpoint selects View 1..*</p> <p>Viewpoint conforms to View 1..*</p> <p>View organized by Viewpoint 1..*</p> <p>View participates in Model 1..*</p> <p>View consists of 1..* Model 1..*</p> <p>Model establishes methods for View 1..*</p> <p>Model aggregates Library viewpoint 1..*</p>	<ul style="list-style-type: none"> Reference Architecture: models the abstract architectural design elements Reference Model: describes the important concepts and relationships in the design space EPN: Distributed Network Architecture Agents: abstract components given a specific abstract role and behavior/responsibilities <ul style="list-style-type: none"> ISO/IEC 42010:2007 Recommended Practice for Architectural Description of Software-intensive Systems <ul style="list-style-type: none"> Needs to be tailored to EP architecture descriptions UML 2 Implementation View Diagrams <ul style="list-style-type: none"> limited expressiveness Not specialised for EP design; only general component and composite structure design Agents: OMG Agent Metamodel; FIPA Agent Model; ... <ul style="list-style-type: none"> agent specific Workflow Management Coalition Reference Model <ul style="list-style-type: none"> workflow model 	<ul style="list-style-type: none"> Abstraction from the PSM design Increases understandability and reusability Agent model is an abstraction from technical IT components to role-based agents on a more abstract level

PLATFORM INDEPENDENT MODEL 4/4

Platform Independent Model / IT Model for design of CEP Systems



Description	Example	Existing Standards (+Gaps)	Benefits of Standardization
Messaging	<ul style="list-style-type: none">■ Transport protocols and routing coordination / negotiation mechanisms■ synchronous vs. asynchronous transport■ Push vs. pull■ Coordination: Publish Subscribe, ContractNet, ...■ ...	<ul style="list-style-type: none">■ Many transport protocols: JMS, JDBC, TCP, UDP, multicast, http, servlet, SMTP, POP3, file, XMPP<ul style="list-style-type: none">– just needed for transport■ Message Routing Patterns<ul style="list-style-type: none">– message routing■ Coordination and Negotiation Mechanisms/Patterns	<ul style="list-style-type: none">■ well understood and standardized transport protocols

PLATFORM SPECIFIC MODEL 1/3

Platform Specific Model (PSM)
describes the concrete Implementation of a CEP Application



	Description	Example	Existing Standards (+Gaps)	Benefits of Standardisation
EPL CQL	<ul style="list-style-type: none">■ Continuous query language running queries against policies such as time windows, numbers of events, etc	<ul style="list-style-type: none">■ SELECT x, y from REUTERS WHERE x.id == y.id AND x.price > y.price * 1.1 IN WINDOW(600, "secs")■ Vendor specific versions include Oracle CQL, Streambase CQL, Sybase CQL, TIBCO BQL	<ul style="list-style-type: none">■ ANSI SQL extended for continuous policy■ ODBMS OQL extended for continuous policy	<ul style="list-style-type: none">■ Interchange queries between vendor platforms (design, deployment) [IT department]■ Encourage training / education on CQL (cf SQL takeup) [university, student]■ Dependancies: common data descriptors
EPL ECA Rules	<ul style="list-style-type: none">■ Event-Condition-Action rules triggered by creation of complex events representing event patterns (eg stream outputs), determining reactions based on certain conditions	<ul style="list-style-type: none">■ IF event. SigMvDet AND SigMvDet.stock = IBM AND SigMvDet.change = up THEN reqBuy(SigMvDet.stock)■ Vendor examples include TIBCO BE, IBM WSBE, Progress Apama■ Open Source, e.g. Prova, Drools	<ul style="list-style-type: none">■ W3C RIF PRD extended for ECA (RIF ECA)■ Reaction RuleML■ JSR-94 rule execution API	<ul style="list-style-type: none">■ Interchange rules between vendor platforms (design, deployment) [IT department]■ Encourage training / education on ECA rules (cf Java takeup) [university, student]■ Dependancies: common data descriptors

PLATFORM SPECIFIC MODEL 2/3

Platform Specific Model (PSM)
describes the concrete Implementation of a CEP Application



	Description	Example	Existing Standards (+Gaps)	Benefits of Standardisation
EPL TRE	<ul style="list-style-type: none">■ Temporal Regular Expressions defining complex events in terms of event / non-event sequences over time	<ul style="list-style-type: none">■ WHEN event.A, event.B WHERE event.A (*2) FOLLOWED BY no event.B IN time(200,ms)■ Vendor examples include Oracle CQL (TRE embedded in SQL), Progress Apama (TRE), TIBCO PML	<ul style="list-style-type: none">■ Perl Regex with temporal extensions	<ul style="list-style-type: none">■ Interchange rules between vendor platforms (design, deployment) [IT department]■ Encourage training / education on TRE patterns (cf Java takeup) [university, student]■ Dependancies: common data descriptors
Application Architecture	<ul style="list-style-type: none">■ Application component layouts / configurations, specifying various processes / engines / agents and how they are connected.■ Architecture may also be considered platform independent in terms of general requirements, but platform specific in terms of associating specific logical layouts, interfaces, and agent / engine numbers to achieve specified SLAs	<ul style="list-style-type: none">■ Agent 1 type inference contains rulesets R1, R2 with destinations D1, D2 Deployment C1 is of 4 x Agent 1 role active-active■ Vendor examples include TIBCO CDD Cluster Deployment Descriptor	<ul style="list-style-type: none">■ OMG AMP Agent Metamodel and Profile (draft)	<ul style="list-style-type: none">■ Compare application designs across vendors [IT operations, Enterprise Architects]■ Dependancies: common agent and interface descriptors

PLATFORM SPECIFIC MODEL 3/3

Platform Specific Model (PSM)
describes the concrete Implementation of a CEP Application

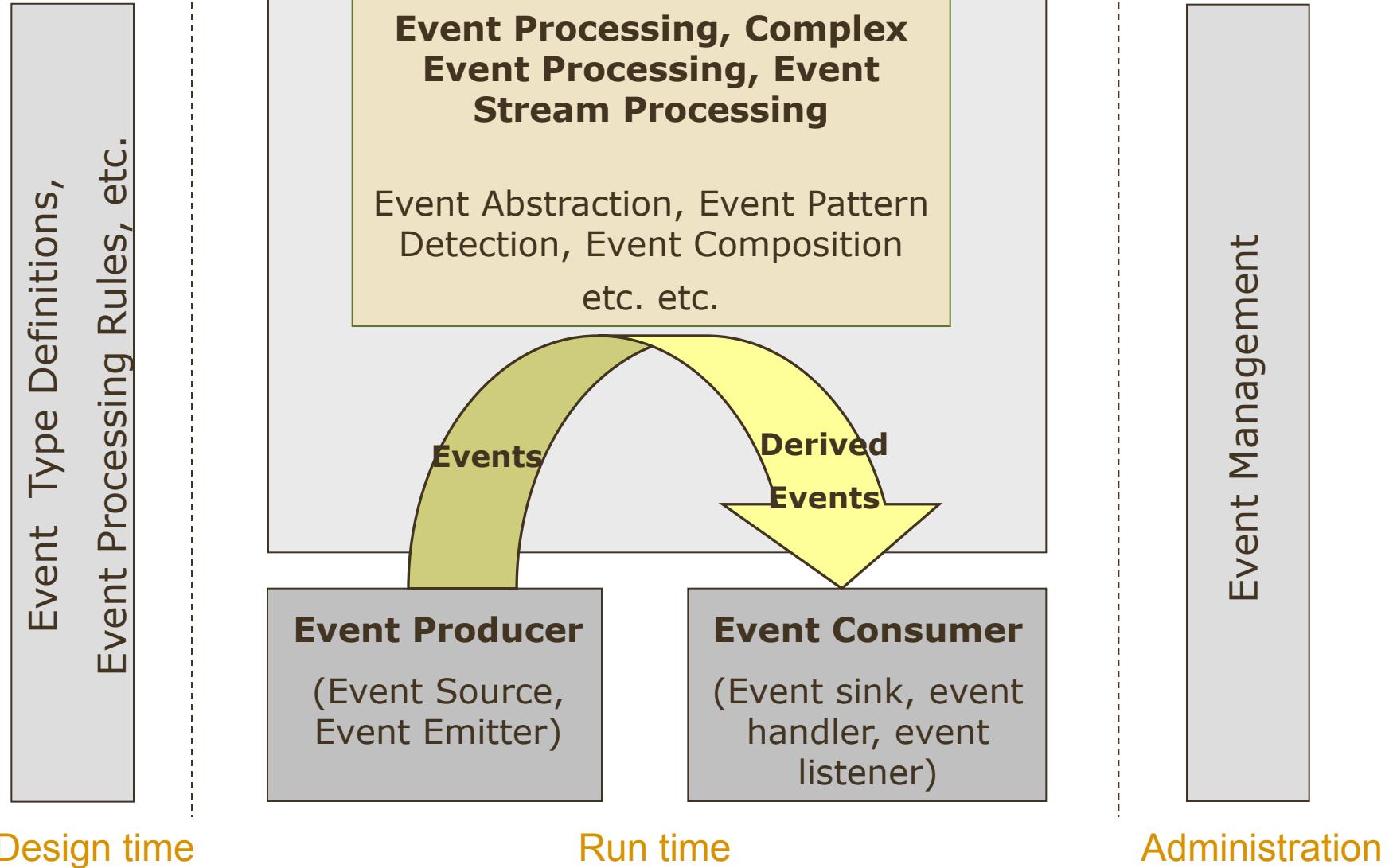


Description	Example	Existing Standards (+Gaps)	Benefits of Standardisation
Messaging System	<ul style="list-style-type: none">■ Middleware type and configuration <ul style="list-style-type: none">■ JMS includes header information and payload information, the latter which includes XML document per some XSD. JMS extensions could be for guaranteed delivery, timeout etc.	<ul style="list-style-type: none">■ JMS (Java spec)■ OMG DDS■ AMQP <ul style="list-style-type: none">■ Gap: some abstract definition of a messaging software per the above	<ul style="list-style-type: none">■ Late selection of messaging type [IT, IT operations]■ Dependancies: common payload descriptors

Agenda

- Introduction to Event Processing
- Semantic Complex Event Processing (SCEP)
- **Event Processing Technical Society (EPTS)**
 - Event Processing Standards Reference Model
 - Event Processing **Reference Architecture**
- Event Processing Function Patterns - Examples
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- Reaction RuleML Standard
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- Summary

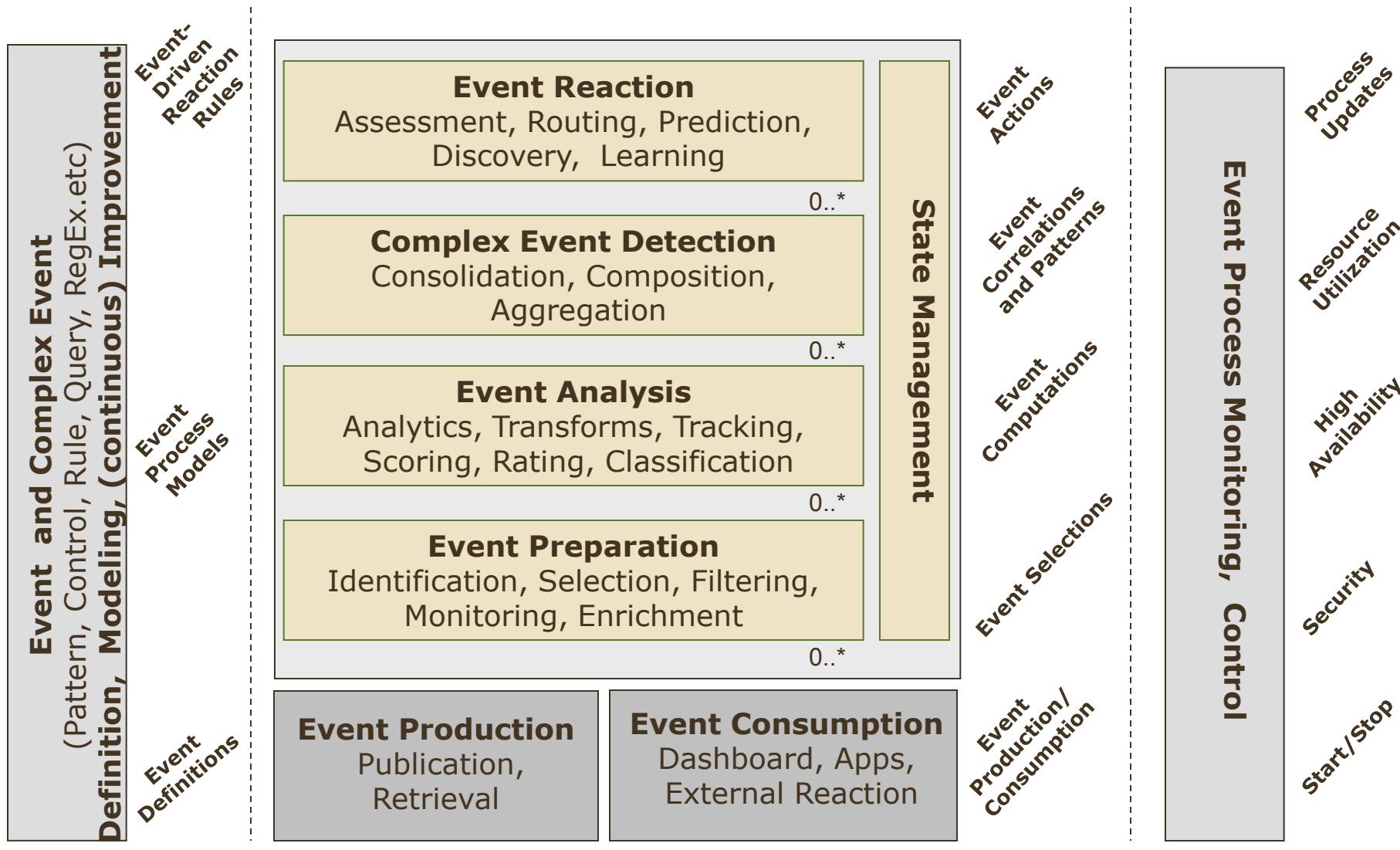
Functional View source: definitions of EP



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

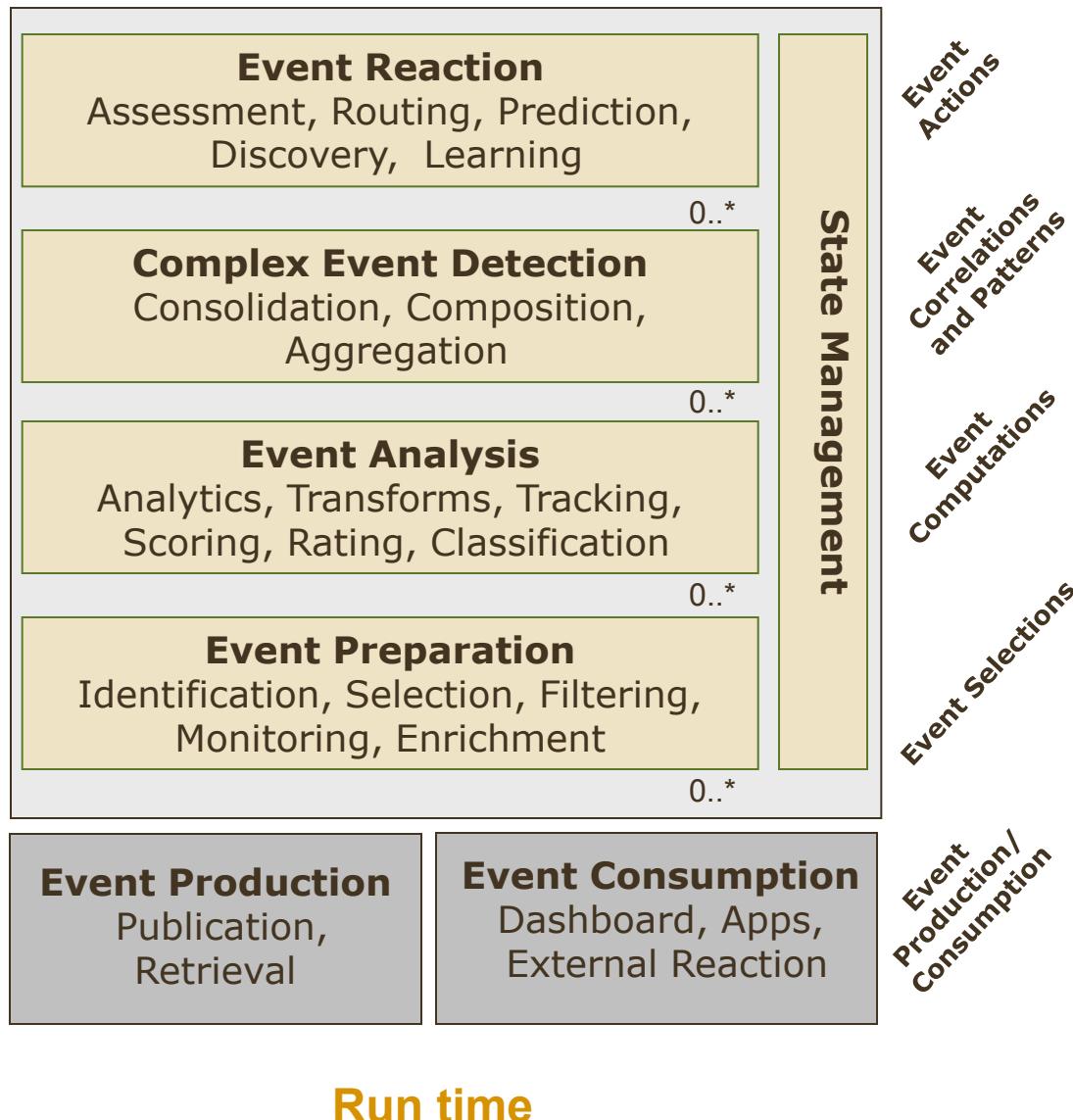


Design time

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 (eg 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.

Event Production
Publication,
Retrieval

Event Consumption
Dashboard, Apps,
External Reaction

Event
Production/
Consumption

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

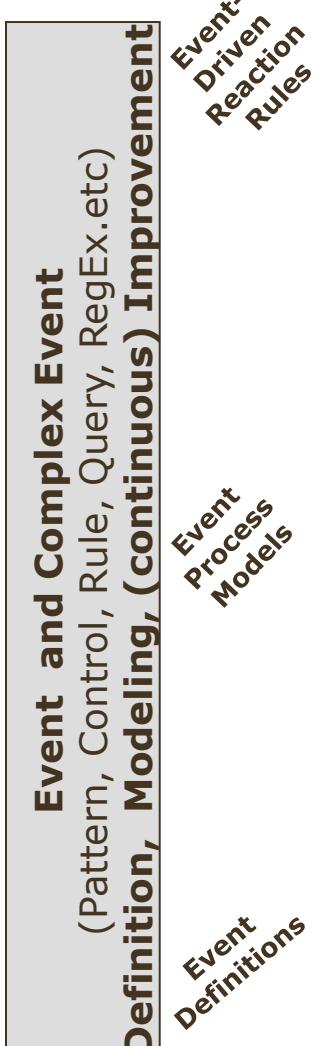
Event Reaction: the process subsequent to event analysis and complex event detection to handle the results of analysis and detection.

- **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



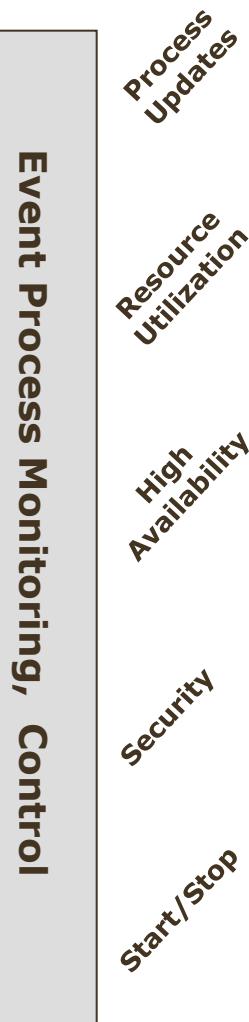
Covers the definition, modeling, improvement / maintenance of the artefacts used in event processing:

- 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)

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.



Agenda

- **Introduction to Event Processing**
- **Semantic Complex Event Processing (SCEP)**
- **Event Processing Technical Society (EPTS)**
 - Event Processing Standards Reference Model
 - Event Processing Reference Architecture
- **Event Processing Function Patterns - Examples**
 - Implementation Examples in the **Prova Rule Engine (Platform Specific)**
- **Reaction RuleML Standard**
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- **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

see: 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

What do we want to cover in an EP Pattern?

Type /
synonym

Operands

Operation /
Operators

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.

Mapping to
EPL template

Rationale /
business rule

EP Design Pattern Description

- **Name & Alternative Names**
- **Classification versus other references**
 - EPTS RA and other references
- **Description**
 - Role, and Business Rule type specification / requirement
- **Structure**
 - EPTS Glossary terms
- **Implementations**

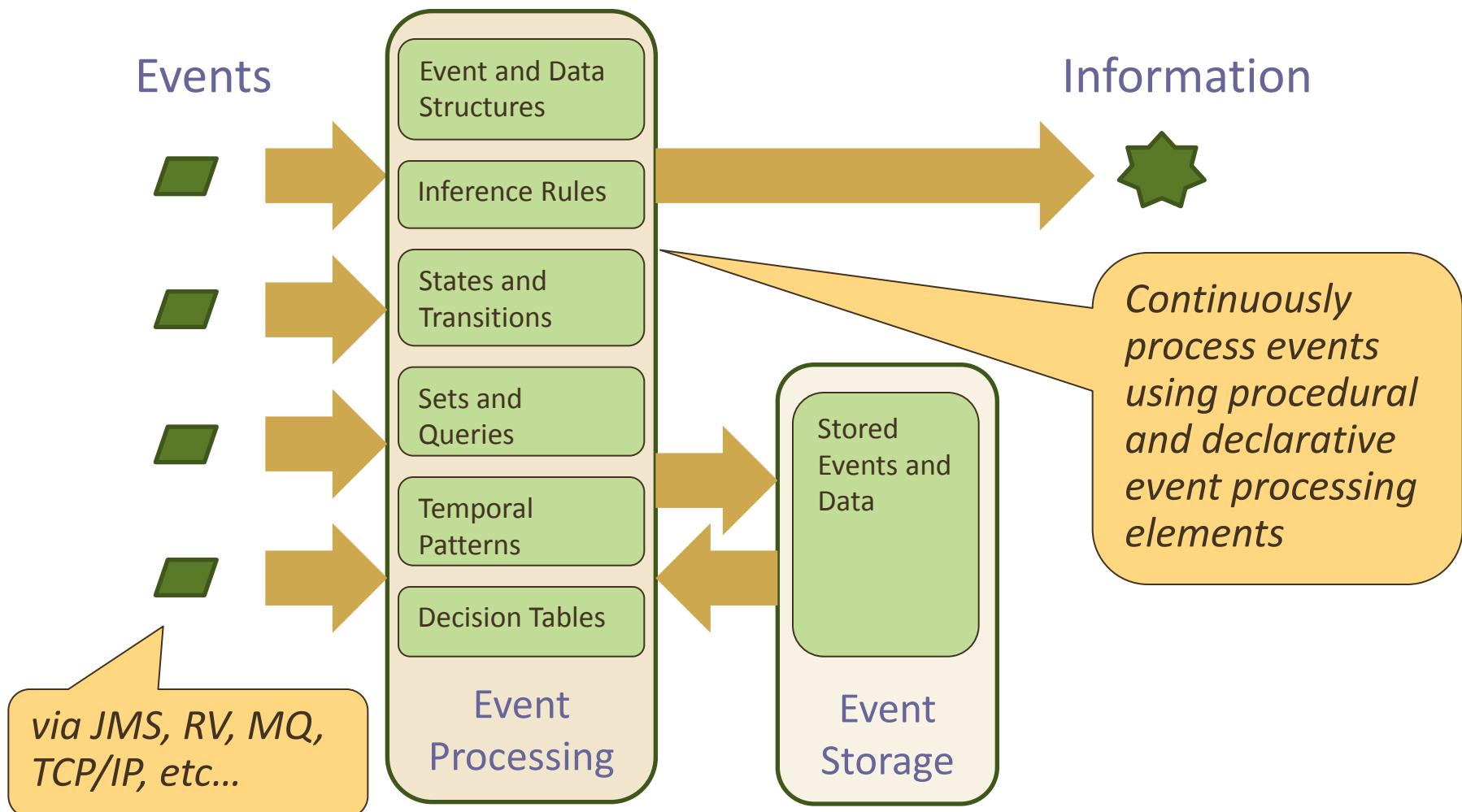
EP Pattern Implementations - PSM

- Differ in Implementation type (Event Processing Language, Execution semantics, etc)
 - Sampled systems covered, e.g.:
 1. **TIBCO BusinessEvents** Rete-production-rule-engine
(with continuous query and event-expression options)
 2. **Oracle CEP** Stream-processing engine
 3. **Prova** Logic Programming Semantic CEP Rule Engine
 4. **IBM WebSphere Decision Server** event-driven production-rule-engine
& **IBM InfoSphere Streams** stream processing engine
- in this talk

Implementations: Sampled Systems: TIBCO BusinessEvents

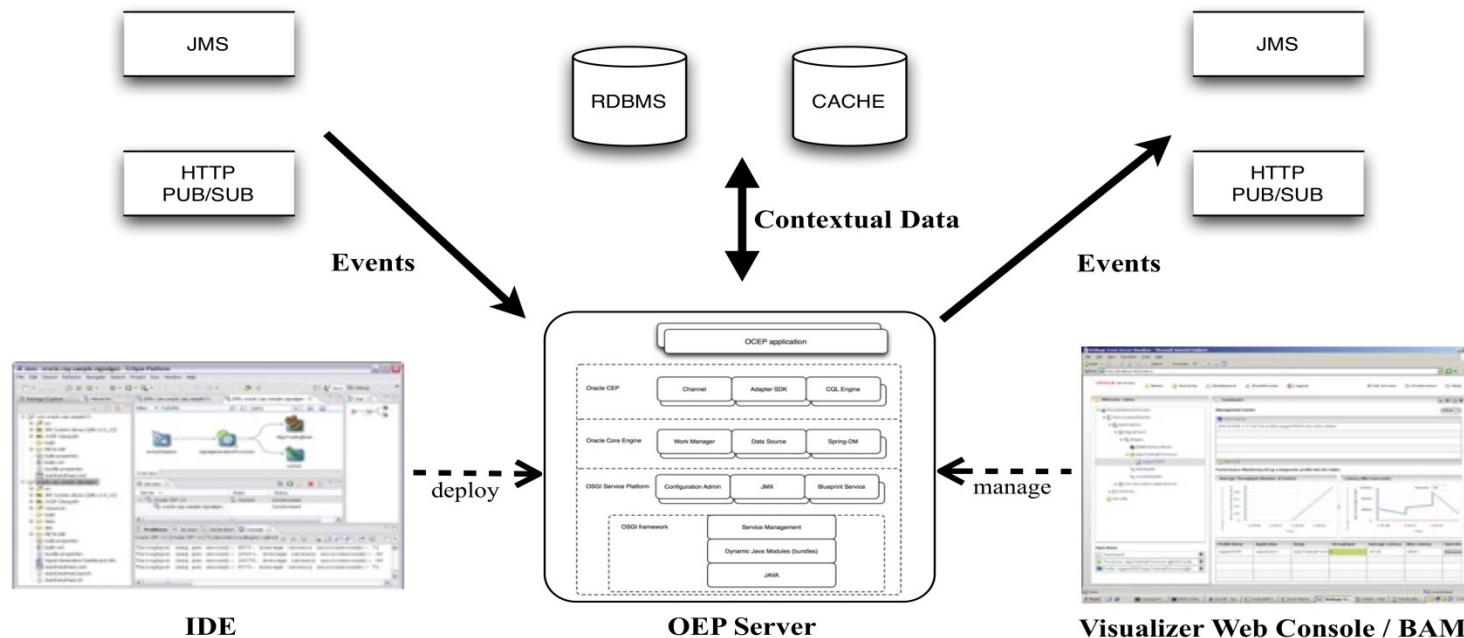
- **TIBCO BusinessEvents** “event server” / event processing engines
- Multiple languages / engines in various packaging options
 - UML-based event and concept (object) class hierarchical models
 - Java-based language
 - Eclipse-based IDE
- Distributed execution model:
 - Inference agents, Query agents, 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: Sampled Systems: TIBCO BusinessEvents (ctd)



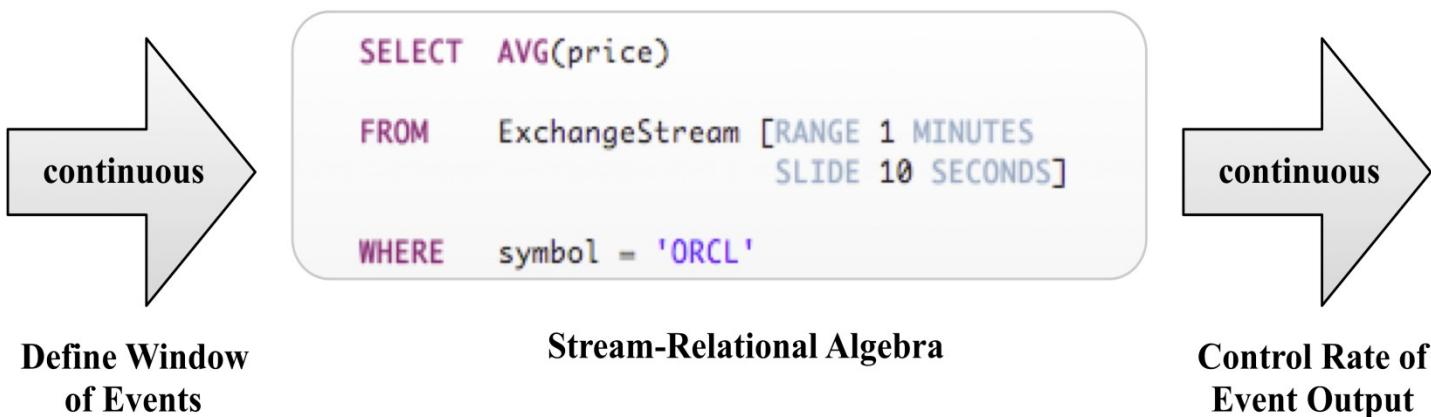
Implementations: Sampled Systems: Oracle Event Processing (formerly Oracle CEP)

- 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



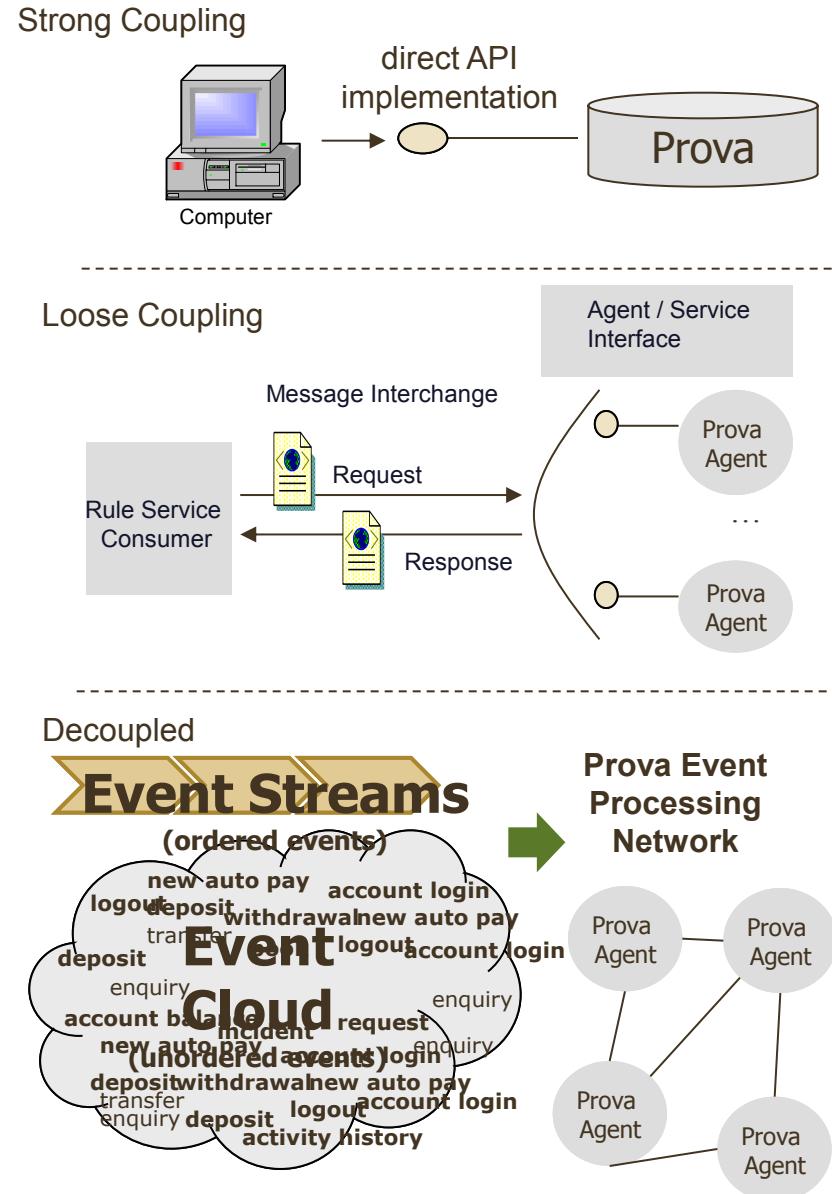
Implementations: Sampled Systems: Oracle Event Processing (formerly Oracle CEP)

- 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



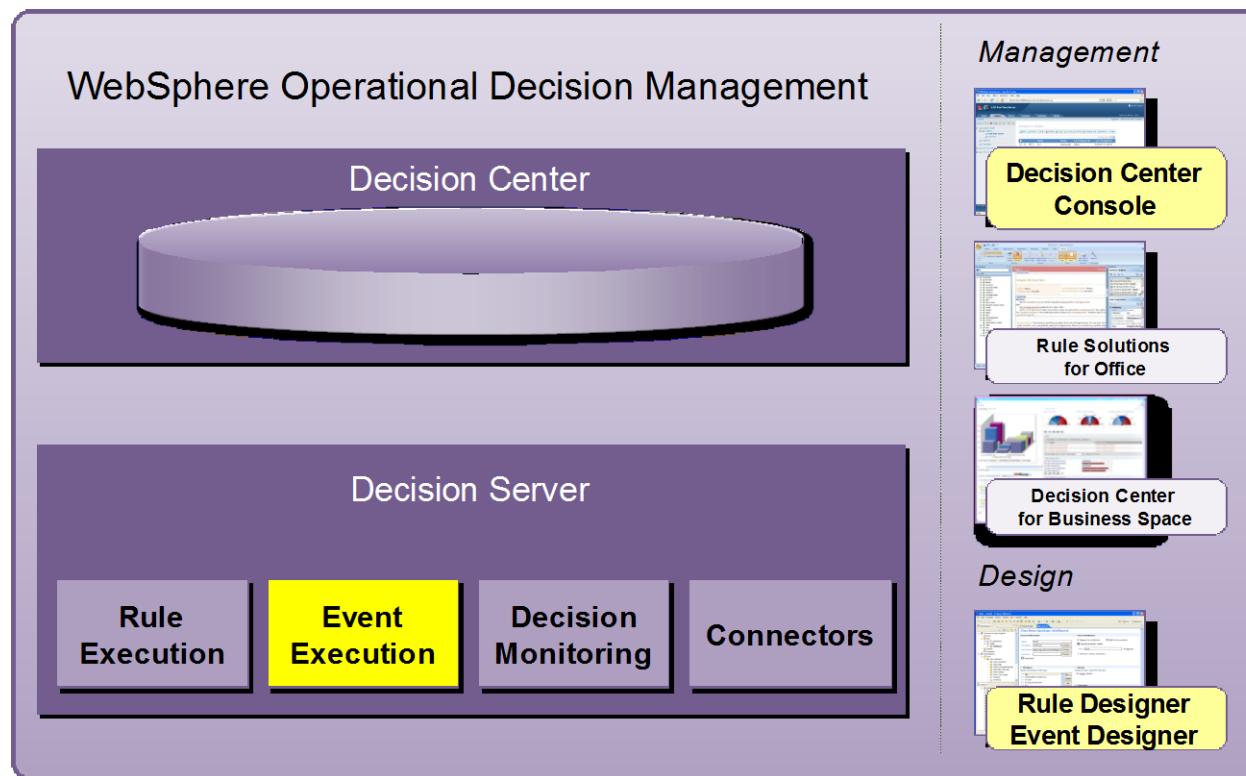
Implementations: Sampled Systems: Prova (<http://prova.ws>)

- 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



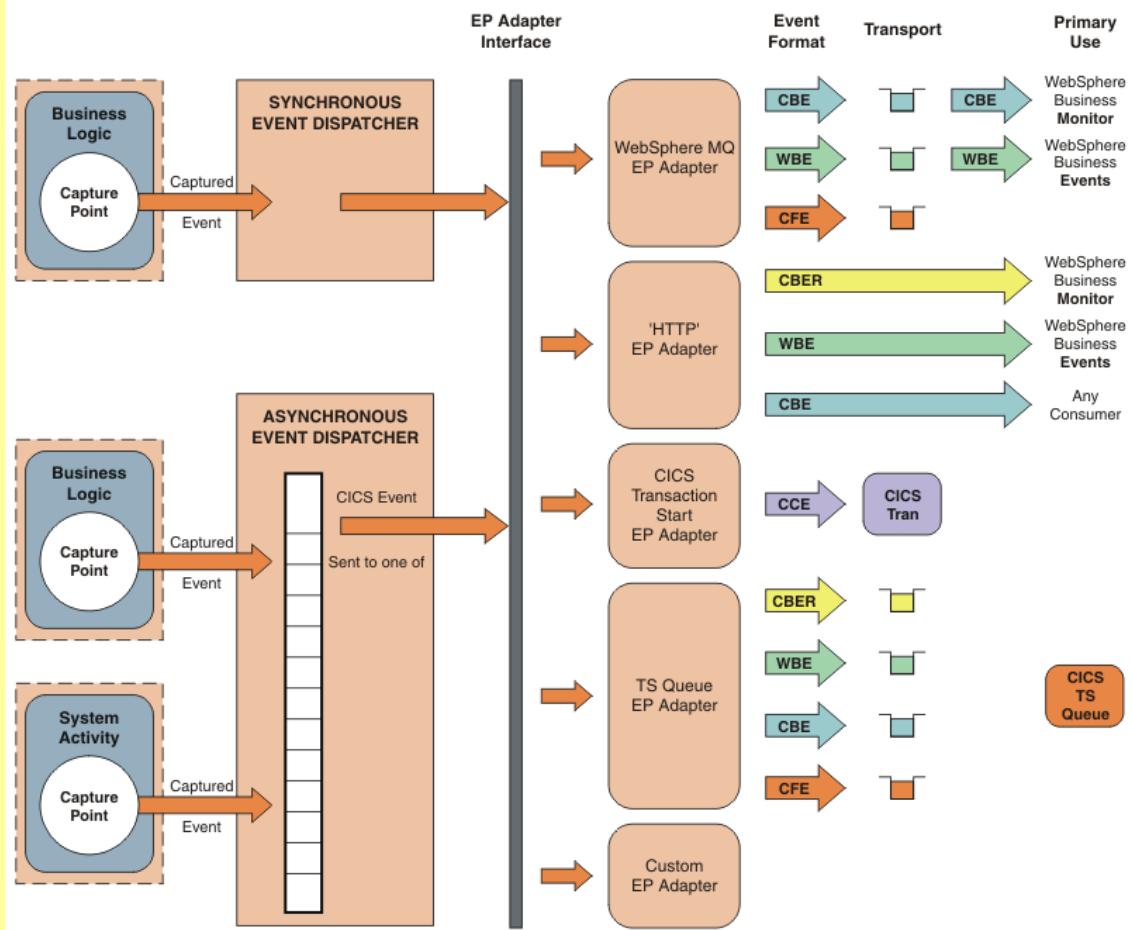
Implementations: Sampled Systems: IBM WODM Decision Server Events

- 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



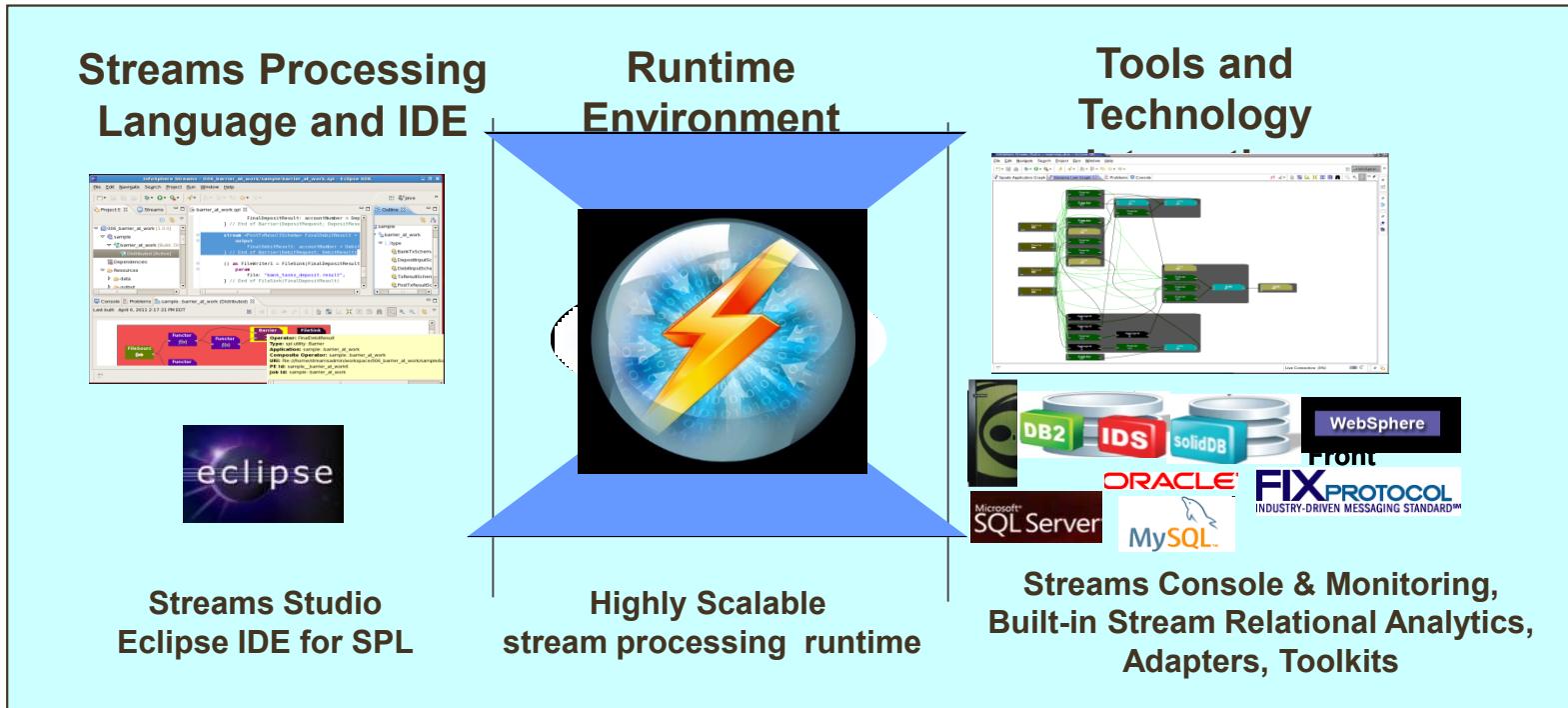
Implementations: Sampled Systems: IBM CICS Transaction Server for z/OS

CICS TS

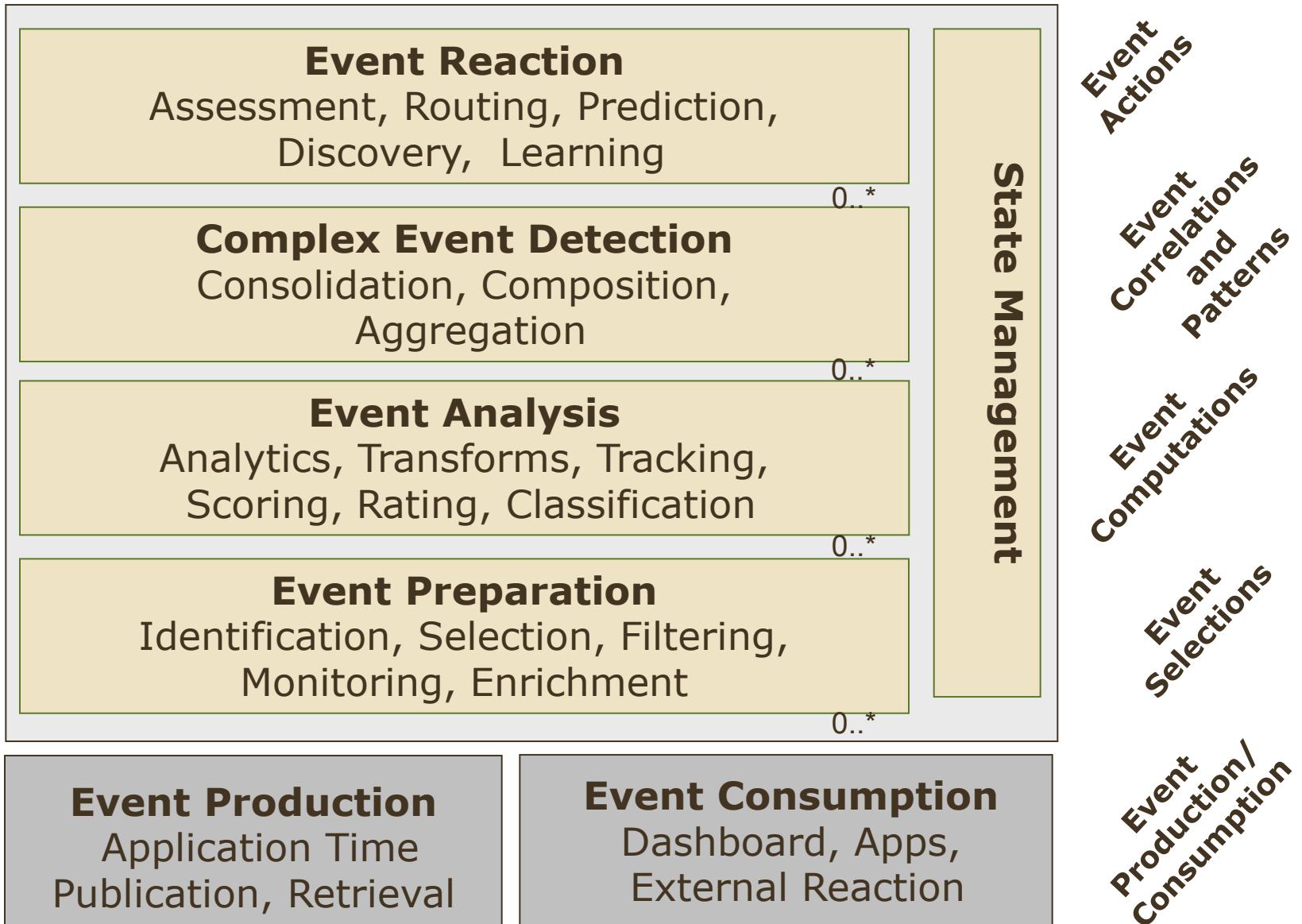


Implementations: Sampled Systems: IBM InfoSphere Streams – Stream Processing Language

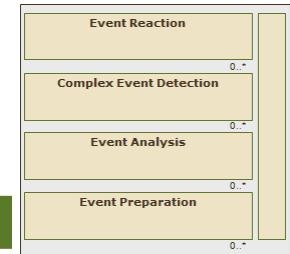
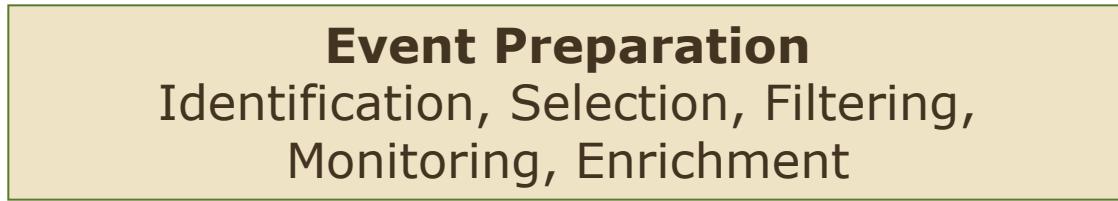
- IBM InfoSphere Streams: platform for analyzing big data in motion i.e. high-volume, continuous data streams SPL
- IBM Streams Processing Language (SPL)
 - Programming language for InfoSphere Streams
 - Exposes a simple graph-of-operators view
 - Provides a powerful code-generation interface to C++ and Java



Patterns Coverage



EP Patterns:RA:Preparation

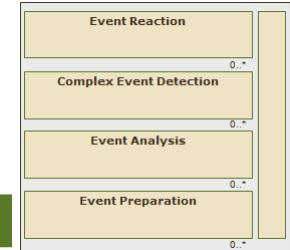


Event preparation is the process of preparing the event and associated payload and metadata for further stages of event processing.

For example, event preparation may involve the separation and discarding of unused event payload data, and the reformatting of that payload data for downstream event processing.

RA:Preparation:Identification:

Event Preparation
Identification, Selection, Filtering,
Monitoring, Enrichment



Identification: incoming events identified relative to prior events and event types

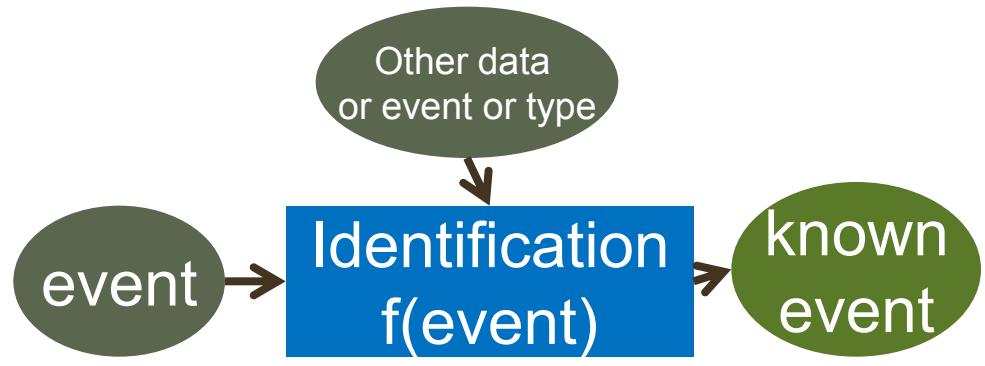
- E.g. associating a SMS event with particular mobile phone account
- E.g. recognizing an event from data (event entity recognition, extraction) and identifying an event to be of a particular event type

Preparation:Identification:Classification

- **Alternative Names:**
 - Event Lookup, Event Entity Recognition, Event Extraction
- **EPTS Reference Architecture:**
 - *“Incoming events will need to be identified relative to prior events and event types, such as associating events with particular sources or sensors or recognizing / extracting events from data relative to event type information (event type systems, event ontologies)”*

Preparation:Identification:Description

- **Role / Explanation**
 - Associating an event with some existing entity (data or past event, event type)
- **Associated Business Rule Specification**
 - a selection (to be enforced during the processing of events):
All <event entities> whose <attribute1> matches the <attribute2> of <some other event or data or type> is related by <relationship> to that <some other event or data or type>.



Preparation:Identification:Structure

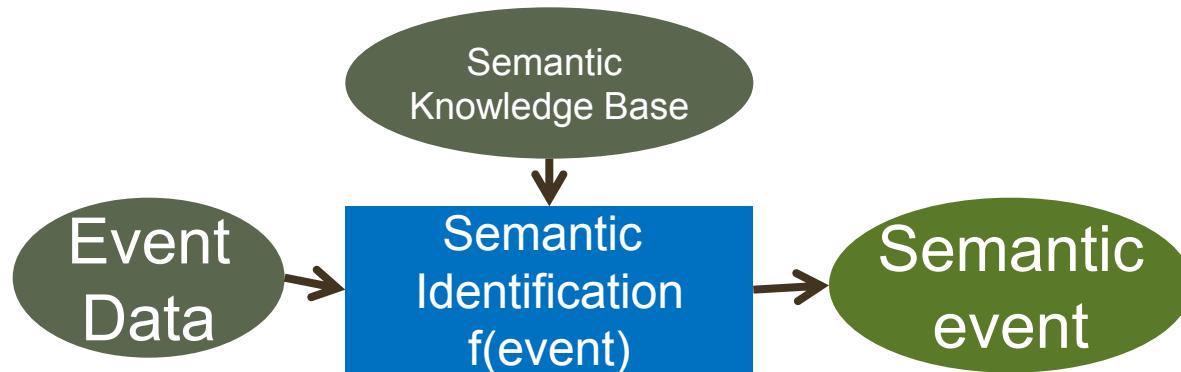
- **EPTS Glossary comparison**
 - Event type (event class, event definition, or event schema)
 - A class of event objects.
 - Event types should be defined within some type definition system ... will usually specify certain predefined data (attributes), examples of which might be:
 - A unique event identifier used to reference the event

Preparation:Identification:Implementations:Prova

Semantic Identification

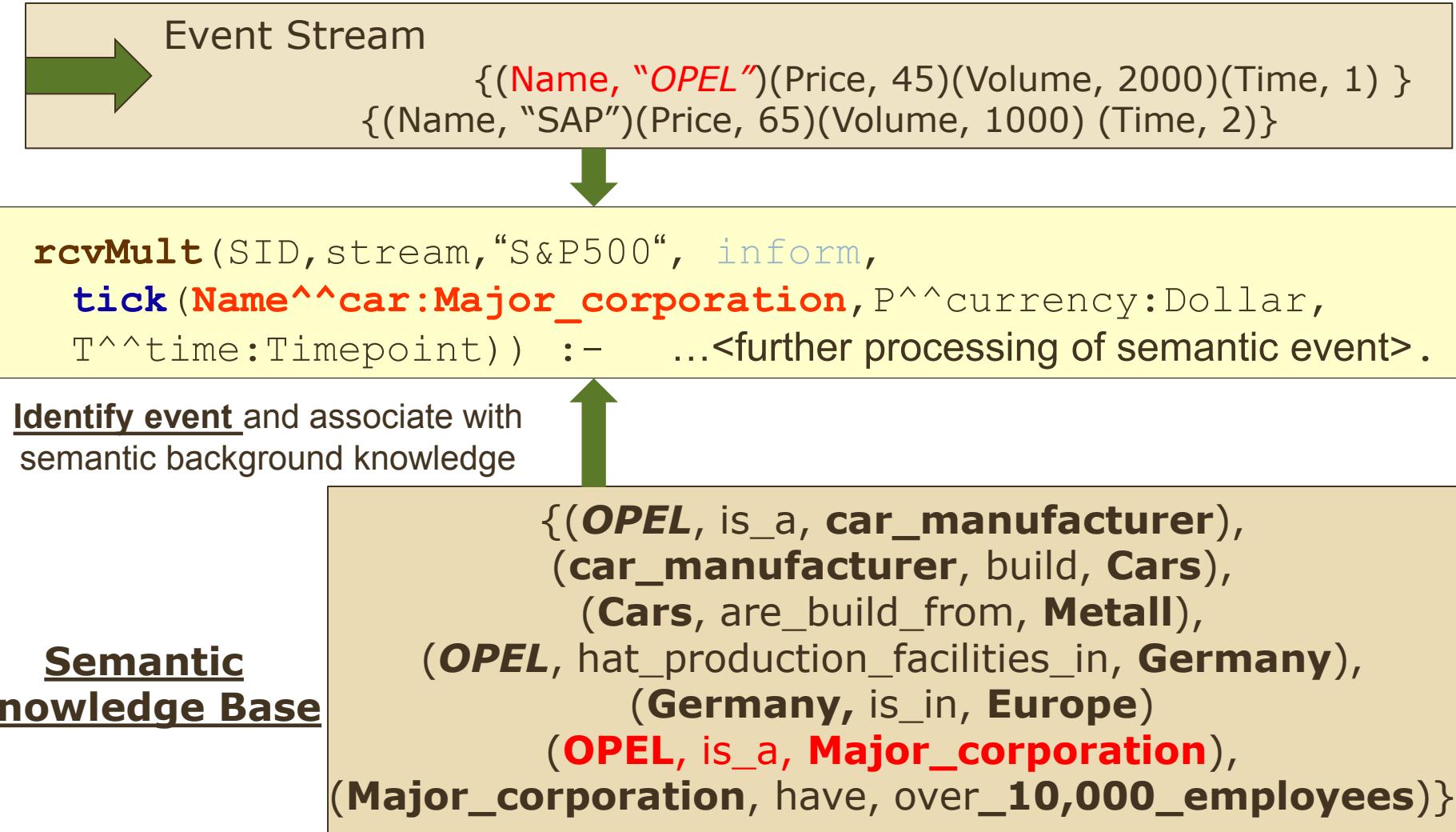
- **General pattern:**

`<rcvMsg / rcvMult> Event ^^ Semantic Type :-`



- **Effect:** find appropriate event entity type that matches the incoming event data and create a reference between the event and the event type (from the semantic knowledge), so that the syntactic (event) data becomes a (semantic) event, which has relations with the semantic background knowledge base

Preparation: Semantic Identification: Prova Semantic Rules



RA:Preparation:Selection:



Selection: particular events selected for further analysis or pattern matching

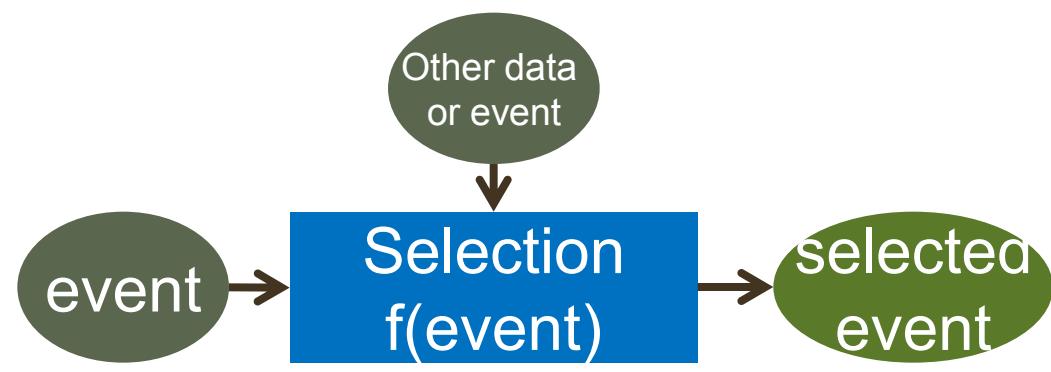
- E.g. Selecting the n'th event as a part of a sampling function
- E.g. Selecting events related to some existing event to allow enrichment of that existing event

Preparation:Selection:Classification

- **Alternative Names:**
 - Event Query
 - See also Event Filtering and Event Identification
- **EPTS Reference Architecture:**
 - *“During event preparation, particular events may be selected for further analysis. Different parts of event processing may require different selections of events. See also event filtering. ”*

Preparation:Selection:Description

- **Role / Explanation**
 - Selecting or locating an event due to some criteria or rules
- **Associated Business Rule Specification**
 - a selection (to be enforced during the processing of events):
All <event entities> whose <attribute1> matches the <attribute2> of <some other event or data> is selected for <processing>.



Preparation:Selection:Structure

- **EPTS Glossary comparison**
 - Event type (event class, event definition, or event schema)
 - A class of event objects.
 - Event types should be defined within some type definition system ... will usually specify certain predefined data (attributes), examples of which might be:
 - A unique event identifier used to reference the event

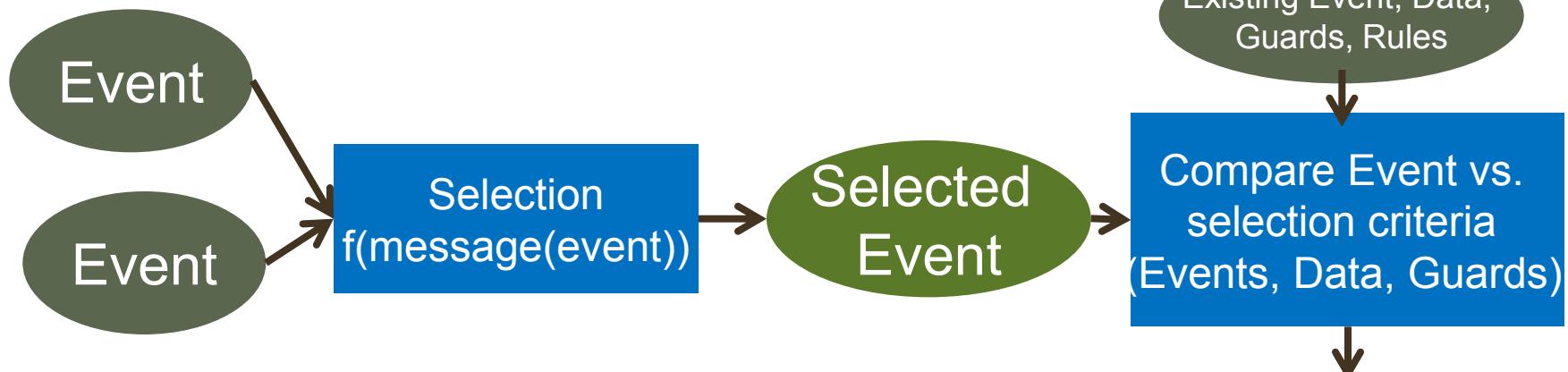
Preparation:Selection:Implementations:Prova

- **General pattern:**

<rcvMsg / rcvMult>

(CID, Protocol, Agent, Performativ, Event | Context)

:- <selection_processing> Event.



- **Effect:** select event according to message context (conversation id, protocol, agent, pragmatic performativ, event context) (see also monitoring) and further process selected events, e.g. comparing them against existing events, data, guards, rules, selection functions

Preparation:Selection:Implementations:Prova

Messaging Reaction Rules in Prova

- Send a message

sendMsg(CID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- Receive a message

rcvMsg(CID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- Receive multiple messages

rcvMult(CID,Protocol,Agent,Performative,[Predicate|Args]|Context)

Selection according to:

- **CID** is the conversation identifier
- **Protocol**: protocol e.g. self, jms, esb etc.
- **Agent**: denotes the target or sender of the message
- **Performative**: pragmatic context, e.g. FIPA ACL
- **[Predicate|Args]|Context** or **Predicate(Arg₁,...,Arg_n)**: Event

Preparation:Selection:Implementations:Prova Example

```
% Select stock ticker events from stream "S&P500"
%
% Each received event starts a new subconversation (CID)
% which further processes the selected event (select)

rcvMult(CID,stream,"S&P500", inform, tick(S,P,T)) :-  

  select(CID,tick(S,P,T)).  

%
% Indefinitely (count=-1) receive further ticker events from other streams that follow the previous selected event in event processing group (group=g1). If the price differs for the same stock at the same time [T1=T2, P1!=P2] then ...
select(CID,tick(S,P1,T1)) :-  

  @group(g1) @count(-1)  

rcvMsg(CID,stream, StreamID ,inform, tick(S,P2,T2))  

[T1=T2, P1!=P2] %guard: if at same time but different price,  

println ( ["Suspicious:",StreamID, tick(S,P2,T2)], " " ).
```

RA:Preparation:Filter



Filter: filter out all events that have some property in their payload / data

- E.g. customer purchases: filter out those with values < \$100

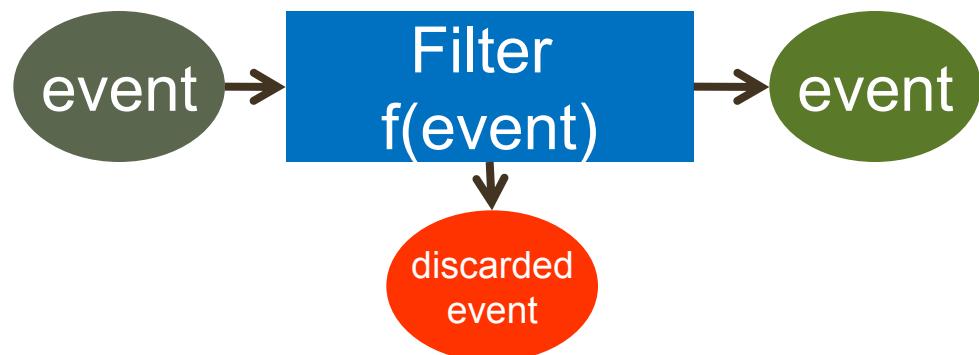
Preparation:Filter:Classification

- **EPTS Reference Architecture:**
 - *"During event preparation, a stream or list of events may be filtered on some payload or metadata information such that some subset is selected for further processing."*

Preparation:Filter:Description

- **Role / Explanation**
 - Comparing some property of the event (an attribute or metadata) with some other value (from some other event, or data)
- **Associated Business Rule Specification**
 - As a constraint: a selection (to be enforced during the processing of events):

*All <event entities>
that have
<filter expression>
must have
<some status>.*



Preparation:Filter:Structure

- **EPTS Glossary comparison**

- A filter pattern can relate to several terms in the EPTS Glossary
 - can be specified in an event pattern (A template containing event templates, relational operators and variables...)
 - .. by an event pattern constraint (A Boolean condition that must be satisfied by the events observed in a system...)
 - .. as part of an event processing rule (A prescribed method for processing events.)
 - .. or as part of event stream processing (Computing on inputs that are event streams.)

—

Preparation:Filter:Implementations: Prova: rules

- **Implementation type:** backward reasoning logical filter
- **General pattern template:**

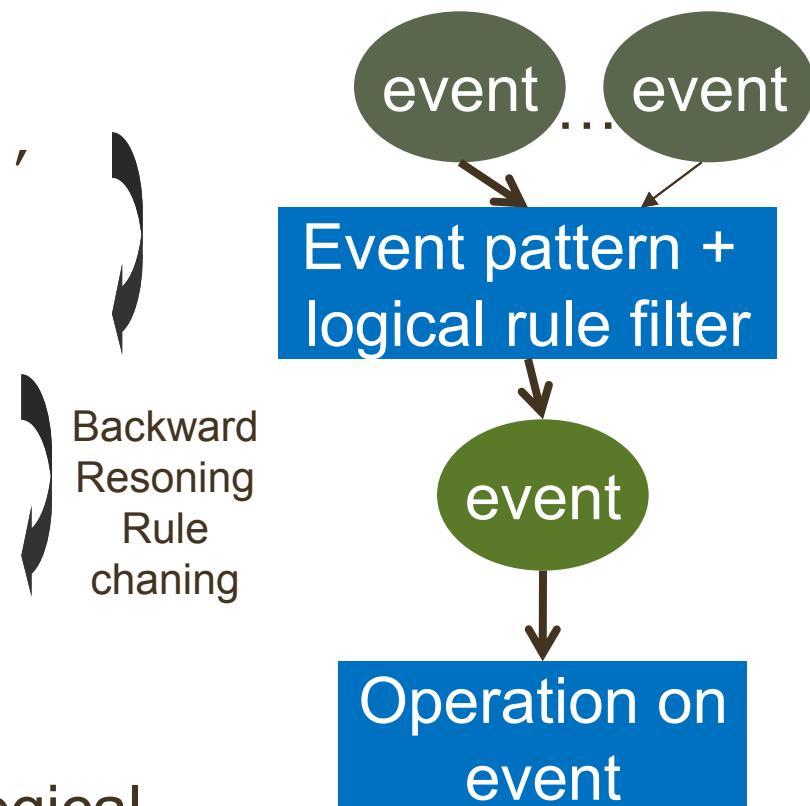
```
<rcvMsg / rcvMult> Event :-  
  logical filter condition(s),  
  ... .
```

```
logical derivation rule 1 :-  
  filter conditions.
```

```
logical derivation rule 2 :-  
  filter conditions.
```

...

- **Effect:** Unify received events with event pattern definition and apply logical filter rules (derivation rules with rule chaining)



Preparation:Filter:Implementations: Prova: Messaging Reaction Rules in Prova

- **Send a message**

sendMsg(XID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- **Receive a message**

rcvMsg(XID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- **Receive multiple messages**

rcvMult(XID,Protocol,Agent,Performative,[Predicate|Args]|Context)

Description:

- *XID* is the conversation identifier
- *Protocol*: protocol e.g. self, jms, esb etc.
- *Agent*: denotes the target or sender of the message
- *Performative*: pragmatic context, e.g. FIPA Agent Communication
- *[Predicate|Args]* or *Predicate(Arg₁,...,Arg_n)*: Message payload

Preparation:Filter:Implementations: Prova: Example

```
% Filter for stocks starting with „A“ and price > 100

recvMult(SID, stream, "S&P500", inform, tick(S, P, T)) :-  

  S = "A.*",  

  P > 100,  

  sendMsg(SID2, esb, "epa1", inform, happens(tick(S, P), T)).
```

Example with rule chaining



```
% Filter for stocks starting with „A“ and price > 100

recvMult(SID, stream, "S&P500", inform, tick(S, P, T)) :-  

  filter(S, P),  

  sendMsg(SID2, esb, "epa1", inform, happens(tick(S, P), T)).  
  

filter(Symbol, Price) :-  

  Price > 100,  

  Symbol = "A.*".
```

Preparation:Filter:Implementations:

Prova: Semantic Event Processing Example

Semantic Query Filer:

Stocks of companies, which have **production facilities** in **Europe** and produce products **out of metal** and Have more than **10,000 employees**.

Event Stream

```
{(Name, "OPEL")(Price, 45)(Volume, 2000)(Time, 1) }
{(Name, "SAP")(Price, 65)(Volume, 1000) (Time, 2)}
```

```
{(OPEL, is_a, car_manufacturer),
 (car_manufacturer, build, Cars),
 (Cars, are_build_from, Metall),
```

```
(OPEL, hat_production_facilities_in, Germany),
 (Germany, is_in, Europe)
```

```
(OPEL, is_a, Major_corporation),
```

```
(Major_corporation, have, over_10,000_employees)}
```

Semantic Knowledge Base

```
rcvMult(SID, stream, "S&P500", inform,
tick(Name^^car:Major_corporation, P^^currency:Dollar,
T^^time:Timepoint)) :- ... <semantic filter inference> .
```

RA:Preparation:Monitoring:



Monitoring: particular event channels are monitored to identify events of interest

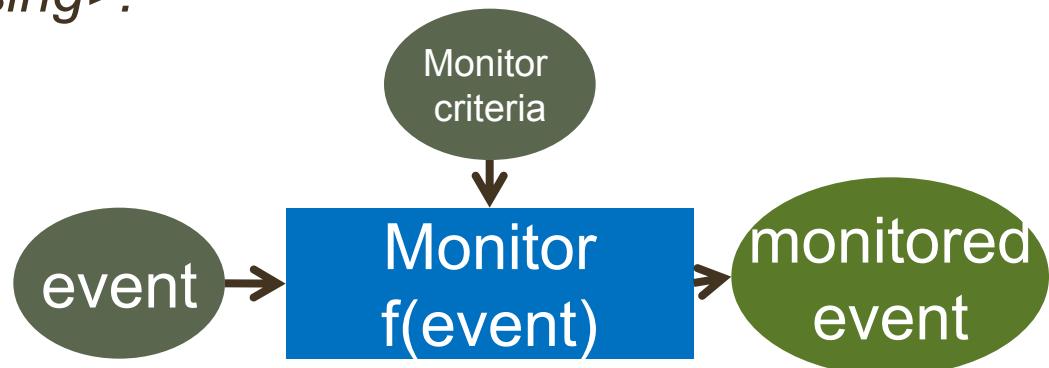
- E.g. Selecting a particular channel and event selector in some middleware subscription

Preparation:Monitoring:Classification

- **Alternative Names:**
 - Event Subscribe
 - See also Event Selection
- **EPTS Reference Architecture:**
 - *“During event preparation, 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.”*

Preparation:Monitoring:Description

- **Role / Explanation**
 - Directing an external agent to select events
 - Applying some monitor function over some event channel
- **Associated Business Rule Specification**
 - a monitor (of an event source):
All <event entities> whose <attribute1> matches the <attribute2> of <some monitoring criteria> is selected for <processing>.



Preparation:Monitoring:Structure

- **EPTS Glossary comparison**

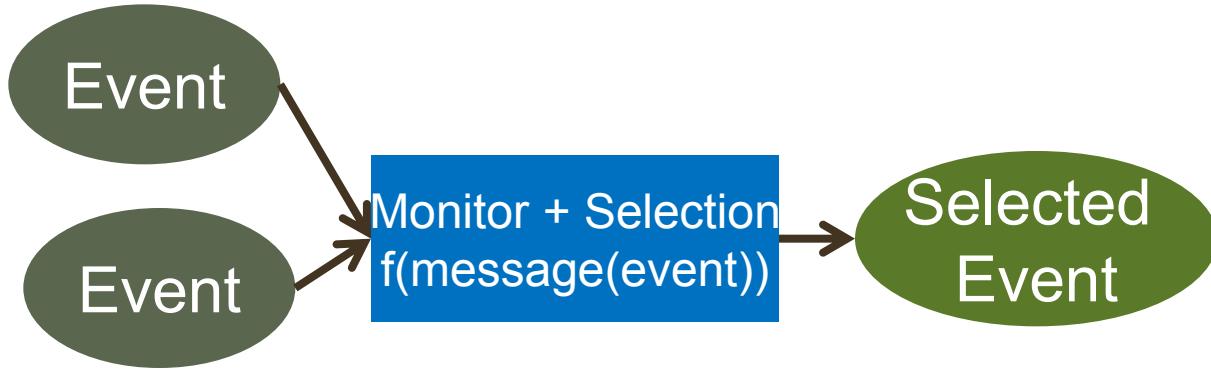
- (Example mechanism): Publish-and-subscribe (pub-sub)
 - A method of communication in which messages are delivered according to subscriptions .
 - Note 1. Subscriptions define which messages should flow to which consumers.
 - Note 2. Event processing applications may use publish-and - subscribe communication for delivering events. However, publish - and - subscribe is not definitional to event processing – other communication styles may be used.
- Subscriber
 - An agent that submits a subscription for publish- and – subscribe communication.

Preparation:Monitoring:Implementations: Prova: Selection

- General pattern:

`<rcvMsg / rcvMult>`

`(CID, Protocol, Agent, Performativ, Event | Context)`
`:- <selection_processing> Event .`



- Effect: monitor message channels and select (see also Selection Pattern) events according to message context (conversation id, protocol, agent, pragmatic performativ, event context);

Preparation:Monitoring:Implementations: Prova: Messaging Reaction Rules in Prova

- **Send a message**

sendMsg(CID,Protocol,Agent,Performative,[Predicate|Args]|Context)

- **Receive a message**

rcvMsg(CID,Protocol,Agent,Performative,[Predicate|Args]|Context)

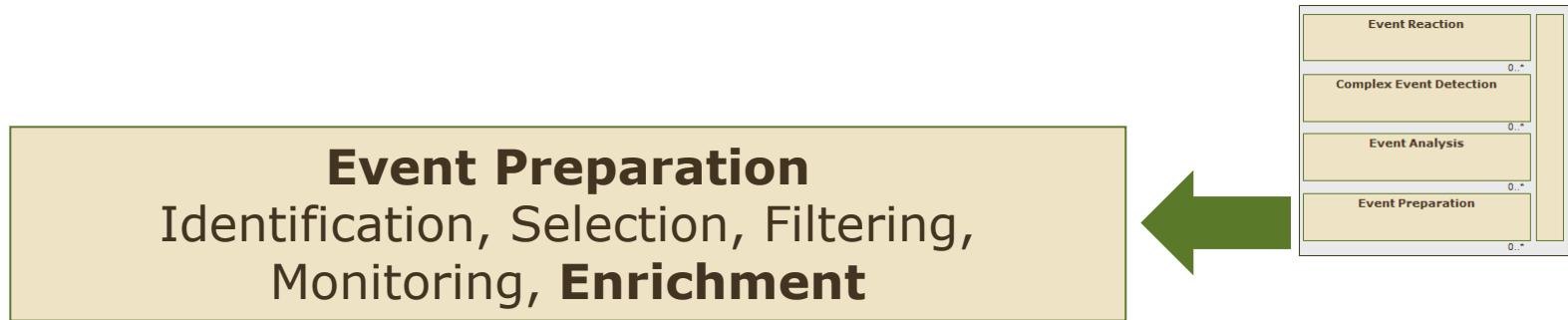
- **Receive multiple messages**

rcvMult(CID,Protocol,Agent,Performative,[Predicate|Args]|Context)

Selection according to:

- **CID** is the conversation identifier
- **Protocol**: protocol e.g. self, jms, esb etc.
- **Agent**: denotes the target or sender of the message
- **Performative**: pragmatic context, e.g. FIPA ACL
- **[Predicate|Args]|Context** or **Predicate(Arg₁,...,Arg_n)**: Event

RA:Preparation:Enrichment:



Enrichment: add some information based on prior events

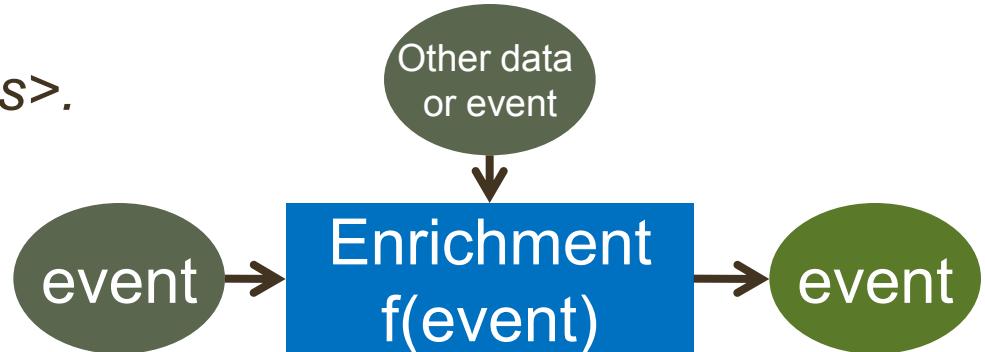
- E.g. customer purchase: add the customer history to the purchase event
- E.g. enrich the event with semantic background knowledge

Preparation:Enrichment:Classification

- **Alternative Names:**
 - Event Annotation, Semantic Event
- **EPTS Reference Architecture:**
 - *"During event preparation, events may be "enriched" through knowledge gained through previous events or data (including semantic background knowledge). "*

Preparation:Enrichment:Description

- **Role / Explanation**
 - Updating some attribute or metadata of the event with some other values, possibly computed, from some other event or data
- **Associated Business Rule Specification**
 - a selection (to be enforced during the processing of events):
All <event entities> that are also <enriched data expression> must have <some status>.



Preparation:Enrichment:Structure

- **EPTS Glossary comparison**
 - Enrichment can be specified in an event pattern (pattern definition containing additional meta data, semantic knowledge)
 - .. as part of an event processing rule (A method for enriching events during processing, e.g., by analysing the meta data and querying external data sources or inferencing semantic knowledge bases)
 - .. or as part of event stream processing (additional additional information to event stream data, e.g. by querying external data sources)

Preparation:Enrichment:Implementations: Prova: rules

- **General pattern template:**

rcvMsg / **rcvMult** <Event Msg Pattern(s)> :-

<query built-in(s)>,

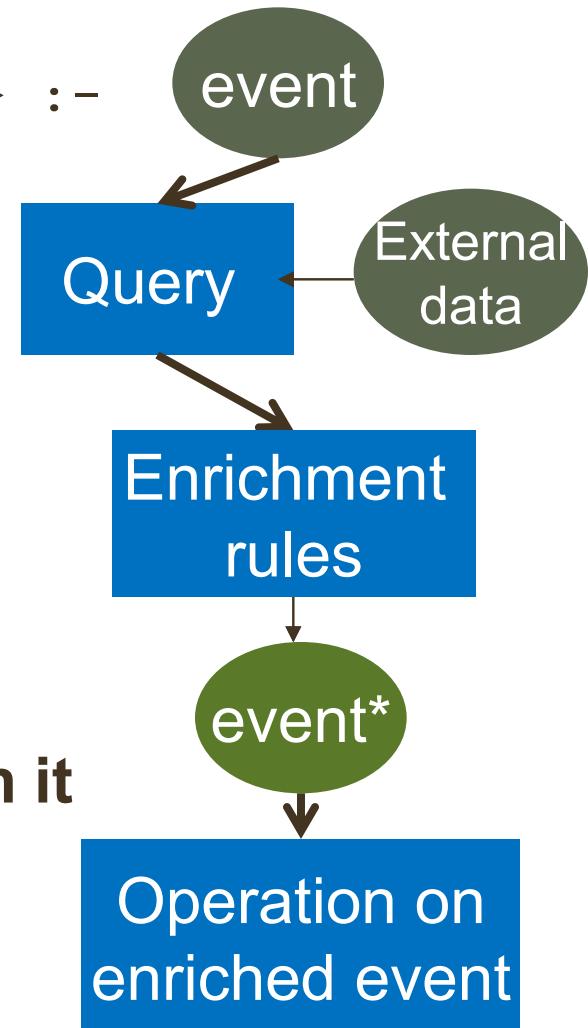
<enrichment condition(s)>

... .

<logical enrichment rule 1> :-

<enrichment operations>.

- **Effect: query data or knowledge from external source and enrich the event with it**



Preparation:Enrichment:Implementations: Prova: External Data and Object Integration

- **Java**

```
..., L=java.util.ArrayList(), ... , java.lang.String("Hello")
```

- **File Input / Output**

```
..., fopen(File, Reader), ...
```

- **XML (DOM)**

```
document(DomTree, DocumentReader) :-  
    XML(DocumentReader),
```

- **SQL**

```
..., sql_select(DB, cla, [pdb_id, "1alx"], [px, Domain]).
```

- **RDF**

```
..., rdf(http://.../rdfs, Subject, "rdf_type", "genel_Gene"),
```

- **XQuery**

```
..., XQuery = ' for $name in  
StatisticsURL//Author[0]/@name/text() return $name',  
xquery_select(XQuery, name(ExpertName)),
```

- **SPARQL**

```
..., sparql_select(SparqlQuery, name(Name), class(Class),  
definition(Def)),
```

Preparation:Enrichment:Implementations:

Prova: Example with SPARQL Query

```
% Filter for car manufacturer stocks and enrich the stock tick
event with data from Wikipedia (DBpedia) about the manufacturer
and the luxury cars

recvMult(SID,stream,"S&P500", inform, tick(S,P,T)) :-  

  carManufacturer(S,Man), % filter car manufacturers  

  findall([luxuryCar|Data], luxuryCar(Man,Name,Car),Data), % query  

  EnrichedData = [S,Data], % enrich with additional data  

  sendMsg(SID2,esb,"epa1", inform, happens(tick(EnrichedData,P),T)).  

% rule implementing the query on DBpedia using SPARQL query
luxuryCar(Manufacturer,Name,Car) :-  

  Query="SELECT ?manufacturer ?name ?car    % SPARQL RDF Query  

    WHERE { ?car <http://purl.org/dc/terms/subject>  

           <http://dbpedia.org/resource/Category:Luxury\_vehicles> .  

           ?car foaf:name ?name .  

           ?car dbo:manufacturer ?man .  

           ?man foaf:name ?manufacturer.  

         } ORDER by ?manufacturer ?name",  

  sparql_select(Query,manufacturer(Manufacturer),name(Name),car(Car)).
```

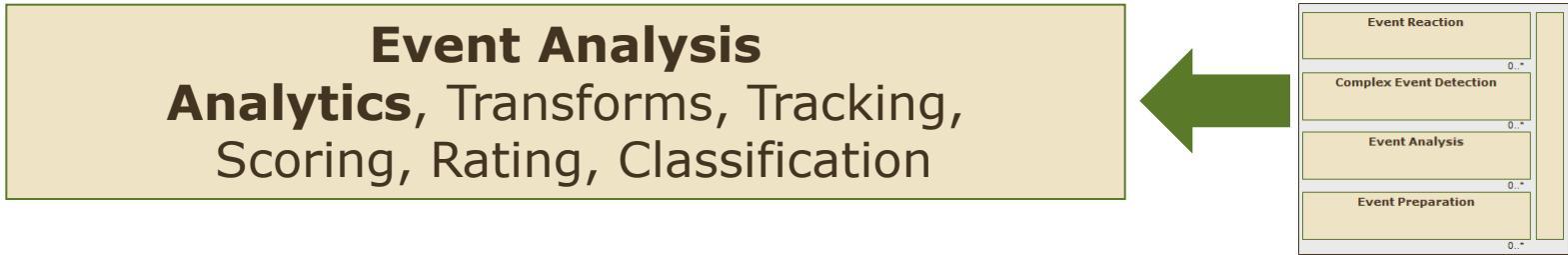
EP Patterns:RA:Analysis



Event Analysis is the process of analysing suitably prepared events and their payloads and metadata for useful information.

For example, event analysis may involve the identification of events against existing entities.

RA:Analysis:Analytics



Analytics: use of mathematical methods like statistics to derive additional information about an event or set of events

- E.g. standard deviation of the response time of a system, used to give a quantitative comparison of current with past performance of a system

Analysis:Analytics:Classification

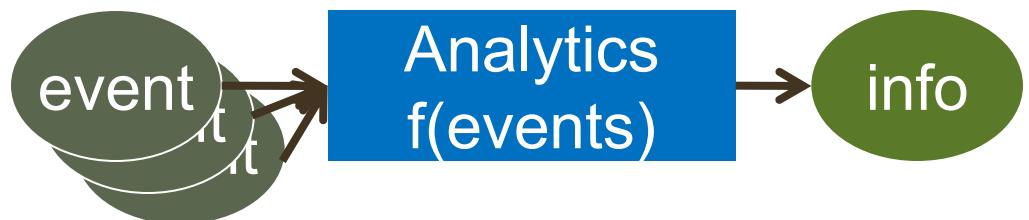
- **Alternative Names:**
 - Event Statistics
- **EPTS Reference Architecture:**
 - *“Event Analytics are the use of statistical methods to derive additional information about an event or set of events.”*

Analysis:Analytics:Description

- **Role / Explanation**
 - Analytics is where multiple data items (in a single event, or more commonly across multiple events), are analysed mathematically to define some additional information
 - For example: an average or mean value of event data can be used to compare against subsequent events to indicate some trend (see Learning)

- **Associated Business Rule Specification**

- an event analytics (event processing) process:
The <analytic fact> is provided by <analytic function> applied to <events>



Analysis:Analytics:Structure

• EPTS Glossary comparison

- **Cause:** An event A is a cause of another event B, if A had to happen in order for B to happen.
- **Event abstraction:** The relationship between a complex event and the other events that it denotes, summarizes, or otherwise represents.
- **Derived event (synthesized event, synthetic event):** An event that is generated as a result of applying **a method or process to one or more other events**.
- **Relationships between events:** Events are related by **time, causality, abstraction, and other relationships**. Time and causality impose partial orderings upon events.
- **Event pattern:** A template containing event templates, **relational operators and variables**.
- **Constraint (event pattern constraint):** A Boolean **condition** that must be satisfied by the events observed in a system.

Analysis:Analytics:Implementations:

Prova: Rules

- General pattern:

```
rcvMsg / rcvMult <Event Msg Pattern(s)> :-  

    <query (past) data>,  

    <analytics functions(s)  

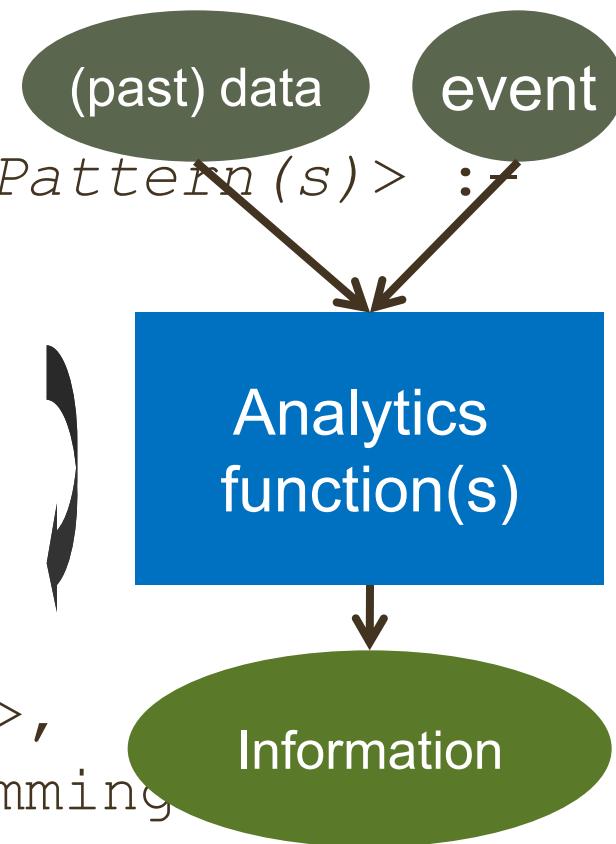
        with data>  

    . . .
```

```
<analytics rule 1> :-  

    <Java call to analytics API>,  

    <or Prova functional programming>
```



- Effect: event-driven statistic function (eg update of mean for a given event type or other condition)

Analysis:Analytics:Implementations: Prova: Java Programming and Functional Programming

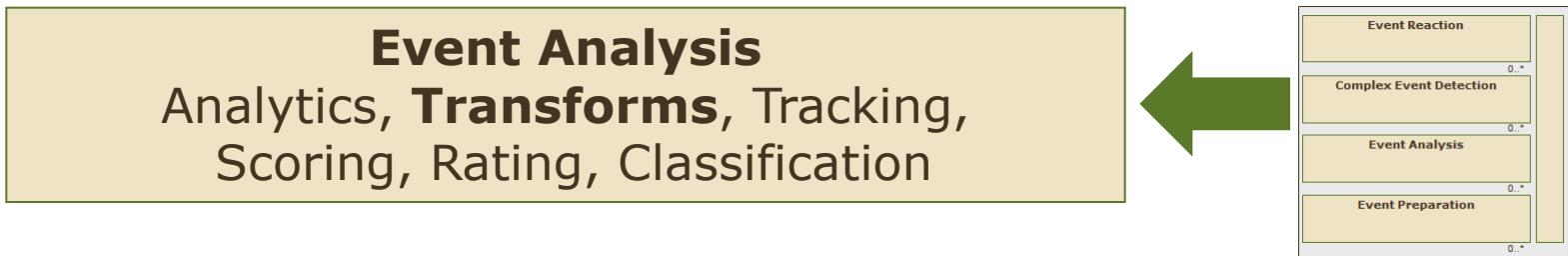
Prova Dynamic Java Programming

- Ability to embed Java calls
... :- `x = java.lang.Math.abs(2.99), x+1, ...`
- Java constructors, object instance + methods and static methods, and public field access; (+ Java exception handling)

Prova Functional Programming

- single- and multi-valued functions,
- functional composition with the extended *derive* built-in;
- partial evaluation;
- lambda functions;
- monadic functions;
- monadic *bind* using a composition of *map* and *join*;
- *maybe*, *list*, *state*, *tree* and *fact* monads

RA:Analysis:Transforms



Transforms: processes carried out on events' payloads or data, such for as standardising schema or information types

- E.g. an insurance application event from a custom IT system may have the customer information translated into an ACORD-type structure for onward processing

Analysis:Transforms:Classification

- **Alternative Names:**
 - Event Processes, Event Operations
- **EPTS Reference Architecture:**
 - *“Event Transforms are processes carried out on event payloads or data, either related to event preparation, analysis or processing.”*

Analysis:Transforms:Description

- **Role / Explanation**
 - Transforms are applied to events (event data) to process into new, derived events or event data
 - For example: a transform may reformat an input data schema into an output schema format
- **Associated Business Rule Specification**
 - an event transform:
The <derived fact> is derived from <input event> when <transform fn> is applied



Analysis:Transforms:Structure

- **EPTS Glossary comparison**

- **Derived** event (synthesized event, synthetic event): An event that is generated as a result of applying **a method or process to one or more other events**.

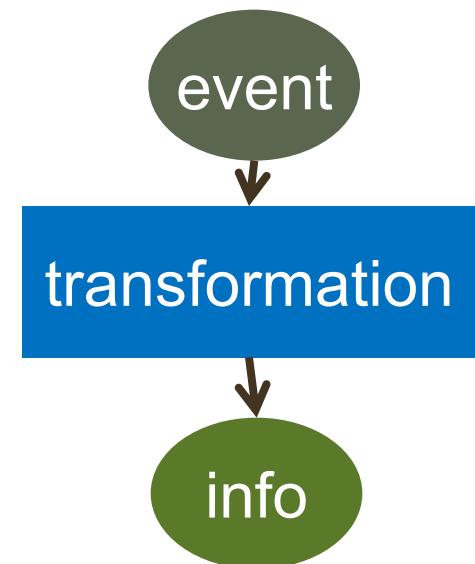
Analysis:Transforms:Implementations: Prova: Rules

- General pattern:

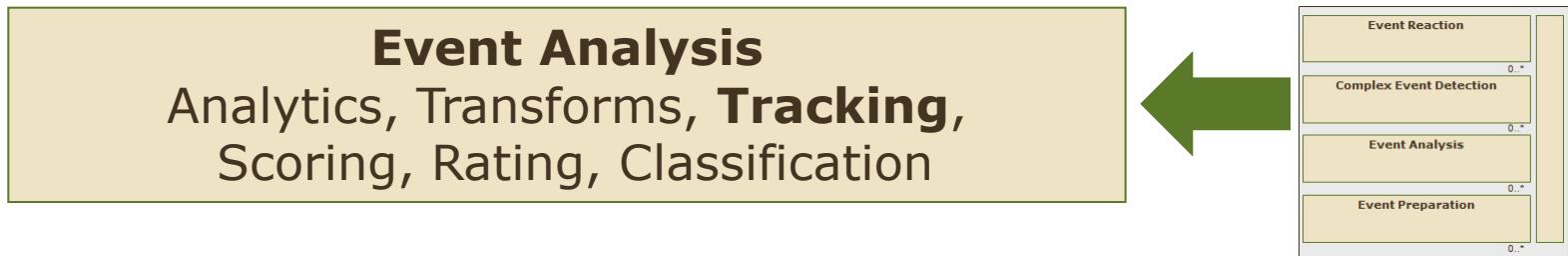
```
rcvMsg / rcvMult <Event Msg  
Pattern(s)> :-  
    <transformation function  
(Output, Input)>,  
    . . . .
```

```
<transformation rule 1> (Output,  
Input)   :-  
    <transformation logic >
```

- Effect: transforms selected properties into new properties, applying function if needed.



RA:Analysis:Tracking



Tracking: use of events to follow some related entity's state
(such as in space, time or process status)

- E.g. “track and trace” of the location of airline baggage

Analysis:Tracking:Classification

- **Alternative Names**
- **EPTS Reference Architecture:**
 - “*Event Tracking is where events related to some entity are used to identify state changes in that entity.*”
 - *Typically state is related to geolocation, business process, etc.*

Analysis:Tracking:Description

- **Role / Explanation**

- Tracking is applied to entities via events (event data) to maintain information about the entity state (eg location).
- For example: a patient may be tracked by active and passive sensors creating movement and activity events in a care facility
- Usually associated with the term “track and trace”

- **Associated Business Rule Specification**

- an track operation:

The <entity> is tracked to <state>

whenever

<event> occurs



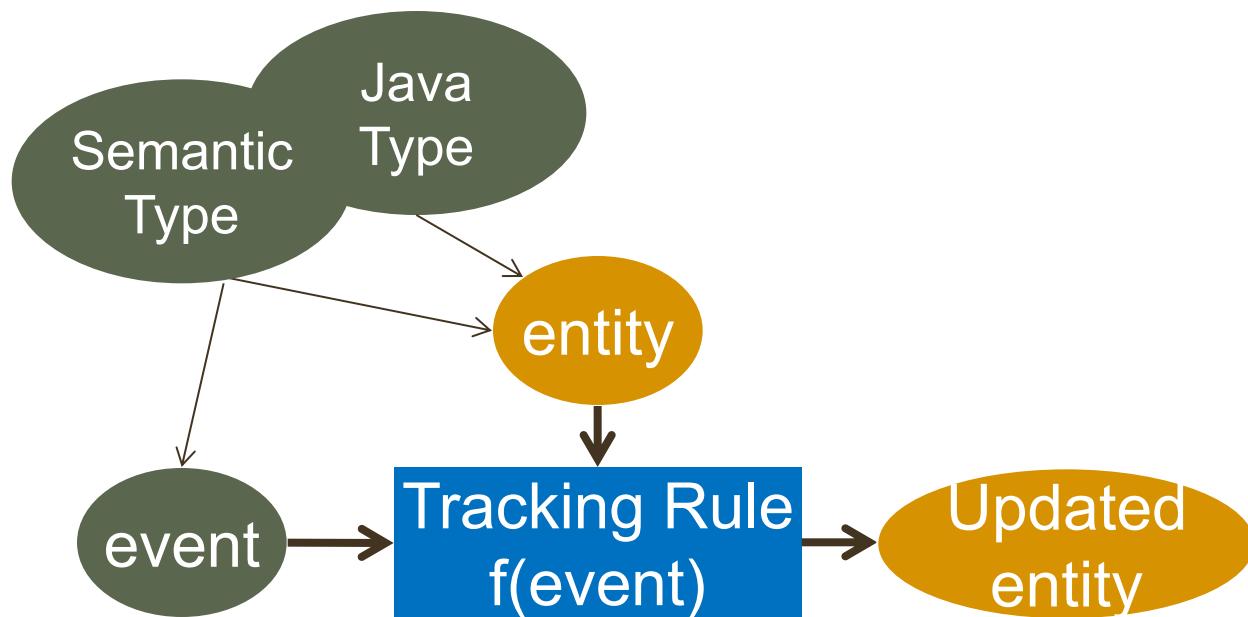
Analysis: Tracking: Structure

- **EPTS Glossary comparison**

- **Derived** event (synthesized event, synthetic event): An event that is generated as a result of applying **a method or process to one or more other events**.

Analysis: Tracking: Implementations

Prova: Rules

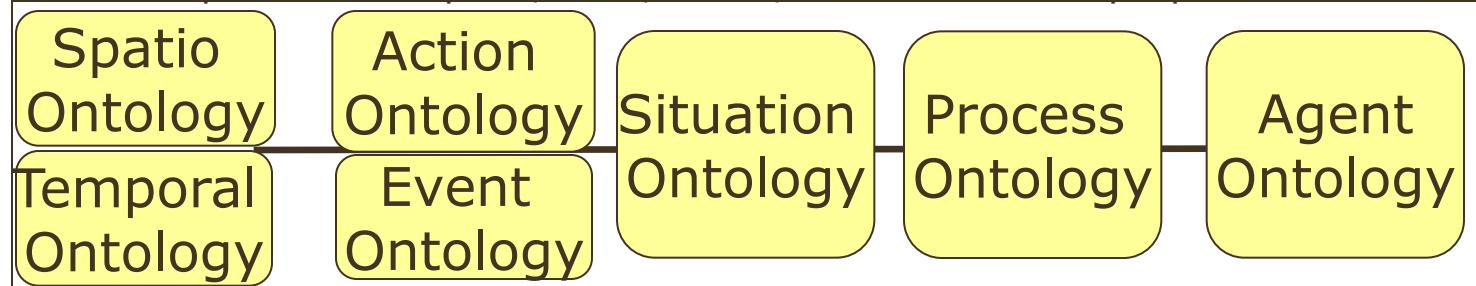


Reaction RuleML Metamodel

<http://www.slideshare.net/swadpasc/reaction-ruleml-ruleml2012paschketutorial>

Top Level Ontologies

General concepts such as space, time, event, action and their properties and relations



Vocabularies **related to specific domains** by specializing the concepts introduced in the top-level ontology

Domain Ontologies

Task Activities Ontologies

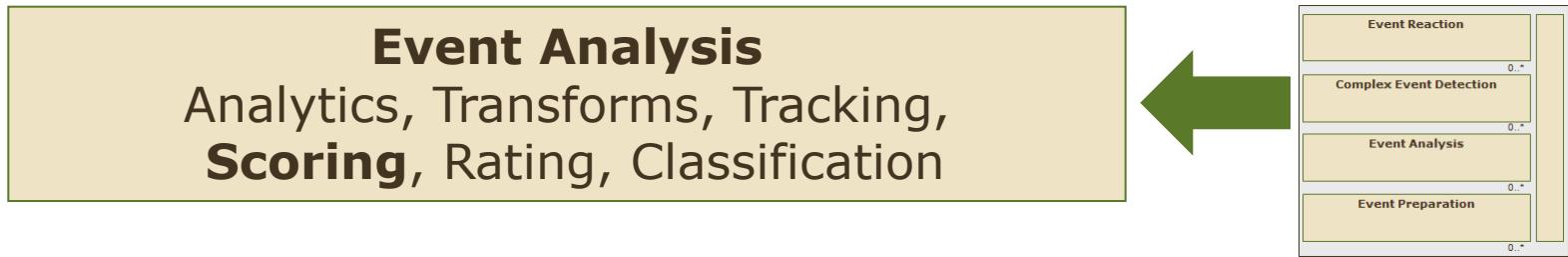
Specific
user/application
ontologies

Application Ontologies

Vocabularies **related to generic tasks or activities** by specializing the concepts introduced in the top-level ontology

E.g. ontologies describing roles played by domain entities while performing application activities

RA:Analysis:Scoring



Scoring: ranking events or event data based on some predefined criteria

- E.g. scoring a customer account application based on weighting specific aspects of the application to give an overall “score”
- See Score Modeling and Predictive Analytics

Analysis:Scoring:Classification

- **Alternative Names**
- **EPTS Reference Architecture:**
 - “*Event Scoring is the process by which events are ranked using a score, usually as a part of a statistical analysis of a set of events*”.
 - Note: Possibly this should be subsumed into Event Analytics

Analysis:Scoring:Description

- **Role / Explanation**

- Scoring is applied to events or entities by associating score values to attributes of the event or entity.
- For example: an intelligence report event may be scored against its attributes (source, time, subject etc) to give an overall indication (score, for example as a %) of its reliability
- Usually associated with predictive analytics

- **Associated Business Rule Specification**

- a score operation on an event:

The <entity> is given a score <s> by applying transform model <m>



Analysis:Scoring:Structure

- **EPTS Glossary comparison**

- **Derived event** (synthesized event, synthetic event): An event that is generated as a result of applying **a method or process to one or more other events**.
- **Relationships** between events: Events are related by **time, causality, abstraction, and other relationships**. Time and causality impose partial orderings upon events.
- **Event pattern**: A template containing event templates, **relational operators and variables**.
- **Constraint** (event pattern constraint): A Boolean **condition** that must be satisfied by the events observed in a system.

Analysis: Scoring: Implementations:

Prova: Rules

- General pattern:

```
rcvMsg / rcvMult <event> :-  

  @score(Old) <event>,  

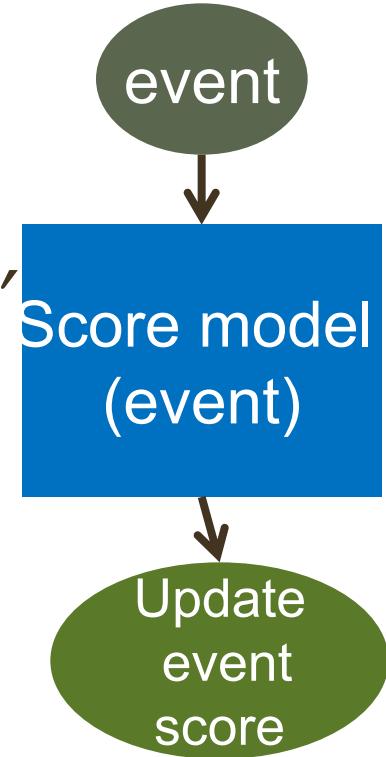
  <score function (event, Old, New)>,  

  <update> @score(New) <event>.  

<score function rule>(E, O, N) :-  

  <compute new score>.
```



- Effect: events' metadata score (and other metadata or event data properties) are applied to score functions and then combined into a single updated score as metadata of the event object

Analysis: Scoring: Implementations:

Prova: Example – Metadata Scoped Reasoning

```
% event instances with metadata @key(value, [,value]*)

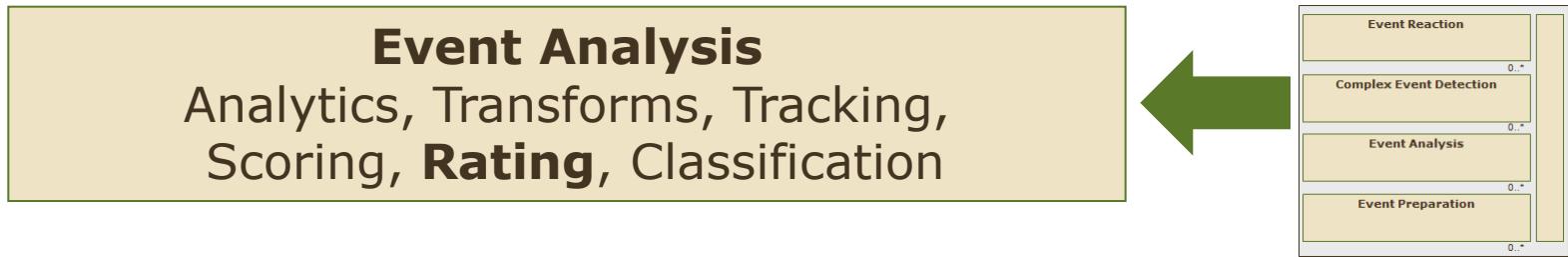
@score(1) @src(stream1) @label(e1) tick("Opel",10,t1).
@score(3) @src(stream2) @label(e2) tick("Opel",15,t1).

happens(tick(S,P),T) :-
    % scope defined over @score metadata bound to "Value"
    @score(Value) tick(S,P,T) [Value>2]. %guard Value > 2
```

```
@score(1) @src(stream1) @label(e1) tick("Opel",10,t1).
@score(3) @src(stream2) @label(e2) tick("Opel",15,t1).

happens(tick(S,P),T) :-
    % select ticks from „stream1“
    @score(Value) @src(stream1) @label(E) tick(S,P,T),
    NewValue = Value + 3, %add 3 to score „Value“
    %update selected events from „stream1“ with „NewValue“
    update(@label(E), @score(NewValue) tick(S,P,T)),
    @score(S) tick(S,P,T) [S>3]. %guard Value > 3
```

RA:Analysis:Rating



Rating: comparison of events or event data with some metric to provide an ordering

- E.g. trade transactions may be rated with respect to overall achieved profit

Analysis:Rating:Classification

- **Alternative Names**
- **EPTS Reference Architecture:**
 - “*Event Rating is where events are compared to others to associate some importance or other, possibly relative, measurement to the event*”.

Analysis:Rating:Description

- **Role / Explanation**
 - Rating provides a quantitative or ordering.
 - For example: an customer order shipment may be rated as “courier” based on the customer order details
 - Could be part of “classification”
 - Dictionary definition: *classification according to grade or rank*
- **Associated Business Rule Specification**
 - an rating operation on an event:
The <event> is rated to <rating> by <rating criteria>



Analysis:Rating:Structure

- **EPTS Glossary comparison**

- **Derived** event (synthesized event, synthetic event): An event that is generated as a result of applying **a method or process to one or more other events**.
- **Relationships** between events: Events are related by **time, causality, abstraction, and other relationships**. Time and causality impose partial orderings upon events.
- **Event pattern**: A template containing event templates, **relational operators and variables**.
- **Constraint** (event pattern constraint): A Boolean **condition** that must be satisfied by the events observed in a system.

Analysis: Scoring: Implementations: Prova: Rules

- **General pattern:**

see scoring

- **Effect:** rate events and select rated events



Analysis:Rating:Implementations: Prova: Metadata Scoped Reasoning and Guards

```
% stream1 is trusted but stream2 is not, so one  
solution is found: X=e1
```

```
@src(stream1) event(e1).  
@src(stream2) event(e2).
```

```
%note, for simplicity this is just a simple fact, but  
more complicated rating, trust, reputation policies  
could be defined
```

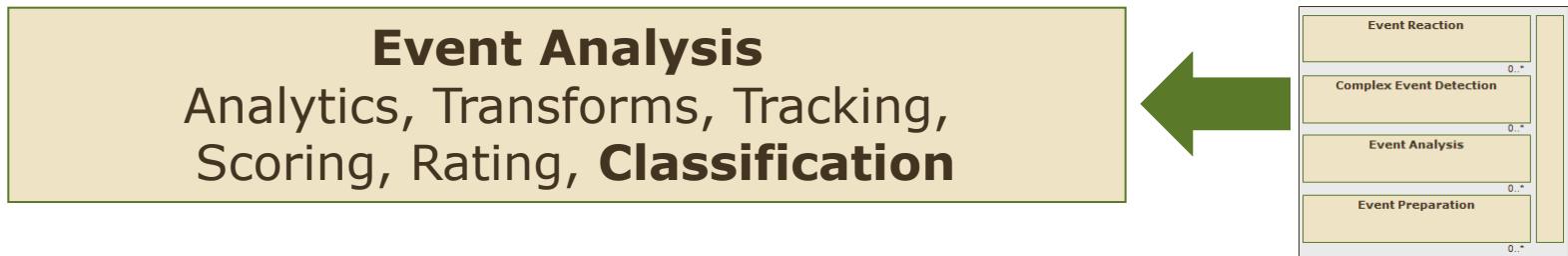
```
trusted(stream1).%only event from „stream1“ are trusted
```

```
ratedEvent(X) :-
```

```
  @src(Source) %scoped reasoning on @src  
  event(X) [trusted(Source)]. %guard on trusted sources
```

```
:-solve(ratedEvent(X)). % => X=e1 (but not e2)
```

RA:Analysis:Classification



Classification: comparison with and association of events with some classification scheme to which the event is applied

- E.g. classifying a credit card user as a “gold shopper” for marketing purposes

Analysis:Classification:Classification

- **Alternative Names**
- **EPTS Reference Architecture:**
 - “*Event Classification is where events are associated with some classification scheme (for use in downstream processing)*”.

Analysis:Classification:Description

- **Role / Explanation**

- Classification provides associates an event to a class or category.
- For example: an customer may be classified as “Gold customer” based on the customer order details
- *Discovery / learning may be involved in the definition of classifications*

- **Associated Business Rule Specification**

- an event is classified:
*The <event> is classified as <class>
by <classification criteria>*



Analysis:Classification:Structure

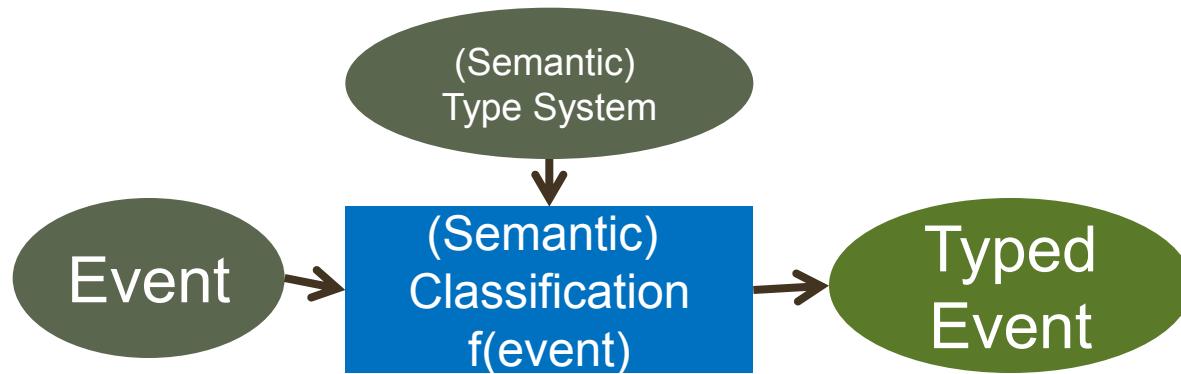
- **EPTS Glossary comparison**

- **Derived** event (synthesized event, synthetic event): An event that is generated as a result of applying **a method or process to one or more other events**.
- **Relationships** between events: Events are related by **time, causality, abstraction, and other relationships**. Time and causality impose partial orderings upon events.
- **Event pattern**: A template containing event templates, **relational operators and variables**.
- **Constraint** (event pattern constraint): A Boolean **condition** that must be satisfied by the events observed in a system.

Analysis:Classification:Implementations: Prova: Rules

- **General pattern:**

```
<rcvMsg / rcvMult> Event ^^ Semantic Type :-  
    <conditional type discovery>,  
    derivedEvent ^^ Semantic Type.
```



- **Effect:** classify event according to a (semantic) type system.
Discover event type (may include learning).

Analysis:Classification:Implementations: Prova: Rules - Example



Event Stream

{(Name, "OPEL")(Price, 45)(Volume, 2000)(Time, 1) }



```
rcvMult(SID, stream, "S&P500", inform,
    tick(Name^^Car:Manufacturer, P, T)) :-  

    size(Name, Employees), Employees > 10000, %type discovery  

    sendMsg(SID, self, 0, inform, tick(Name^^Major_Cooperation, P, T))
```

Classification according to
Type System

Classification by Type Discovery
(may also include Type Learning)



Semantic
Knowledge Base
(T-Box Model is
used as Type
System)

```
{(OPEL, is_a, Car_Manufacturer),  

 (Car_Manufacturer, sameAs, Motor_Vehicle_Manufacturer),  

 (Car_Manufacturer, subClassOf, Automotive_Company),  

 (Automotive_Company, sameAs, Automotive_Corporation),  

 (Automotive_Company, subClassOf, Automotive_Industry),  

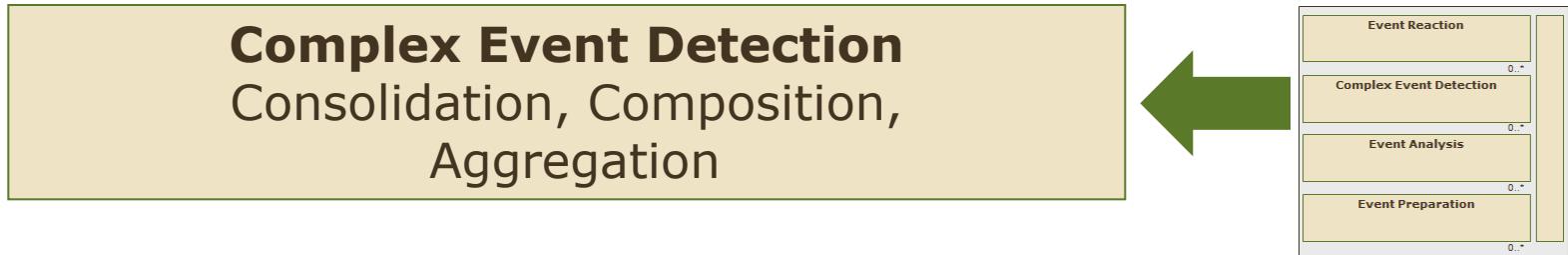
 (Automotive_Corporation, subClassOf, Corporation)  

 (Major_Corporation, have, over_10,000_employees),  

 (Major_Corporation, subClassOf, Corporation)}  

.
```

EP Patterns:RA:Detection

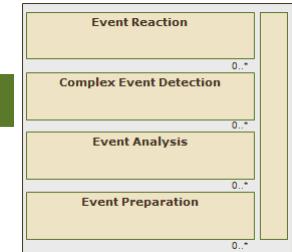


Complex Event Detection is the process by which event analysis results in the creation of new event information, or the update of existing complex events.

For example, complex event detection may result from identifying a pattern of event occurrences that indicate some complex event has taken or is taking place.

RA:Detection:Consolidation

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



Consolidation: combine events of similar type together into a new event or combine disparate events together into a main or primary event

- E.g. multiple events occur to indicate a complex event; the complex event is given a new consolidate identity as derived primary event or main event
- E.g. several messages occur that together form a complex event indicating a business transaction
- E.g. multiple events of the same type occur within a time period to indicate an event called “a high rate of events”
- E.g. abstraction of events into a situation which is initiated or terminated by the (derived complex) event as effect of the detection and consolidation of the complex event, e.g. the complex event “plane take-off” has the effect initiating the situation “flying”

Detection:Consolidation:Classification

- **Alternative Names:**
 - Event Reinforcement from Subevents, Event / Situation Reasoning
- **EPTS Reference Architecture:**
 - *"During complex event detection, combining events together into a "main" or "primary" event. Similar to event aggregation and composition", but with a **new explicit derived event** from the aggregated/composed events (the consolidated derived event might not contain all events from which it was derived);*
 - *a special abstraction of the **effect** of a (derived) event is a situation which is initiated or terminated by events*

Detection:Consolidation:Description

- **Role / Explanation**

- Consolidation is used to describe several events supporting the creation of a single composite event, generally where component events together provide evidence.
- For example: a “conflict” event (effect = situation “in conflict”) is considered to take place when both a “conflict declaration” occurs and an “act of violence” occurs.

- **Associated Business Rule Specification**

- a consolidation (to be enforced during event processing):
The <collection of events> supporting <event> that are <related by some relationship constraint>.



Detection:Consolidation:Structure

- **EPTS Glossary comparison**

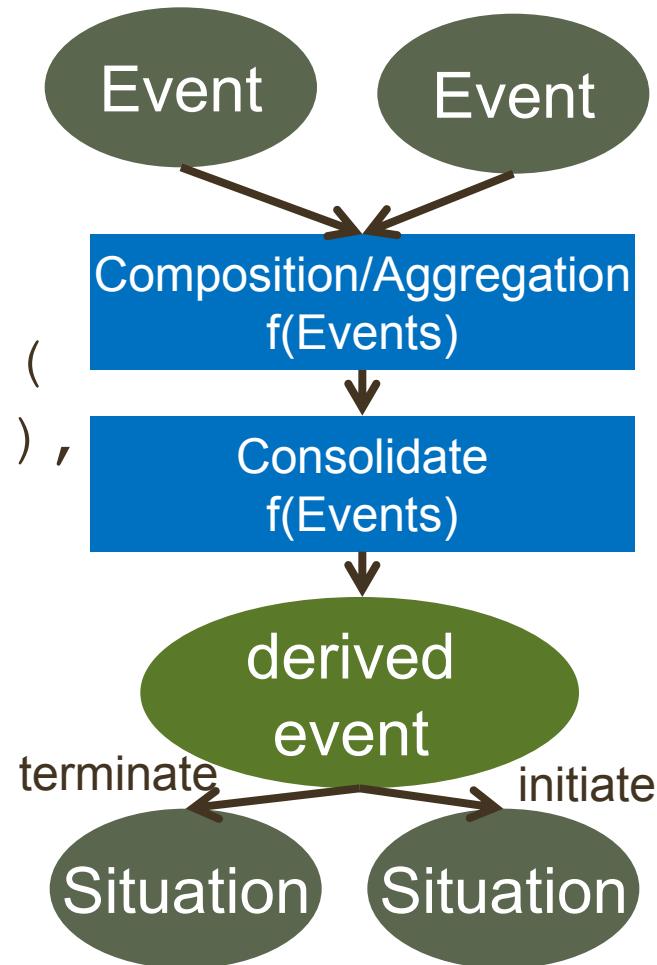
- The term consolidated event is sometimes used for some forms of composite or derived event.
- Composite event: a derived, complex event
 - is created by combining base events using a specific set of event constructors such as disjunction, conjunction, sequence, etc.
 - always includes the base (member) events from which it is derived.
- Derived event (/synthesized event): an event that is generated as a result of applying a method or process to one or more other events.

Detection:Consolidation:Implementations: Prova: Consolidation

- General pattern:

```
<rcvMult> Event :-  
    DerivedEvent = <consolidate>  
        <compose/aggregate> Events ,  
  
<initiate / terminate> (  
    DerivedEvent, Situation) .
```

- Effect:** a detected complex event (see composition, aggregation) is given a **new derived event identity**. The effect of the event occurrence / detection might initiate or terminate a situation.

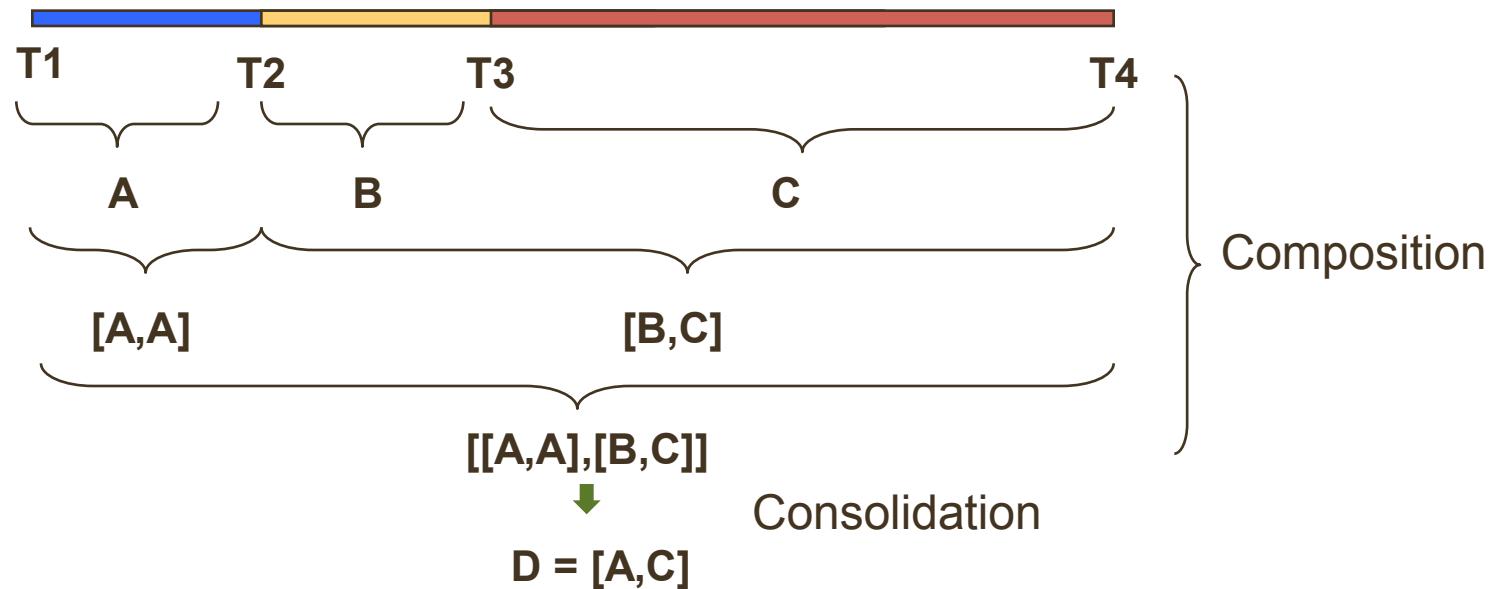


Detection:Consolidation:Implementations: Prova: Consolidation Example

- Interval-based Event Calculus semantics (model-theory + proof theory) based on **time intervals** modeled as **fluents**

$$I: T_{interval} \times FI \rightarrow \{\text{true, false}\}$$

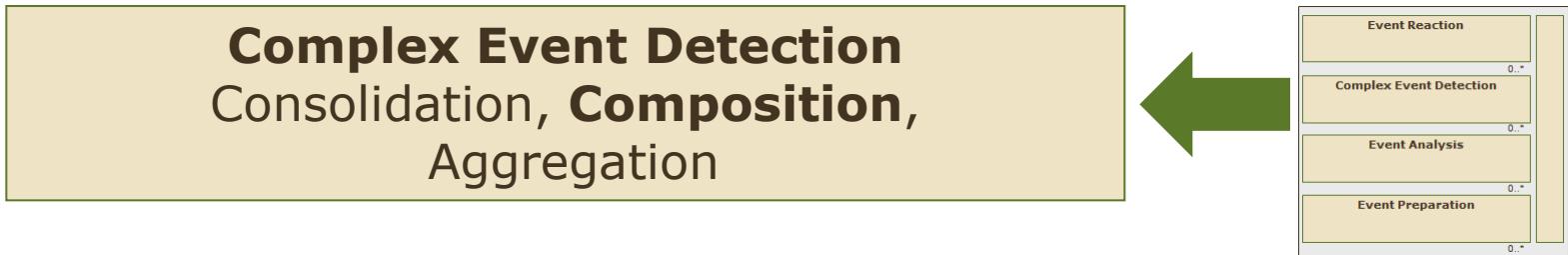
- Example: $D = A; (B; C)$ (*consolidation: derive event D from sequence composition*)



- Example: derived situation from complex event detection (consolidation: initiate situation1 by event D)

```
initiates (D, situation1, T) .
holdsAt (situation1, t5) ? => yes
```

RA:Detection:Composition



Composition: composing new complex events from existing, possibly source, events

- E.g. deduce a complex event from a history of past events

Interesting compositions

1. By type (see consolidation and aggregation)
2. By sequence, order, and missing events
3. By combination of operators
(e.g. sequence of increasing values until stops increasing)

Detection:Composition:Classification

- **EPTS Reference Architecture:**
- "Event composition is about composing new, complex events from existing, possibly source, events."
- *For example, event composition may take information from a range of past events to determine that a new, complex, event has occurred and needs to be composed.*

Detection:Composition:Description

- **Role / Explanation**
 - Composition is used to describe combining multiple events into a composite event, generally where the component events justify the existence of the complex event.
 - For example: a telephone call event is considered to take place when a call end event occurs after a call start event.
- **Associated Business Rule Specification**
 - a composition (to be enforced during event processing):
*The <collection> of <event entities> that
are <related by
some relationship
constraint>.*



Detection:Composition:Structure

- **EPTS Glossary comparison:**

- Composite event: A derived event that is created by combining a set of other simple or complex events (known as its members) using a specific set of event constructors such as disjunction, conjunction, and sequence.
- Event pattern: A template containing event templates, relational operators and variables. An event pattern can match sets of related events by replacing variables with values.
- Event pattern detection: Finding instances of an event pattern.
- Event template: An event form or descriptor some of whose parameters are variables. An event template matches single events by replacing the variables with values.

Detection:Composition:Implementations: Prova Complex Event Processing

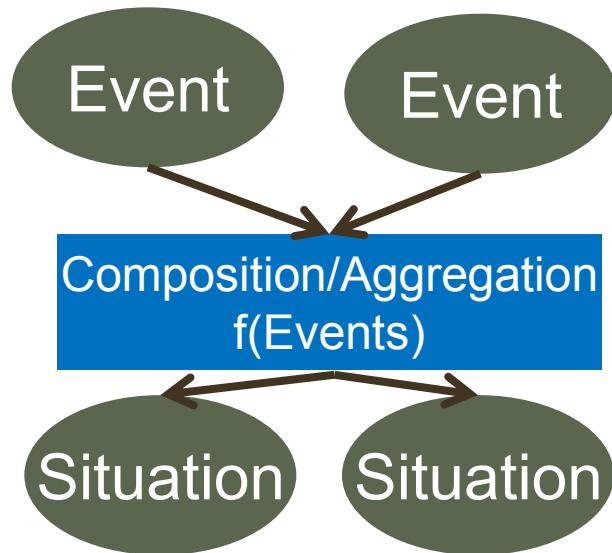
- General pattern:

<rcvMult> Event :-

<compose/aggregate>
Events

<initiate / terminate> (
DerivedEvent, Situation) .

- **Effect:** construct evidence for composed event from incoming events. Composed events might lead to a certain situation.

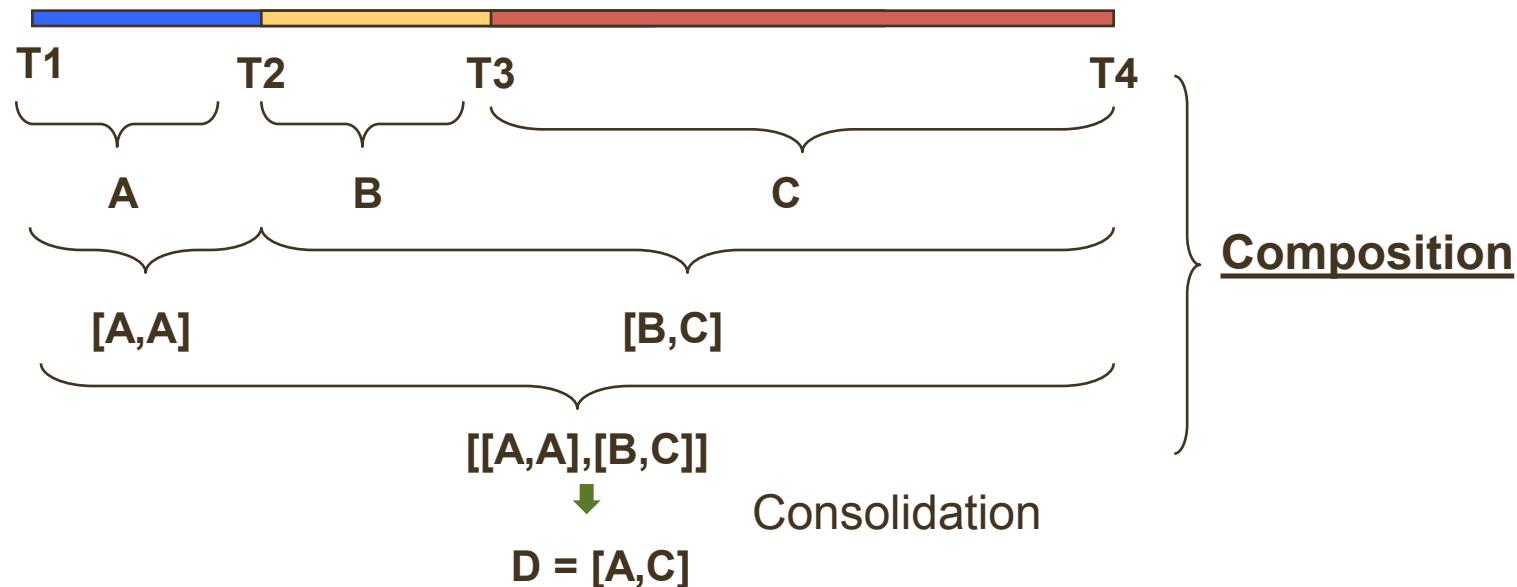


Detection:Consolidation:Implementations: Prova: Composition Example

- Interval-based Event Calculus semantics (model-theory + proof theory) based on time intervals modeled as fluents

$$I: T_{interval} \times FI \rightarrow \{\text{true, false}\}$$

- Example: $D = A; (B; C)$ (*consolidation: derive event D from sequence composition*)

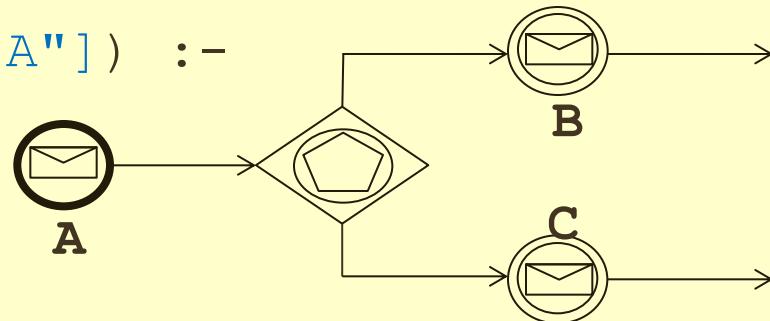


- Example: derived situation from complex event detection (consolidation: initiate situation1 by event D)

```
initiates(D, situation1, T).
holdsAt(situation1, t5). => yes
```

Detection:Composition:Implementations: Prova: Event Composition as Event-Driven Workflow

```
rcvMsg(XID, Process, From, event, ["A"]) :-  
    fork_b_c(XID, Process).
```



```
fork_b_c(XID, Process) :-
```

```
    @group(p1) rcvMsg(XID, Process, From, event, ["B"]), ... .
```

```
fork_b_c(XID, Process) :-
```

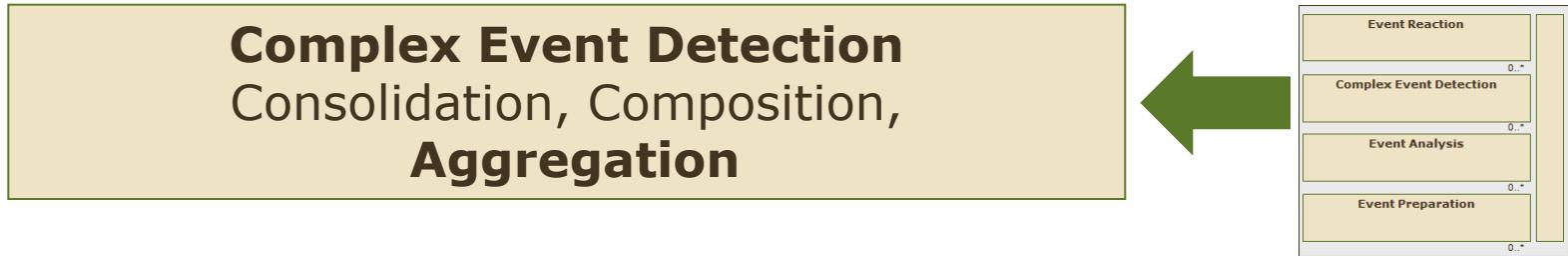
```
    @group(p1) rcvMsg(XID, Process, From, event, ["C"]), ... .
```

```
fork_b_c(XID, Process) :-
```

% OR reaction group "p1" waits for either of the two event message handlers "B" or "C" and terminates the alternative reaction if one arrives

```
@or(p1) rcvMsg(XID, Process, From, or, _) .
```

Detection:Aggregation



Aggregation: combine events of dissimilar type together into an event

- E.g. combine credit card purchase event with competitor store nearby not-busy event, and generate competitor offer event for a complementary product

Detection:Aggregation:Classification

- **Alternative Names:**
 - Event Summarization
- **EPTS Reference Architecture:**
 - *"During complex event detection, combining events to provide new or useful information, such as trend information and event statistics. Similar to event consolidation "*

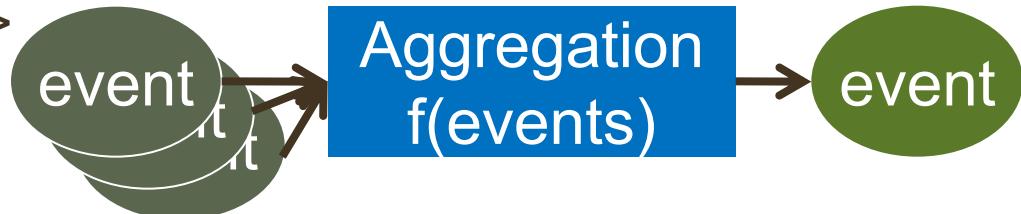
Detection:Aggregation:Description

- **Role / Explanation**

- Aggregation is where multiple events support the creation of a composite event, either as components or evidence, and are of different type.
- For example: a “start call” event aggregated with an “end call” event indicates a “telephone call event”
- For example: generate a (composite) event that contains the medium price of a stock over a 10 minute stream of events.

- **Associated Business Rule Specification**

- a summarisation (to be enforced during event processing):
*The <aggregation fn> of <event entities> that
are <selection constraint>
must have
<some constraint>.*



Detection:Aggregation:Structure

- **EPTS Glossary comparison:** see previous

Detection:Aggregation:Implementations: Prova: rules

- General pattern template:

```

<rule_head> :-  

  <Create Aggregator>,  

  @group(<reaction group>)  

  @timer|size(Start, Interval, Aggregator)  

  rcvMsg <EventPattern> [<Aggregate Operation>  

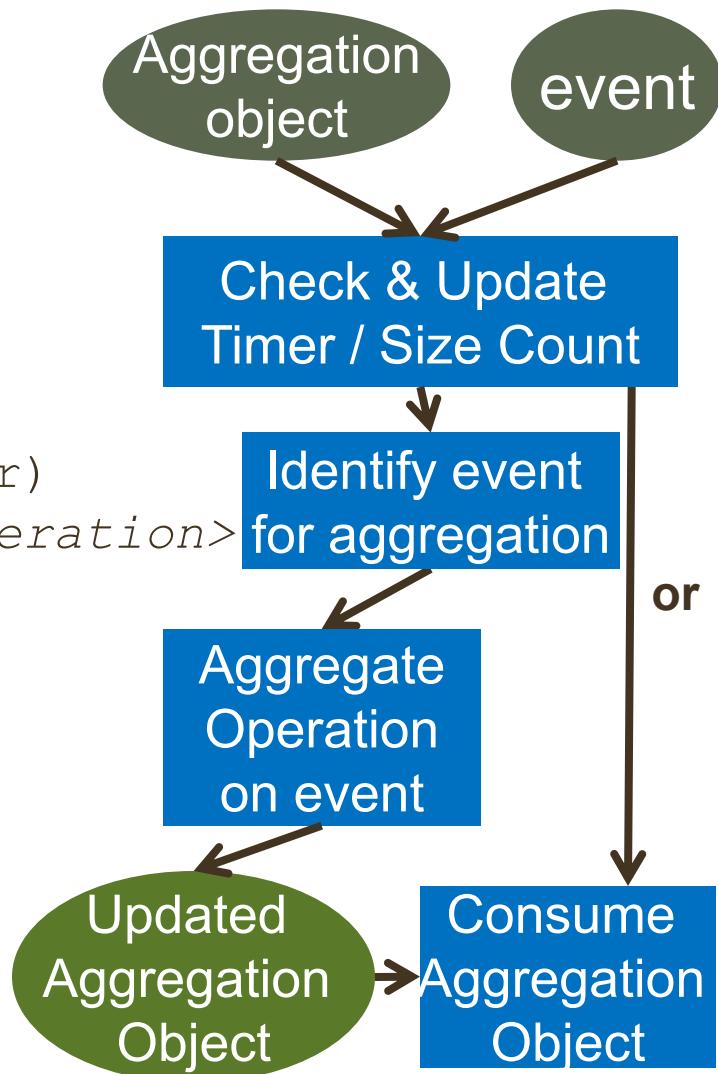
  

<rule_head> :-  

  @or(<reaction group>)  

  rcvMsg <Aggregator>,  

  ... <consume Aggregator>.
  
```



- Effect: Repeated incremental aggregations over new events

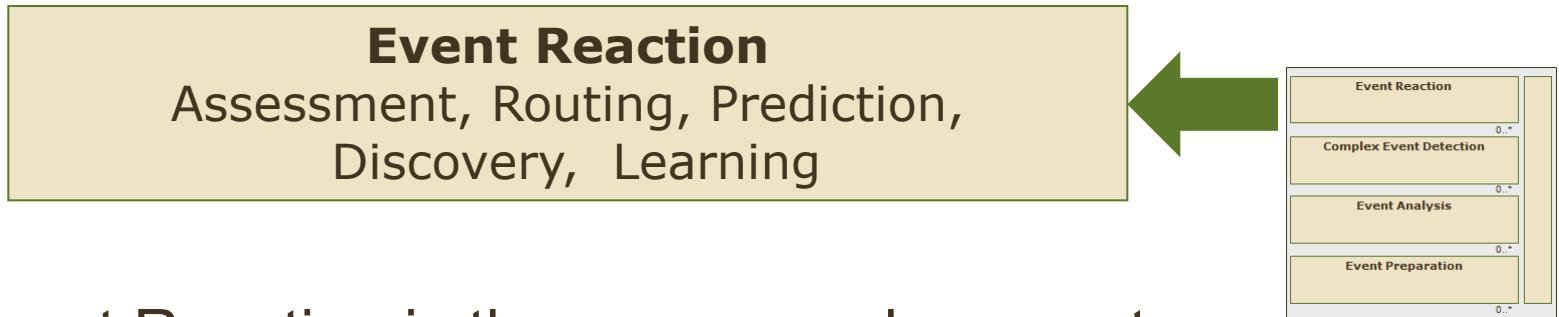
Detection:Aggregation:Implementations: Prova: Example with Time Counter

% This reaction operates indefinitely. When the timer elapses (after 25 ms), the groupby map Counter is sent as part of the aggregation event and consumed in or group, and the timer is reset back to the second argument of @timer.

```
groupby_rate() :-  
    Counter = ws.prova.eventing.MapCounter(), % Aggr. Obj.  
    @group(g1) @timer(25,25,Counter) % timer every 25 ms  
    recvMsg(XID, stream, From, inform, tick(S, P, T)) % event  
        [IM=T, Counter.incrementAt(IM)]. % aggr. operation
```

```
groupby_rate() :-  
    % receive the aggregation counter in the or reaction  
    @or(g1) recvMsg(XID, self, From, or, [Counter]),  
    ... <consume the Counter aggregation object>.
```

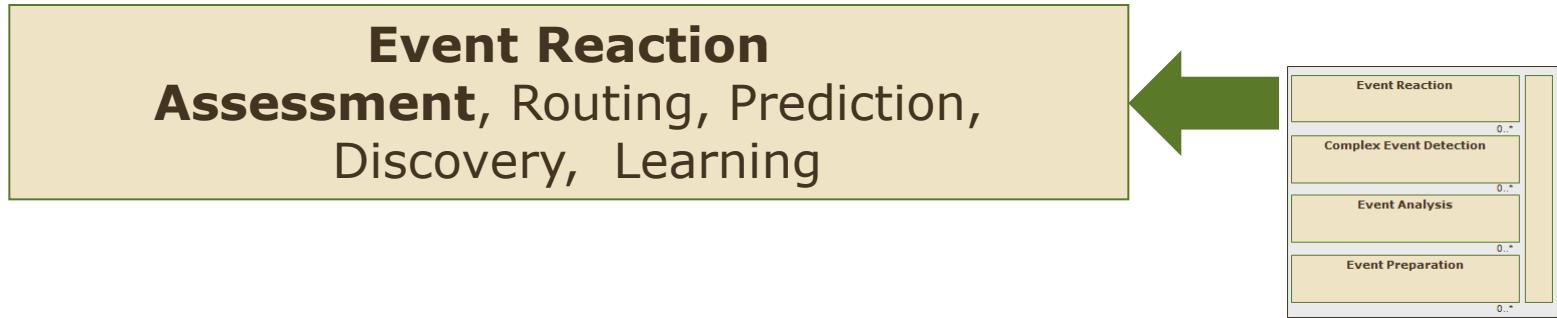
EP Patterns:RA:Reaction



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

For example, an event reaction could be the invocation of some service to process the event in some particular way.

RA:Reaction:Assessment:



Assessment: evaluate the event for inclusion in some process, collection, classification or complex event

- E.g. assess a cash withdrawal event for signs of it being a fraud event

Reaction:Assessment:Classification

- **Alternative Names:**
 - Event Classification
- **EPTS Reference Architecture:**
 - *Event assessment is the process by which an event is assessed for inclusion in some process, incorporation in some other event, etc.*

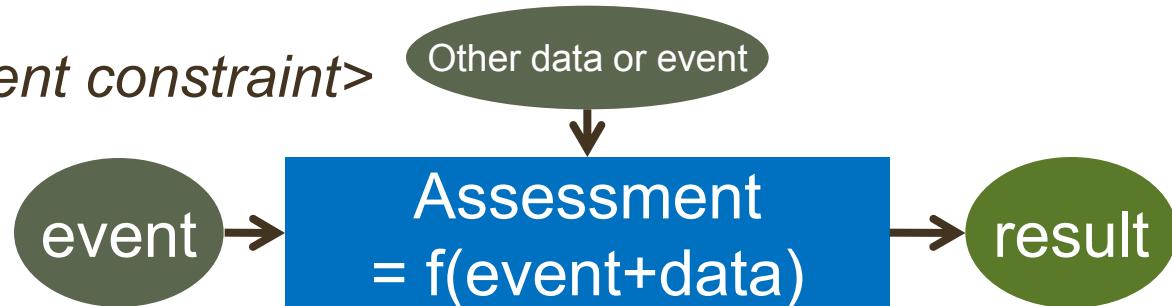
Reaction:Assessment:Description

- **Role / Explanation**

- Assessment is used to describe the process of evaluating an event for the purposes of inclusion in some process, collection or classification.
- For example: assess a parcel scan event to check whether it is classed as “in order” or “out of order” relative to some presumed parcel process

- **Associated Business Rule Specification**

- a constraint (to be enforced during event processing):
*The <event> that
satisfies <assessment constraint>
achieves a
<boolean result>*



Reaction:Assessment:Structure

- **EPTS Glossary comparison**

- Constraint (also event pattern constraint): A Boolean condition that must be satisfied by the events observed in a system.
- Event sink (event consumer) is an entity that receives events.
 - Examples:
 - Software module
 - Database
 - Dashboard
 - Person

Reaction:Assessment:Implementations: Prova: Rules

- **General pattern:**

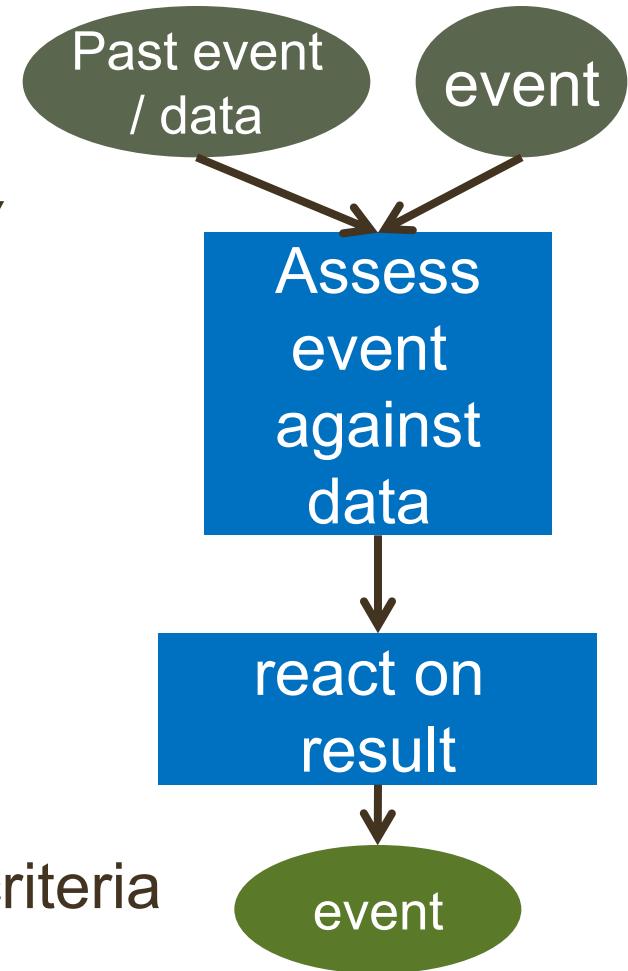
```
<rcvMsg / rcvMult> Event :-  
    assessment condition(s),  
    ....
```

```
assessment rule 1 :-  
    conditions.
```

```
assessment rule 2 :-  
    filter conditions.
```

```
....
```

- **Effect:** match event to assessment criteria and react result



Reaction:Assesment:Implementations: Prova: Example – Detect Suspicious Logins

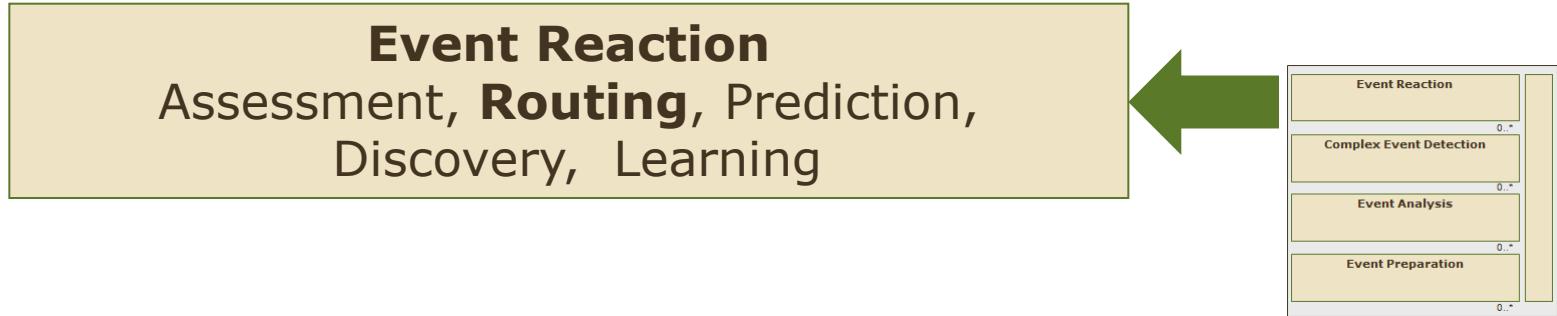
```
% detect suspicious logins by assessing the IP numbers
% of the login events from the same user login

rcvMsg(XID, Protocol, From, request, login(User, IP) ) :-  

    % if the next follow up event (@count(1)) that follows
    % the previous login is send from another IP (IP2!=IP)
    @group(g1) @count(1)
    rcvMsg(XID, Protocol, From, request, login(User, IP2) )
    [IP2!=IP],  

    println(["Suspicious login", User, IP, IP2], " ") .
```

RA:Reaction:Routing:



Routing: based on the event type and data pass the event on to the appropriate service

- E.g. customer purchase event: pass on to a provisioning service based on the type of product / product classification

Reaction:Routing:Classification

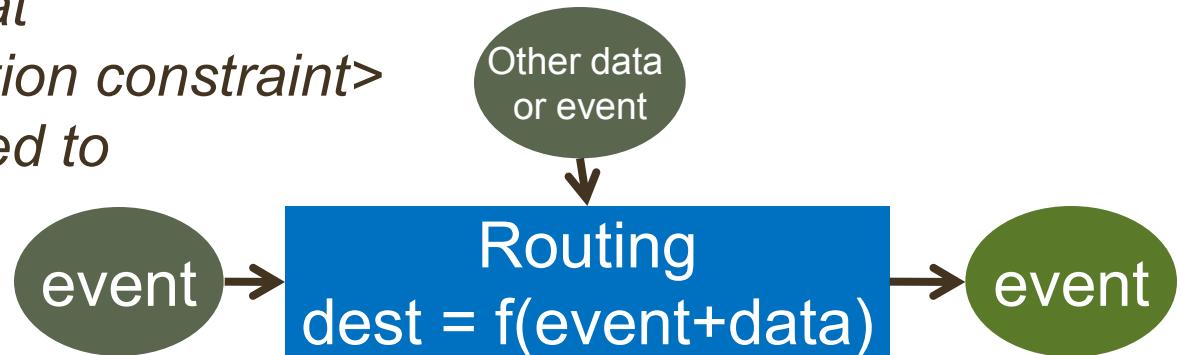
- **Alternative Names:**
 - Event Summarization, Composite event, Complex event
- **EPTS Reference Architecture:**
 - *"During event reaction, event routing is the process by which an event is redirected to some process, computation element, or other event sink. "*

Reaction:Routing:Description

- **Role / Explanation**
 - Routing is used to describe the process of adding one or more new destinations to an event.
 - For example: route an input event to the appropriate specialist agent for that event type.

- **Associated Business Rule Specification**

- a constraint (to be enforced during event processing):
*The <event> that
satisfies <selection constraint>
must be assigned to
<destination>.*



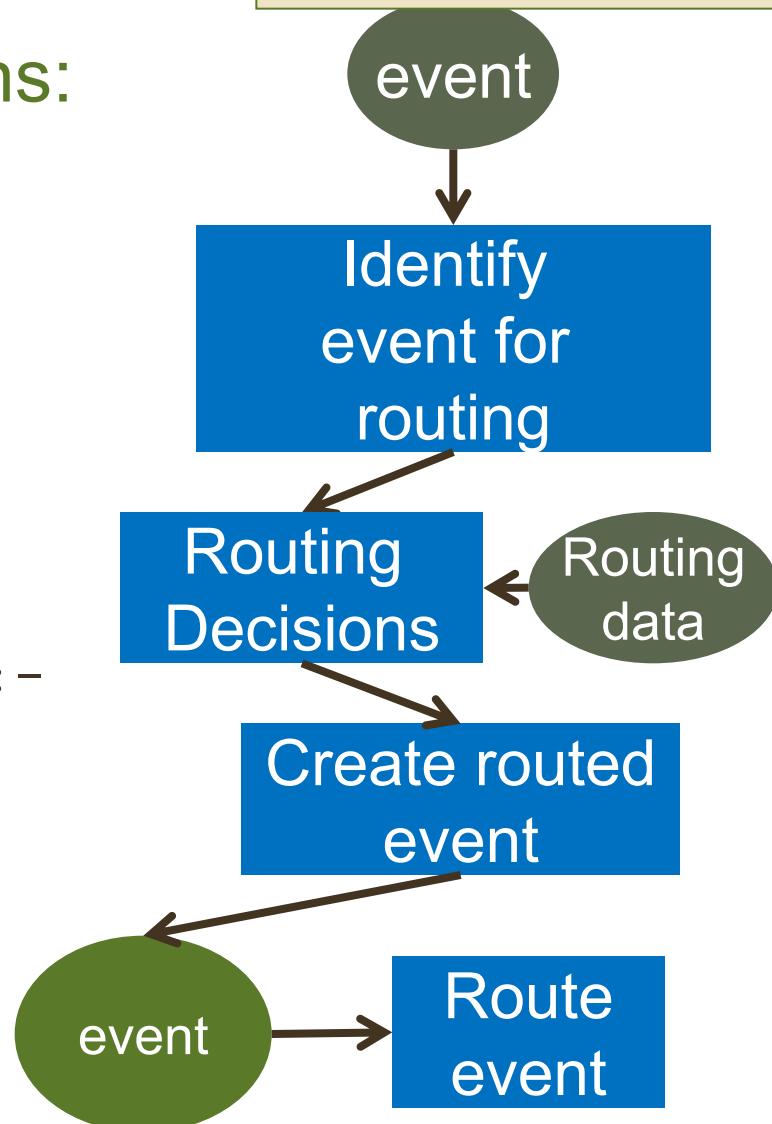
Reaction:Routing:Structure

- **EPTS Glossary comparison**
 - Routing (a process on events) is not defined, but is related to associating an event to an event sink.
 - Event sink (event consumer) is an entity that receives events.
 - Examples:
 - Software module
 - Database
 - Dashboard
 - Person

Reaction:Routing:Implementations: Prova: rules

- **General pattern template:**

```
rcvMsg <Event Msg Pattern> :-  
    <routing decisions>,  
    sendMsg <route event >.  
  
<routing decision rule 1>      :-  
    <decision conditions>.  
  
<routing decision rule 2>      :-  
    <decision conditions>.  
    ...
```



- **Effect: Events are routed according to the routing decision rules**

Reaction:Routing:Implementations:

Prova: Example with Agent (Sub-) Conversations

```

recvMsg(XID, esb, From, query-ref, buy(Product) :-  

    routeTo(Agent), % derive processing agent  

    % send order to Agent in new subconversation SID2  

    sendMsg(SID2, esb, Agent, query-ref, order(From, Product)),  

    % receive confirmation from Agent for Product order  

    recvMsg(SID2, esb, Agent, inform-ref, oder(From, Product)) .  

% route to event processing agent 1 if Product is luxury  

routeTo(epa1, Product) :- luxury(Product).  

% route to epa 2 if Product is regular  

routeTo(epa2, Product) :- regular(Product).  

% a Product is luxury if the Product has a value over ...  

luxury(Product) :- price(Product, Value), Value >= 10000.  

% a Product is regular if the Product ha a value below ...  

regular(Product) :- price(Product, Value), Value < 10000.

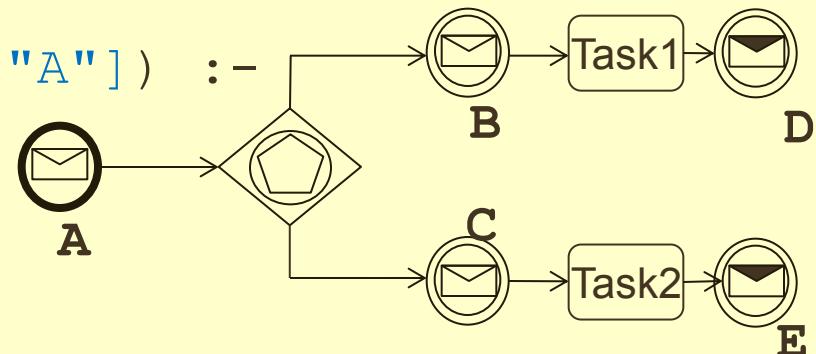
```

Reaction:Routing:Implementations:

Prova: Event Routing in Event-Driven Workflow

```
rcvMsg(XID, Process, From, event, ["A"]) :-  

    fork_b_c(XID, Process).
```



```
fork_b_c(XID, Process) :-  

    @group(p1) rcvMsg(XID, Process, From, event, ["B"] ),  

    execute(Task1), sendMsg(XID, self, 0, event, ["D"] ).
```

```
fork_b_c(XID, Process) :-  

    @group(p1) rcvMsg(XID, Process, From, event, ["C"] ),  

    execute(Task2), sendMsg(XID, self, 0, event, ["E"] ).
```

```
fork_b_c(XID, Process) :-  

    % OR reaction group "p1" waits for either of the two  

    event message handlers "B" or "C" and terminates the  

    alternative reaction if one arrives  

    @or(p1) rcvMsg(XID, Process, From, or, _ ).
```

Agenda

- Introduction to Event Processing
- Semantic Complex Event Processing (SCEP)
- Event Processing Technical Society (EPTS)
 - Event Processing Standards Reference Model
 - Event Processing Reference Architecture
- Event Processing Function Patterns - Examples
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- Reaction RuleML Standard
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- Summary

RuleML Enables ...

Rule

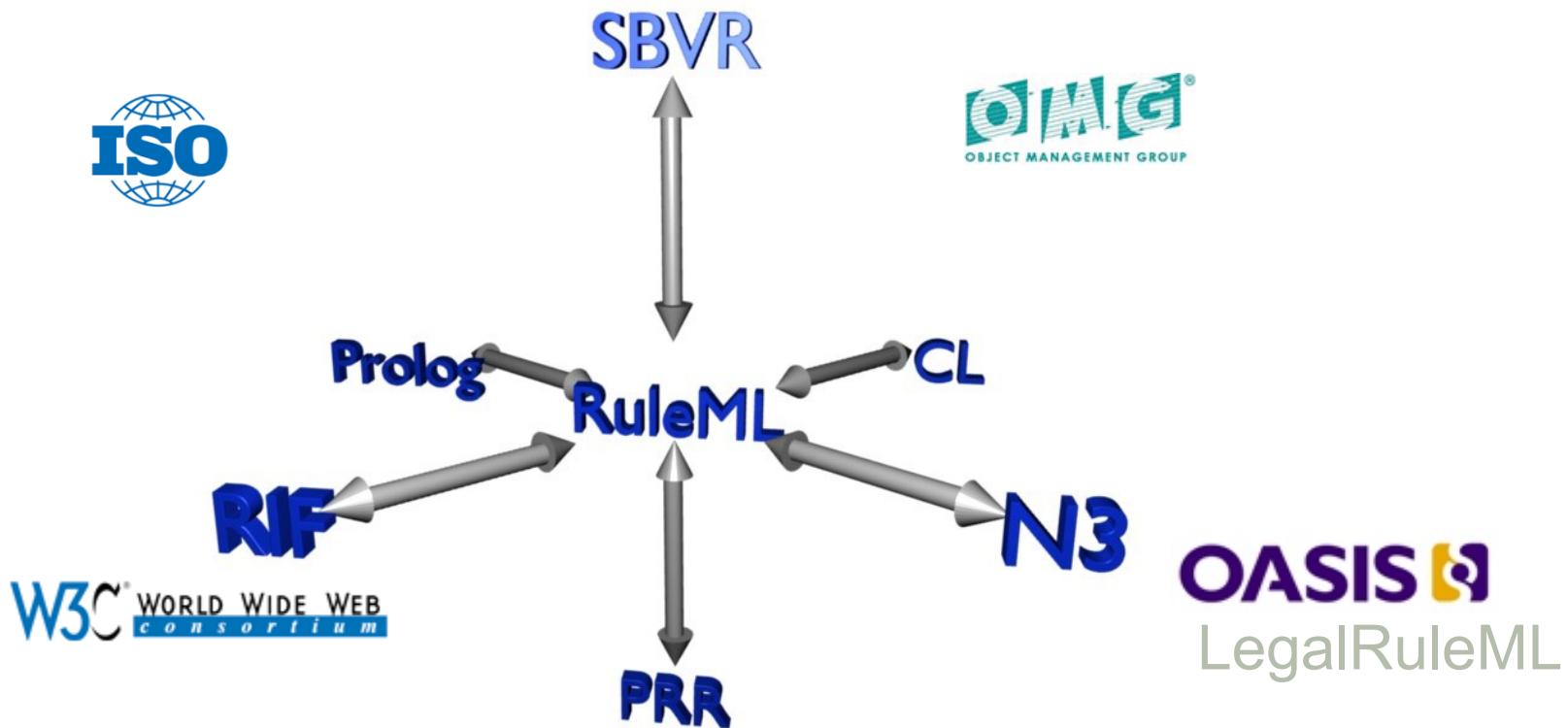
modelling
markup
translation
interchange
execution
publication
archiving

in

UML
RDF
XML
ASCII
..

RuleML Interoperation Effort

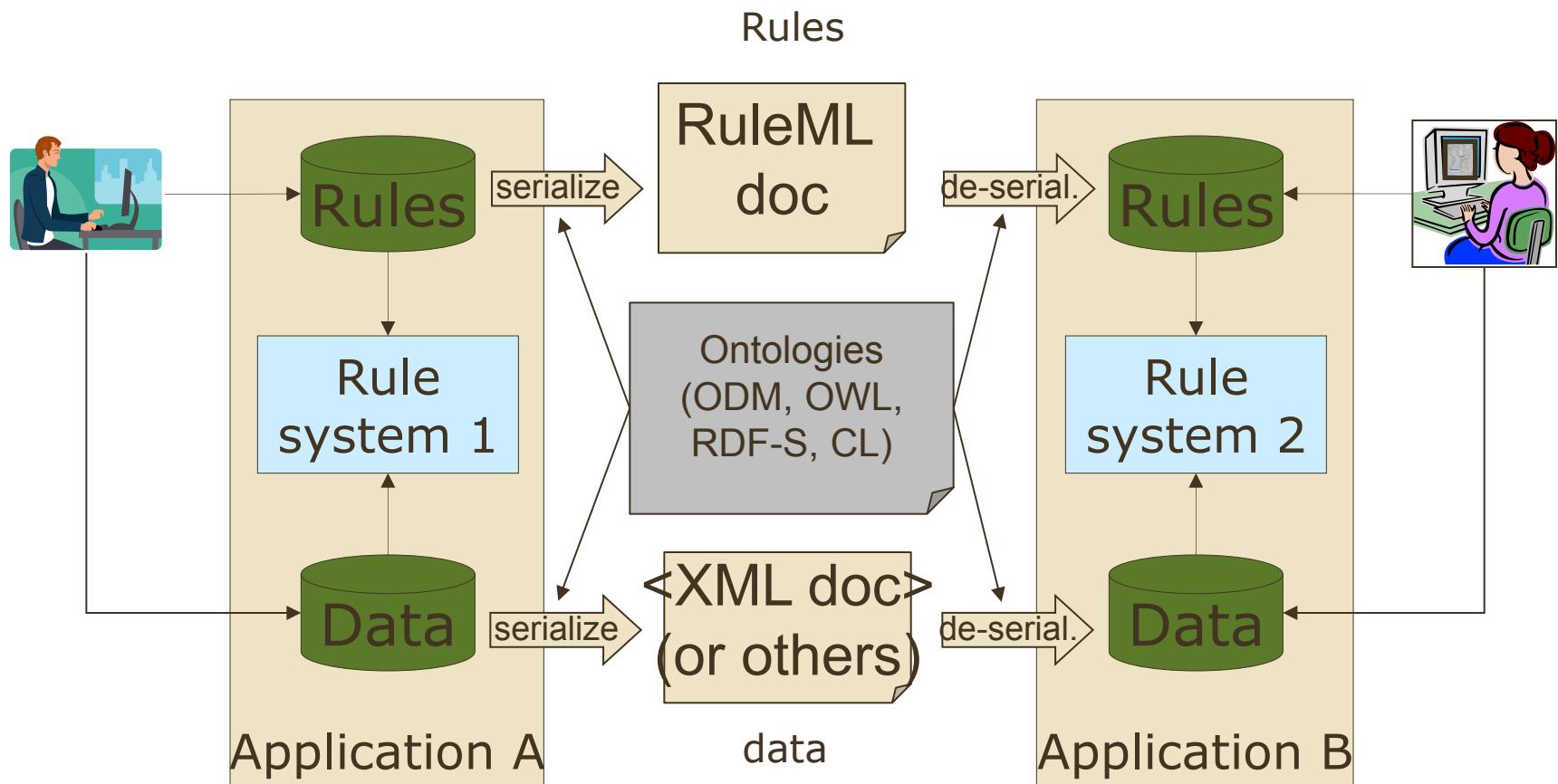
- XML-based interchange between (sublanguages of) RIF, SWRL, SWSL, SBVR, PRR, DMN, N3, Prolog, Rulelog, CL etc.



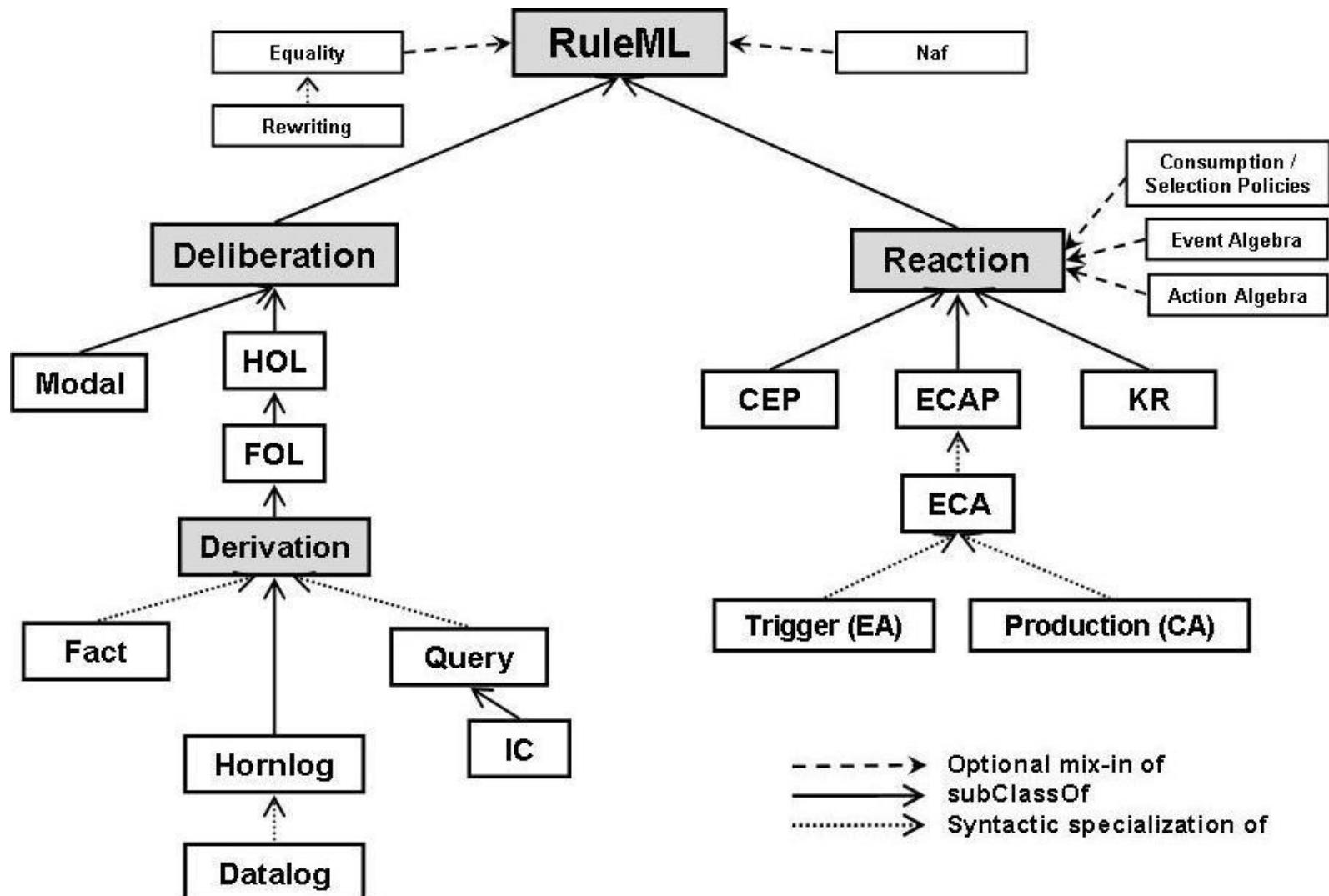
RuleML Basis for other Languages - Examples

- **Semantic Web Rule Language (SWRL)**
 - Uses RuleML Version 0.89
- **Semantic Web Services Language (SWSL)**
 - Uses RuleML Version 0.89
- **W3C Rule Interchange Format**
 - Uses RuleML Version 0.91 with frames and slots
- **OASIS LegalRuleML**
 - Uses RuleML Version 1.0
- **OMG PRR**
 - Input from RuleML
- **OMG API4KB**
 - Input from Reaction RuleML 1.0
- ...

RuleML Rule interchange (on the PIM level)

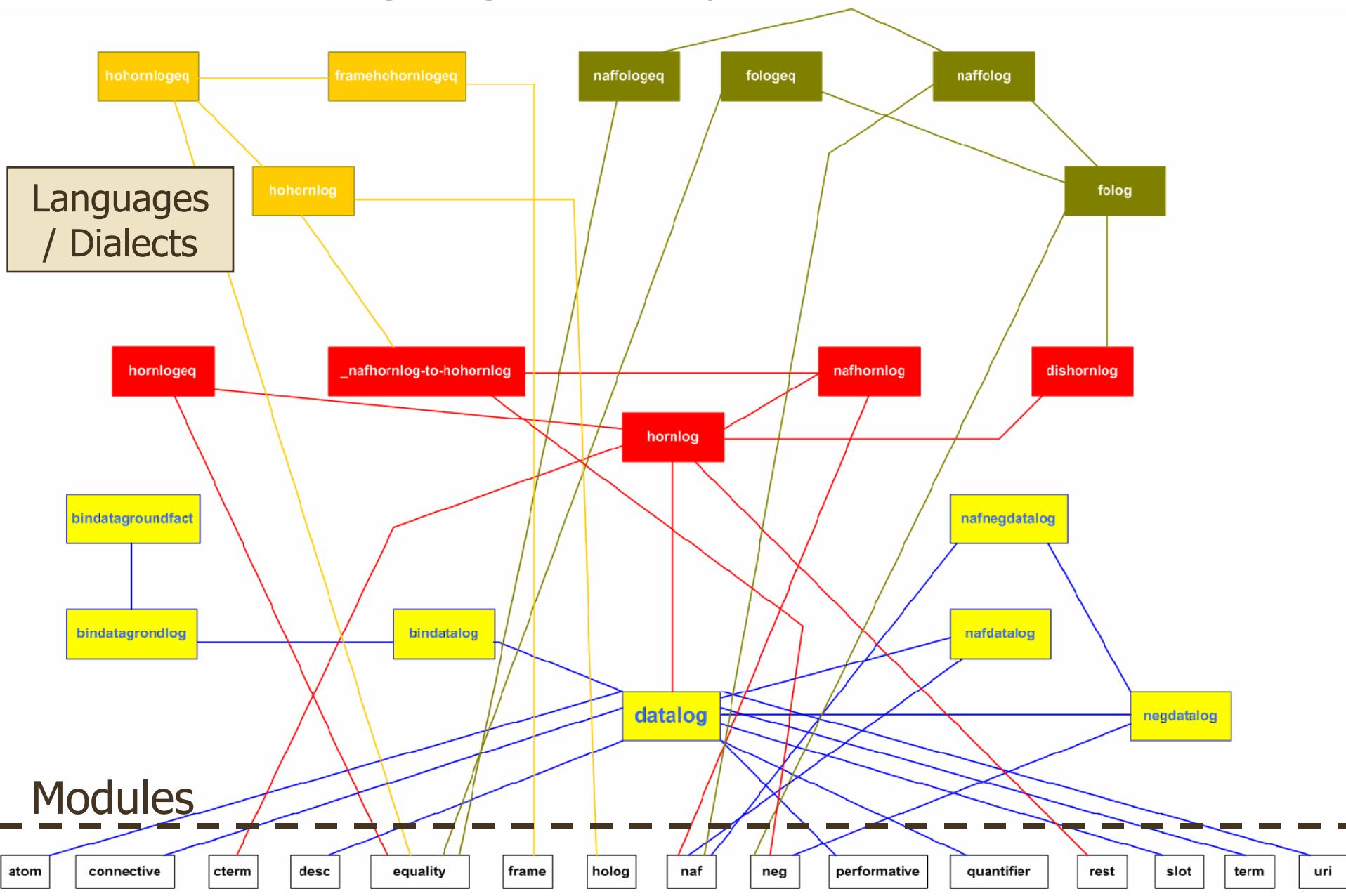


RuleML Family of Sublanguages



Harold Boley, Adrian Paschke and Omair Shafiq: **RuleML 1.0: The Overarching Specification of Web Rules**, in Semantic Web Rules, Proceedings of RuleML-2010, Lecture Notes in Computer Science, 2010, Volume 6403/2010, 162-178.

RuleML Language Family – Derivation RuleML



RuleML Sublanguages Customized by MYNG as Relax NG Schemas (1)



Selection Form

Instructions

Make a selection from the form below, then click "Refresh Schema" to update the Schema URL. The main module is also displayed below the form. To reset the form to the default (supremum) values, click "Reset Form".

[Reset Form](#) [Refresh Schema](#)

Schema URL = http://ruleml.org/1.0/relaxng/schema_rnc.php?backbone=x3f&default=x7&termseq=x7&lng=x1&propo=x3ff&implies=x7&terms=xf3f&quant=x7&expr=xf&serial=xf

Expressivity	Treatment of Attributes	Term Sequences:	Language	Serialization
"Backbone"	With Default Values	Number of Terms	(Check One)	Options
(Check One)	(Check One)	(Check One)		(Check Zero or More)
<input type="radio"/> Atomic Formulas	<input type="radio"/> Required to be	<input type="radio"/> None	<input checked="" type="radio"/> English Abbreviated Names	<input checked="" type="checkbox"/> Unordered Groups
<input type="radio"/> Ground Fact	Absent	<input type="radio"/> Binary (Zero or Two)	<input type="radio"/> English Long Names	<input checked="" type="checkbox"/> Stripe-Skipping
<input type="radio"/> Ground Logic	<input type="radio"/> Required to be	<input checked="" type="radio"/> Polyadic (Zero or	<input type="radio"/> French Long Names	<input checked="" type="checkbox"/> Explicit Datatyping
<input type="radio"/> Datalog	Present	More)		<input checked="" type="checkbox"/> Schema Location Attribute
<input type="radio"/> Horn Logic	<input checked="" type="radio"/> Optional			
<input type="radio"/> Disjunctive Logic				
<input checked="" type="radio"/> Full First-Order Logic				

<input type="radio"/> Required to be	<input type="radio"/> None	<input checked="" type="radio"/> English Abbreviated Names
Absent	<input type="radio"/> Binary (Zero or Two)	<input type="radio"/> English Long Names
<input type="radio"/> Required to be	<input checked="" type="radio"/> Polyadic (Zero or	<input type="radio"/> French Long Names
Present	More)	
<input checked="" type="radio"/> Optional		

<input checked="" type="checkbox"/> Stripe-Skipping
<input checked="" type="checkbox"/> Explicit Datatyping
<input checked="" type="checkbox"/> Schema Location Attribute

RuleML Sublanguages Customized by MYNG as Relax NG Schemas (2)

Propositional Options
(Check Zero or More)
 IRIs
 Rulebases
 Entailments
 Degree of Uncertainty
 Strong Negation
 Weak Negation (Negation as Failure)
 Node Identifiers
 In-Place Annotation
 XML base
 XML id

Implication Options
(Check Zero or More)
 Equivalences
 Inference Direction
 Non-Material

Term Options
(Check Zero or More)
 Object Identifiers
 Slots
 Slot Cardinality
 Slot Weight
 Equations
 Oriented Equations
 Term Typing
 Data Terms
 Skolem Constants
 Reified Terms

Quantification Options
(Check Zero or More)
 Implicit Closure
 Slotted Rest Variables
 Positional Rest Variables

Expression Options
(Check Zero or More)
 Generalized Lists
 Set-valued Expressions
 Interpreted Expressions

RuleML Core Technology

- **XML schemas**
 - In Relax NG
 - Based on MYNG customization approach and Web GUI
 - In XML Schema Definition Language (XSD)
- **Presentation & Visualization syntaxes**
 - Positional-Slotted Language (POSL) and Prolog (Prova)
 - Grailog
- **Transformations**
 - XSLT normalizer, ANTLR parsers, JAXB unmarshalling of RuleML/XML into Java objects
- **Translators (interchange/interoperation tools)**
 - Prolog, Jess, N3, ...
- **Model-theoretic semantics**
 - For (Naf-free, OID/slot-free) FOL, Hornlog, Datalog RuleML: Classical
 - Positional-Slotted Object-Applicative (PSOA) RuleML

RuleML Extended Technology

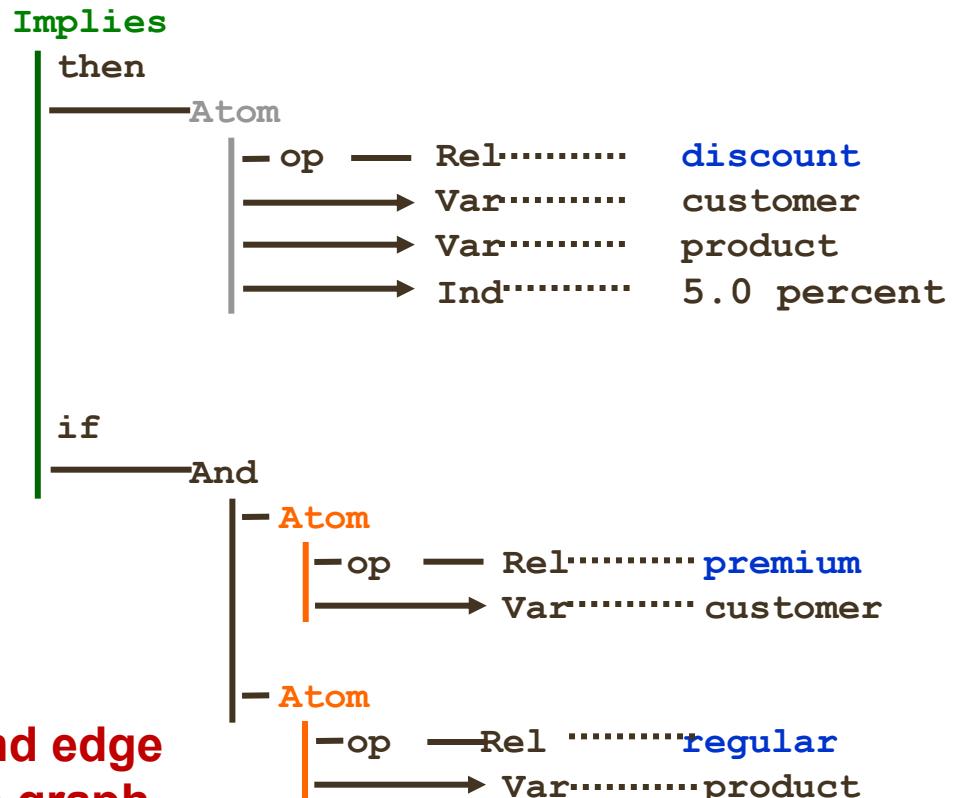
- **Translators (interchange/interoperation tools), e.g.**
 - Translators between RuleML and Prolog ([Prova](#)), Jess, N3, ...
 - RuleML ↔ POSL
 - PSOA → TPTP
 - Attempto Controlled English (ACE) → RuleML
- **IDEs (editors, generators)**
- **Engines (e.g. OO jDREW, Prova, DR-DEVICE)**
- **APIs (e.g. [Rulestore API](#), [API4KB](#))**
- **Multi-agent frameworks (e.g. [Rule Responder](#), [EMERALD](#))**
- **Other tools**
[\(http://wiki.ruleml.org/index.php/RuleML Implementations\)](http://wiki.ruleml.org/index.php/RuleML_Implementations)

RuleML Language Design – Striped Syntax

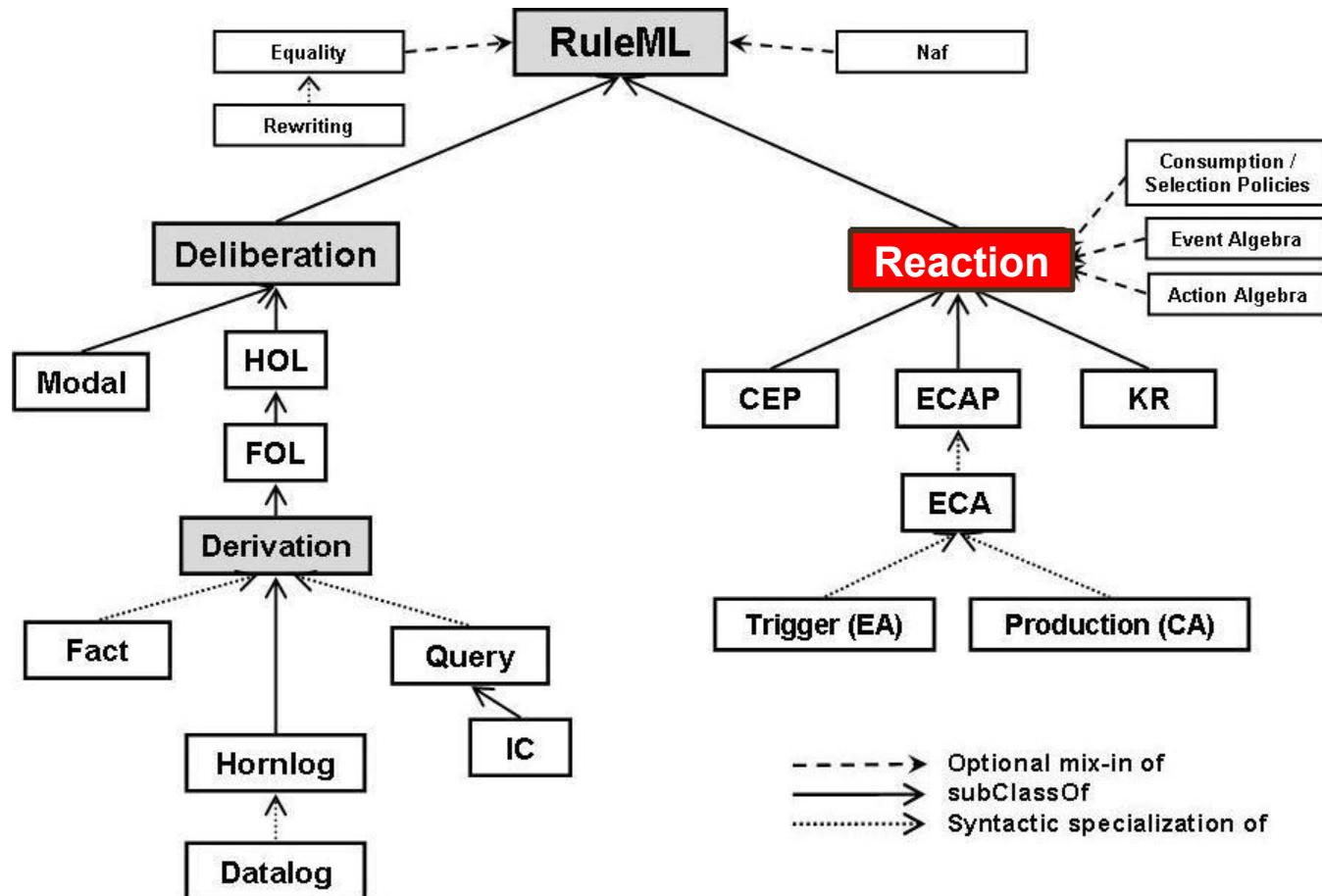
"The **discount** for a *customer* buying a *product* is **5.0 percent** if the *customer* is **premium** and the *product* is **regular**."

```
<Implies>--> node tag  
<then>--> edge tag  
  <Atom>  
    <op><Rel>discount</Rel></op>  
    <Var>customer</Var>  
    <Var>product</Var>  
    <Ind>5.0 percent</Ind>  
  </Atom>  
</then>  
<if>  
  <And>  
    <Atom>  
      <op><Rel>premium</Rel></op>  
      <Var>customer</Var>  
    </Atom>  
    <Atom>  
      <op><Rel>regular</Rel></op>  
      <Var>product</Var>  
    </Atom>  
  </And>  
</if>  
</Implies>
```

Iteration of node and edge tags maps to (RDF) graph structure



The Reaction RuleML Family



Reaction Rules: Four Sub-branches

- PR Production Rules
(Condition-Action rules)
- ECA Event-Condition-Action (ECA) rules
- CEP Rule-based Complex Event Processing
(complex event processing reaction rules,
(distributed) event messaging reaction rules,
query reaction rules, etc.)
- DR and KR Knowledge Representation AI Reasoning with Temporal/Event/Action/
Situation/ Transition/Process Logics and
Calculi

Reaction Rules: Specializable Syntax

<Rule @key @keyref @style>		
Info, Life Cycle Mgt.	<meta> <!-- descriptive metadata of the rule -->	</meta>
Interface	<scope> <!-- scope of the rule e.g. a rule module -->	</scope>
	<evaluation> <!-- intended semantics -->	</evaluation>
	<signature> <!-- rule signature -->	</signature>
	<qualification> <!-- e.g. qualifying rule metadata, e.g. priorities, validity, strategy -->	</qualification>
	<quantification> <!-- quantifying rule declarations, e.g. variable bindings -->	</quantification>
Imple- mentation	<oid> <!-- object identifier -->	</oid>
	<on> <!-- event part -->	</on>
	<if> <!-- condition part -->	</if>
	<then> <!-- (logical) conclusion part -->	</then>
	<do> <!-- action part -->	</do>
	<after> <!-- postcondition part after action, e.g. to check effects of execution -->	</after>
	<else> <!-- (logical) else conclusion -->	</else>
	<elsedo> <!-- alternative/else action, e.g. for default handling -->	</elsedo>
	</Rule>	

Reaction RuleML – Example Rule Types

- **Derivation Rule:**
(temporal/event/action/situation reasoning)

```
<Rule style="reasoning">
    <if>...</if>
    <then>...</then>
</Rule>
```
- **Production Rule:**

```
<Rule style="active">
    <if>...</if>
    <do>...</do>
</Rule>
```
- **Trigger Rule:**

```
<Rule style="active"> >
    <on>...</on>
    <do>...</do>
</Rule>
```
- **ECA Rule:**

```
<Rule style="active">
    <on>...</on>
    <if>...</if>
    <do>...</do>
</Rule>
```

Rule Interface and Rule Implementation

```
<Rule style="active">
  <evaluation>
    <Profile type="&rule!;DefiniteProductionRule" direction="forward" style="active"/>
  </evaluation>
  <signature>
    <Atom>
      <op><Rel>likes</Rel></op>
      <arg><Var mode="+" /></arg> <!-- mode=+ i.e. input argument -->
      <arg><Var mode="?" /></arg> <!-- mode=- i.e. input or output argument -->
    </Atom>
  </signature>
```

Interface with evaluation semantics and rule signature declaration

```
<if>
  <Atom>
    <op><Rel>likes</Rel></op>
    <arg><Var>X</Var></arg>
    <arg><Ind>wine</Ind></arg>
  </Atom>
</if>
<do>
  <Assert> <formula>
    <Atom>
      <op><Rel>likes</Rel></op>
      <arg><Ind>John</Ind></arg>
      <arg><Var>X</Var></arg>
    </Atom>
  </formula> </Assert> </do>
</Rule>
```

Implementation

Separation of Interface and Implementation

(Distributed Knowledge)

```
<Rule key="ruleinterface1">  
  <evaluation><Profile> ...p1... </Profile><evaluation>  
  <evaluation><Profile> ...p2 ...</Profile></evaluation>  
  <signature>...s1...</signature>  
</Rule>
```

```
<Rule key="ruleinterface2">  
  <evaluation><Profile> ...p3... </Profile><evaluation>  
  <signature>...s2...</signature>  
</Rule>
```

```
...  
  
<Rule keyref="ruleinterface2" key="ruleimpl2 ">  
  <if> ... </if>  
  <do> </do>  
</Rule>
```

```
<Rule keyref="ruleinterface1" key="ruleimpl1 ">  
  <if> ... </if>  
  <do> </do>  
</Rule>
```

Interface 1
(@key=ruleinterface1)

Interface 2
(@key=ruleinterface2)

Implementation 2
(@keyref=ruleinterface2)

Implementation 1
(@keyref=ruleinterface1)

Example: Distributed Rule Interface and Implementation

<-- rule interface with two alternative interpretation semantics and a signature.
The interface references the implementation identified by the corresponding key -->

```
<Rule keyref="#r1">  
  <evaluation index="1">  
    <!-- WFS semantic profile define in the metamodel -->  
    <Profile type="&ruleml;Well-Founded-Semantics" direction="backward"/>  
  </evaluation>
```

```
  <evaluation index="2">  
    <!-- alternative ASS semantic profile define in the metamodel -->  
    <Profile type="&ruleml;Answer-Set-Semantics" direction="backward"/>  
  </evaluation>
```

<!-- the signature defines the queryable head of the backward-reasoning rule -->

```
<signature>  
  <Atom><Rel>getPapers</Rel><Var mode="+" /><Var mode="+" /><Var mode="-" /></Atom>  
</signature>
```

```
</Rule>
```

<!-- implementation of rule 1 which is interpreted either by WFS or by ASS semantics
and only allows queries according to it's signature definition. -->

```
<Rule key="#r1" style="reasoning">  
  <if> ... </if>  
  <then>  
    <Atom><Rel>getPapers</Rel><Var>Type</Var><Var>Status</Var><Var>Papers</Var></Atom>  
</then>
```

Interface
with
evaluation
semantics
and
public rule
signature

**Implemen
tation**

Descriptive and Qualifying Metadata

```
<Assert key="#module1">
  <meta> <!-- descriptive metadata -->
    <Atom><Rel iri="dc:creator"/><Ind>Adrian Paschke</Ind></arg></Atom>
  </meta>
  <meta>
    <Time type="&ruleml;TimelInstant"><Data xsi:type="xs:date">2011-01-01</Data></Time>
  </meta>
  <meta>
    <Atom><Rel>source</Rel><Ind>./module1.prova</Ind></Atom>
  </meta>
  <qualification> <!-- qualifying metadata -->
    <Atom> <!-- the module is valid for one year from 2011 to 2012 -->
      <Rel>valid</Rel>
      <Interval type="&ruleml;TimelInstant">
        <Time type="&ruleml;TimelInstant"><Data xsi:type="xs:date">2011-01-01</Data></Time>
        <Time type="&ruleml;TimelInstant"><Data xsi:type="xs:date">2012-01-01</Data></Time>
      </Interval>
    </Atom>
  </qualification>
<Rulebase key="#innermodule1.1">
  <meta>...</meta> <qualification> ... </qualification>
  <Rule key="#rule1">
    <meta>...<meta>       <qualification> ... </qualification>
  </Rule>
  <Rule key="#rule2">
    <meta>...<meta>       <qualification> ... </qualification>
  </Rule>
</Rulebase>
</Assert>
```

} Descriptive Metadata

} Qualifying Metadata

Dynamic Views with Scopes and Guards

```
<Assert key="#module1">
  <meta> <!-- descriptive metadata -->
    <Atom><Rel iri="dc:creator"/><Ind>Adrian Paschke</Ind></arg></Atom>
  </meta>
  <qualification> <!-- qualifying metadata -->
    <Atom> <!-- the module is valid for one year from 2011 to 2012 -->
      <Rel>valid</Rel>
      <Interval type="&ruleml;TimeInstant">
        <Time type="&ruleml;TimeInstant"><Data xsi:type="xs:date">2011-01-01</Data></Time>
        <Time type="&ruleml;TimeInstant"><Data xsi:type="xs:date">2012-01-01</Data></Time>
      </Interval>
    </Atom>
  </qualification>
<Rulebase key="#innermodule1.1">
  <!-- the rule base scopes apply to all knowledge in the rule base -->
  <scope> <!-- scope : only knowledge authored by „Adrian Paschke“ -->
    <Atom><Rel iri="dc:creator"/><Ind>Adrian Paschke</Ind></arg></Atom>
  </scope>
  <scope> <!-- scope : only valid knowledge; validity value will be bound to variable -->
    <Atom><Rel>valid</Rel><Var>Validity</Var></Atom>
  </scope>
  <Rule key="#rule1">
    <scope> <Atom><Rel>source</Rel><Ind>./module1.prova</Ind></Atom></scope>
    <guard> <!-- guard on the validity: “current date during validity of module1”>
      <Operator type="&ruleml;During"><Expr iri="...getDate()"/><Var>Validity</Var></Operator>
    </guard>
    <if> ... </if> <then> </then>
  </Rule>
</Rulebase>
</Assert>
```

Outer Scope

Inner Scope + Guard

Quick Overview: Reaction RuleML Dialects

* + variants and alternatives

- **Derivation RuleML (*if-then*)^{*}**

- Time, Spatial, Interval, Situation (+ algebra operators)

- **KR RuleML (*if-then or on-if-do*)^{*}**

- Happens_(@type), Initiates, Terminates, Holds, fluent

- **Production RuleML (*if-do*)^{*}**

- Assert, Retract, Update, Action

- **ECA RuleML (*on-if-do*)^{*}**

- Event, Action, + (event / action algebra operators)

- **CEP** (arbitrary combination of on, if, do)

- Receive, Send, Message

Selected Reaction RuleML Algebra Operators

- **Action Algebra**
Succession (Ordered Succession of Actions), *Choice* (Non-Deterministic Choice), *Flow* (Parallel Flow), *Loop* (Loops), *Operator* (generic Operator)
- **Event Algebra**
Sequence (Ordered), *Disjunction* (Or) , *Xor* (Mutual Exclusive), *Conjunction* (And), *Concurrent* , *Not*, *Any*, *Aperiodic*, *Periodic*, *AtLeast*, *ATMost*, *Operator* (generic Operator)
- **Interval Algebra (Time/Spatio/Event/Action/... Intervals)**
During, *Overlaps*, *Starts*, *Precedes*, *Meets*, *Equals*, *Finishes*, *Operator* (generic Operator)
- **Counting Algebra**
Counter, *AtLeast*, *AtMost*, *Nth*, *Operator* (generic Operator)
- **Temporal operators**
Timer, *Every*, *After*, *Any*, *Operator* (generic Operator)
- **Negation operators**

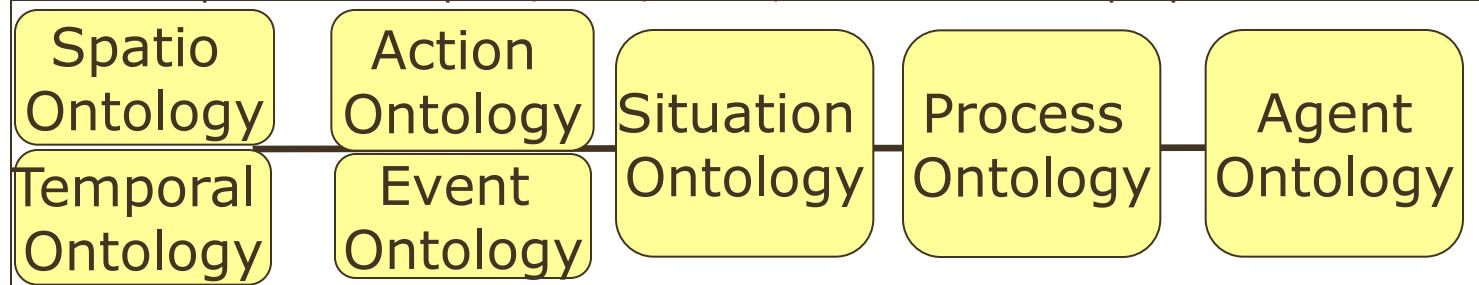
Reaction RuleML Metamodel and External Ontologies + Data - Examples

```
<Quantifier type="&ruleml;Forall"> == <Forall>
<Operator type="&ruleml;And"> == <And>
<Operator type="&ruleml;Conjunction"> == <Conjunction>
<Negation type="&ruleml;InflationaryNegation"> == <Naf>
<Action type="&ruleml;Assert"> == <Assert>
<Action type="&ruleml;Retract"> == <Retract>
<Event type="&ruleml;SimpleEvent"> == <Atom> ... </Atom>
<Event type="ibm:CommonBaseEvent"> == IBM CBE
<Operator type="snoop:Squence"> == Snoop Algebra
  == <Operator type="&ruleml;Sequence"> == <Sequence>
<Ind iri="person.xml#xpointer{//Person/LastName[1]/text()}" />
<Action iri="BPEL.xml#xpointer{//invoke[@name=checkHotel]}"/>
```

Reaction RuleML Metamodel

Top Level Ontologies

General concepts such as space, time, event, action and their properties and relations



Vocabularies **related to specific domains** by specializing the concepts introduced in the top-level ontology

Domain Ontologies

Task Activities Ontologies

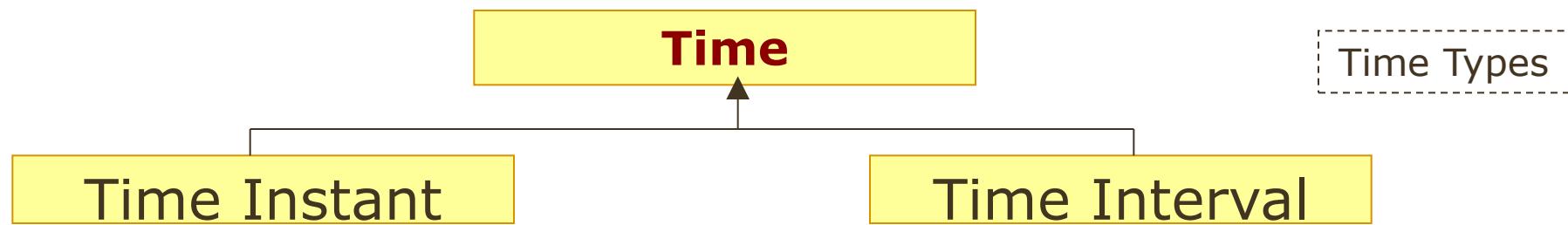
Specific
user/application
ontologies

Application Ontologies

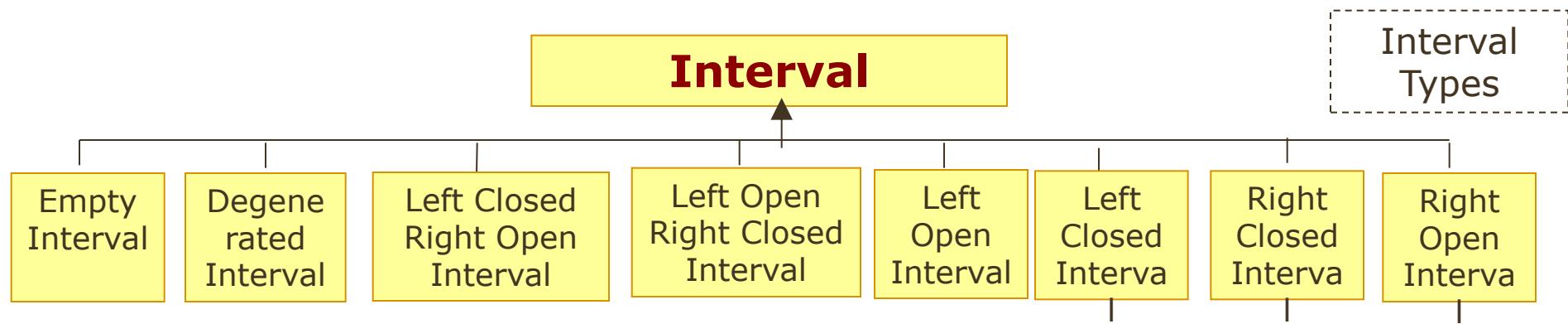
Vocabularies **related to generic tasks or activities** by specializing the concepts introduced in the top-level ontology

E.g. ontologies describing roles played by domain entities while performing application activities

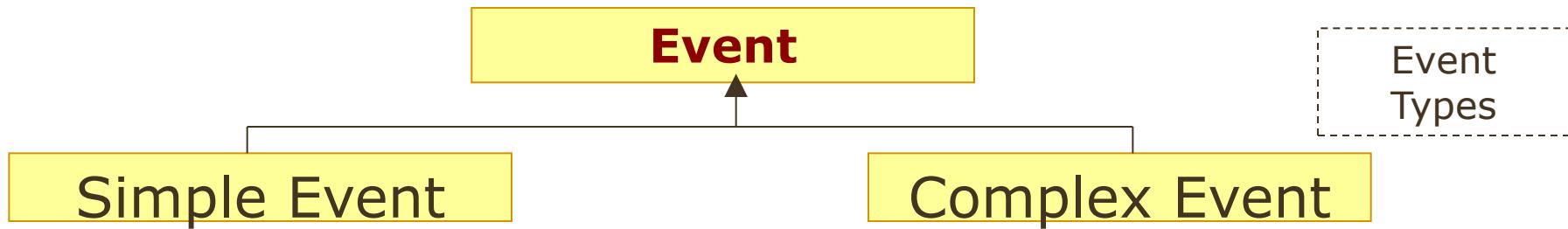
Time Top Ontology Metamodel



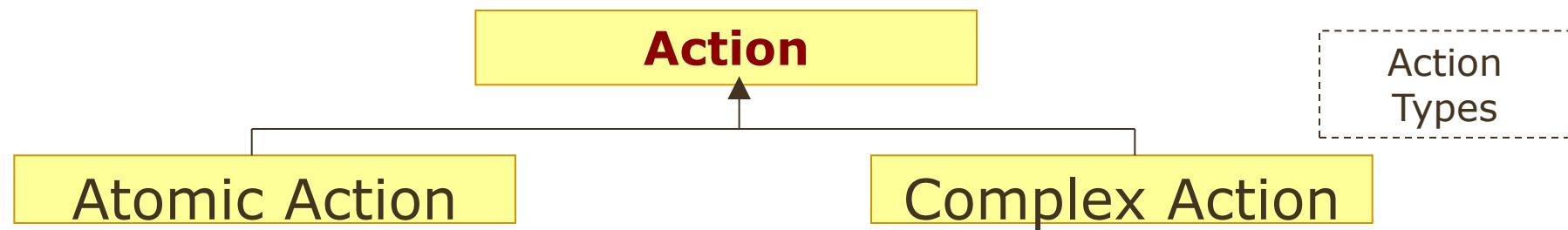
Interval Top Ontology Metamodel



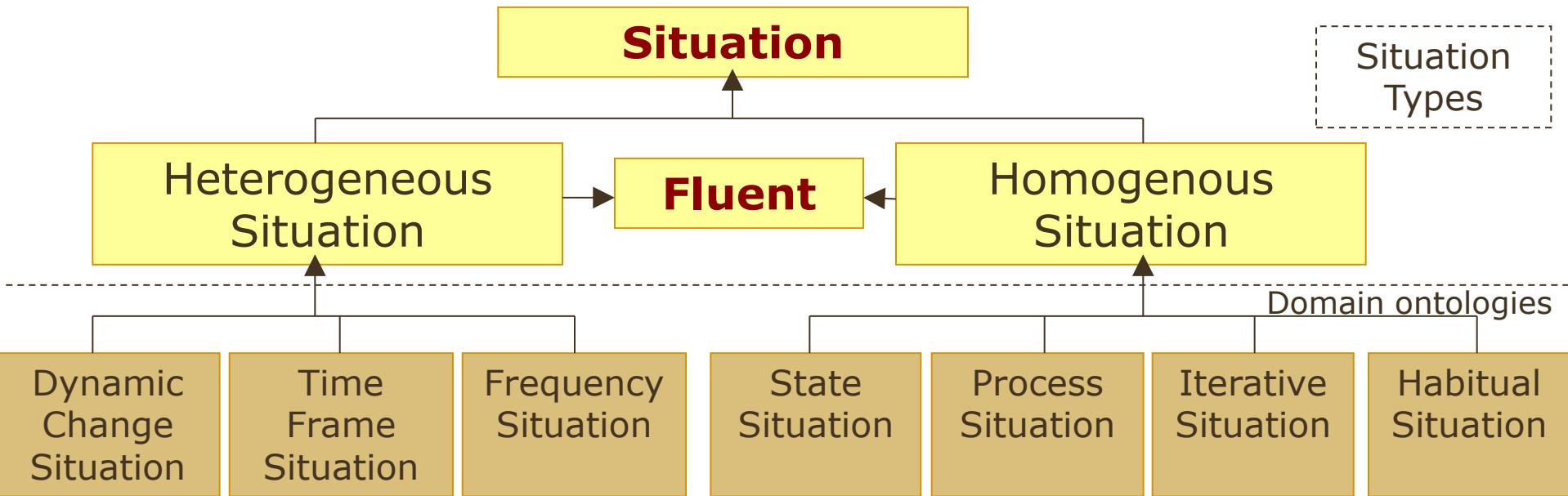
Event Top Ontology Metamodel



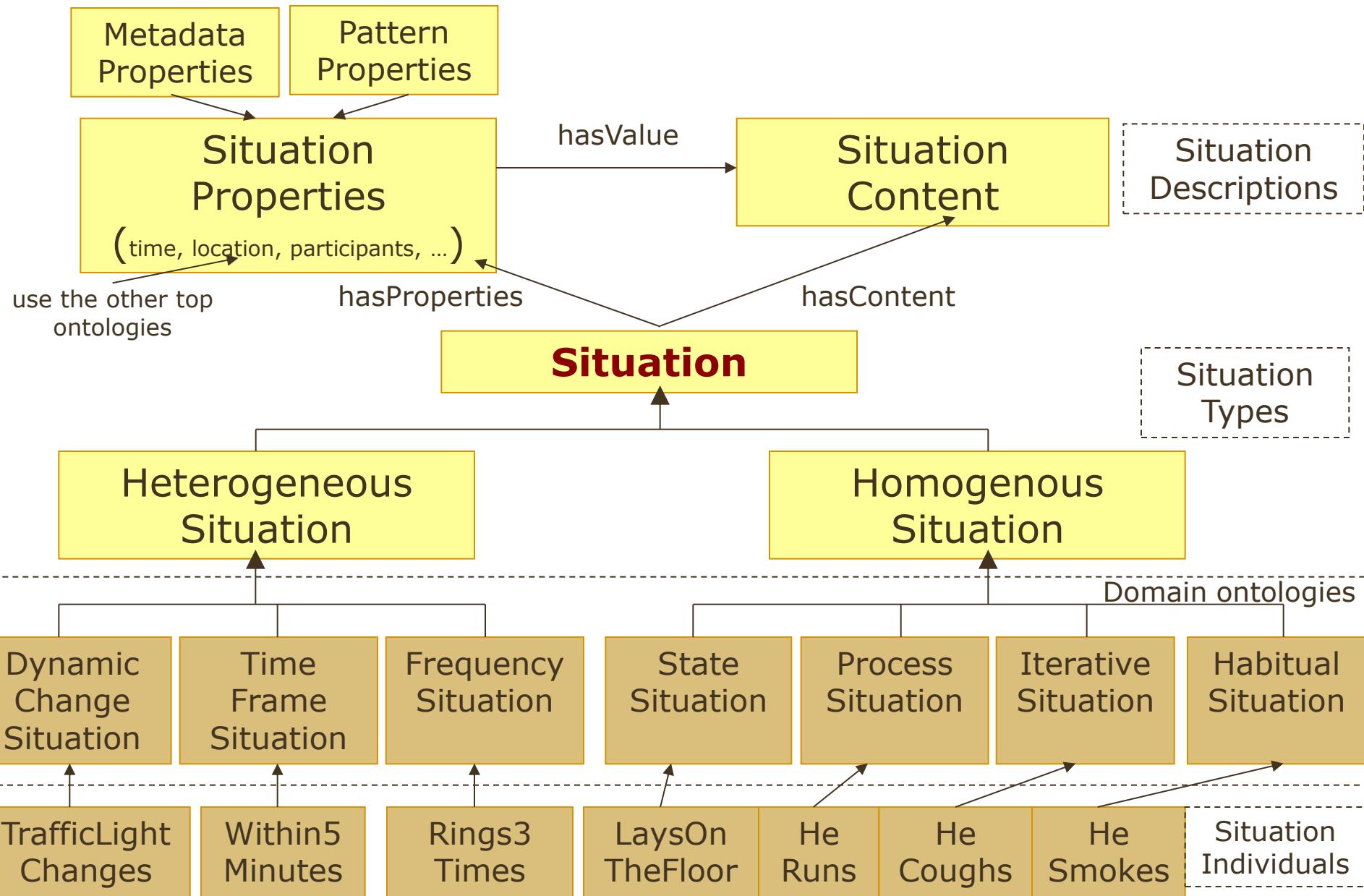
Action Top Ontology Metamodel



Situation Top Ontology Metamodel

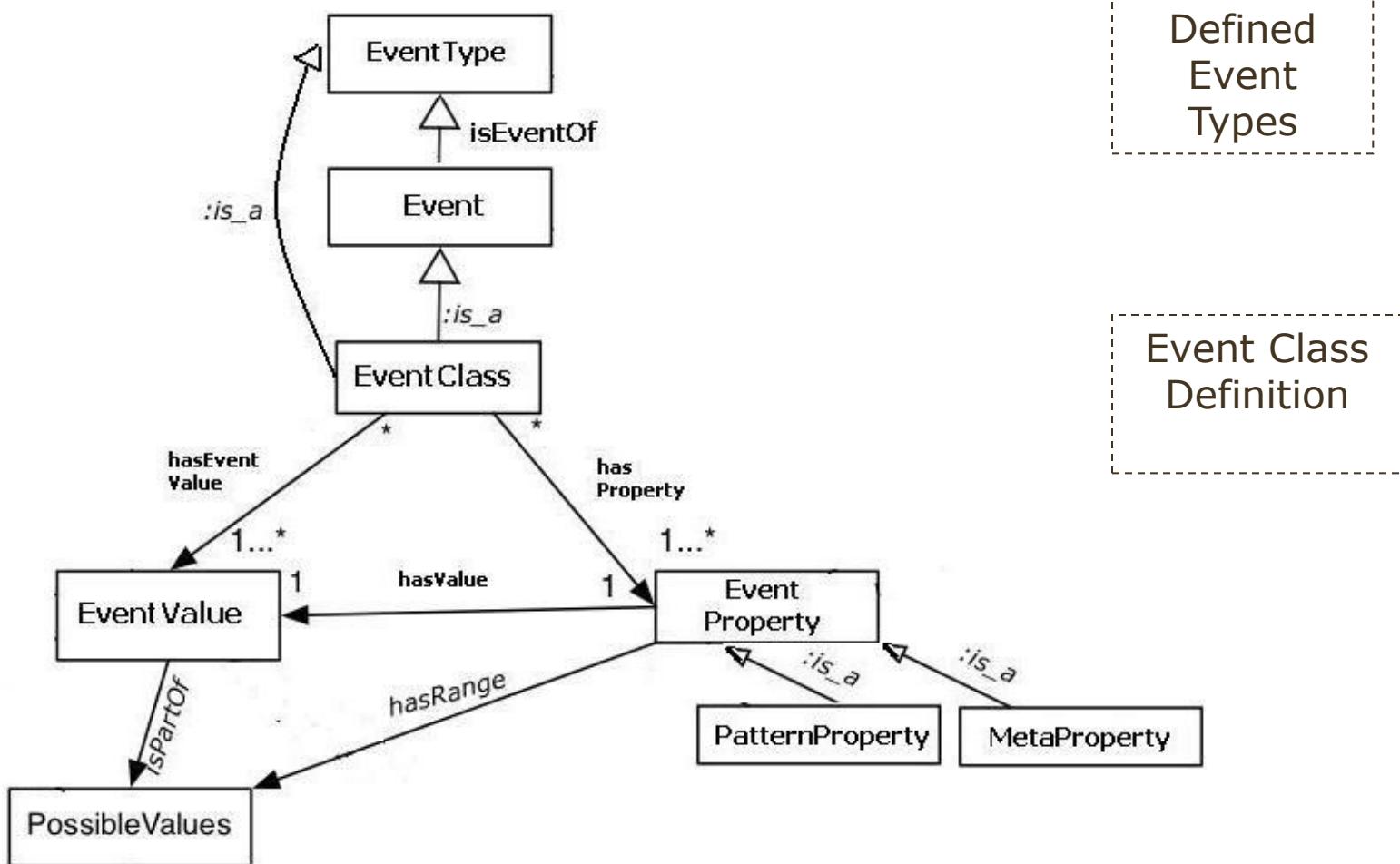


Example: Situation Top Ontology Metamodel



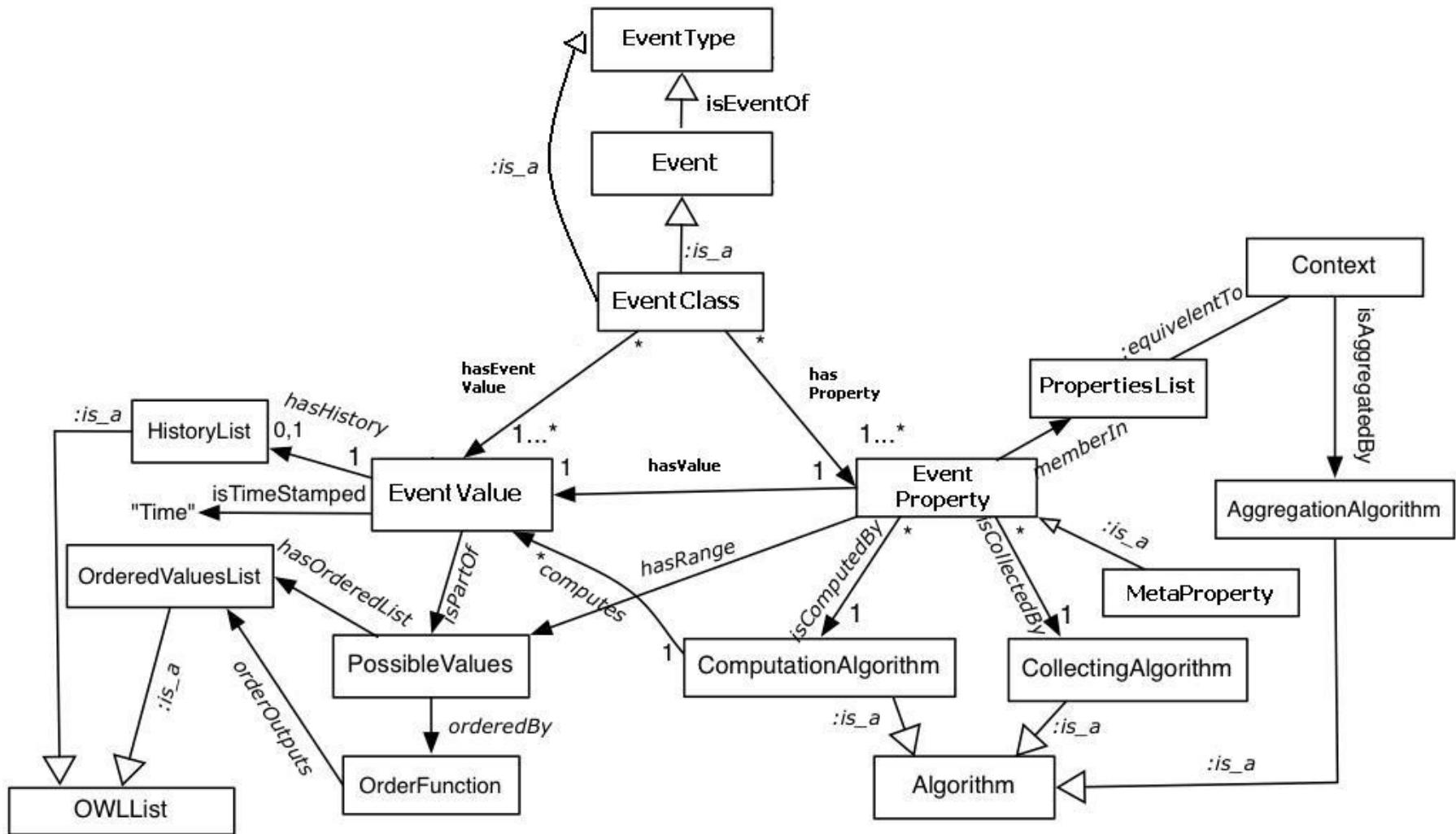
Example - Event MetaModel

(for defining Event Types as Instances of the MetaModel Event Class)



Extended Event Meta Model Ontology

(with computational properties and complex value definitions)



The Generic Syntax Pattern of An Event

```
<Event @key @keyref @iri @type>
```

references to external ontologies

```
<!-- event info and life cycle management, modularization -->
<meta>  <!-- R: (semantic) metadata of the event -->           </meta>
<scope>  <!-- R: scope of the event -->                      </scope>
```

```
<!-- event pattern description -->
<evaluation><!-- R: semantics, e.g. selection, consumption-->
</evaluation>
<signature>  <!-- R: event pattern declaration -->          </signature>
```

```
<!-- event instance -->
<qualification> <!-- R: e.g. qualifying event declarations, e.g.
                  priorities, validity, strategy -->           </qualification>
<quantification> <!-- R: quantifying rule declarations --> </quantification>
<oid>            <!-- R: object identifier -->                   </oid>
<content>        <!-- R: event instance content -->             </content>
</Event>
```

*Complex Event **Pattern** Definition*

```
<Event key="#ce2" type="&ruleml;ComplexEvent">
  <signature> <!-- pattern signature definition -->
    <Sequence>
      <signature>
        <Event type="&ruleml;SimpleEvent">
          <signature><Event>...event_A...</Event></signature>
        </Event>
      </signature>
      <signature><Event type="&ruleml;ComplexEvent" keyref="ce1"/></signature>
      <signature>
        <Event type="cbe:CommonBaseEvent" iri="cbe.xml#xpointer(/CommonBaseEvent)" />
      </signature>
    </Sequence>
  </signature>
</Event>

<Event key="#ce1">
  <signature> <!-- event pattern signature -->
    <Concurrent>
      <Event><meta><Time>...t3</Time></meta><signature>...event_B</signature></Event>
      <Event><meta><Time>...t3</Time></meta><signature>...event_C</signature></Event>
    </Concurrent>
  </signature>
</Event>

<Event key="#e1" keyref="#ce2"><content>...</content></Event>
```

Event Pattern defining Event Templates signature

Event Instance

Rule Interface and Rule Implementation

```
<Rule style="active">
  <evaluation>
    <Profile> e.g. selection and consumptions policies </Profile>
  </evaluation>
  <signature>
    <Event key="#ce2">
      ... see example from the previous slide
    </Event>
  </signature>
  <on>
    <Event keyref="#ce2"/> <!-- use defined event pattern for detecting events -->
  </on>
  <do>
    <Assert safety="transactional"> <!-- transactional update -->
      <formula>
        <Atom>
          <op><Rel>likes</Rel></op>
          <arg><Ind>John</Ind></arg>
          <arg><Var>X</Var></arg>
        </Atom>
      </formula>
    </Assert>
  </do>
</Rule>
```

Interface

(Note in reaction rules the signature is the event pattern which detects the event instances)

Implementation

Execution Semantics

(can be defined in the <evaluation> semantics element)

1. Definition

- Definition of event/action pattern e.g. by event algebra
- Based on declarative formalization or procedural implementation
- Defined over an atomic instant or an interval of time, events/actions, situation, transition etc.

2. Selection

- Defines selection function to select one event from several occurred events (stored in an event instance sequence e.g. in memory, database/KB) of a particular type, e.g. “first”, “last”
- Crucial for the outcome of a reaction rule, since the events may contain different (context) information, e.g. different message payloads or sensing information

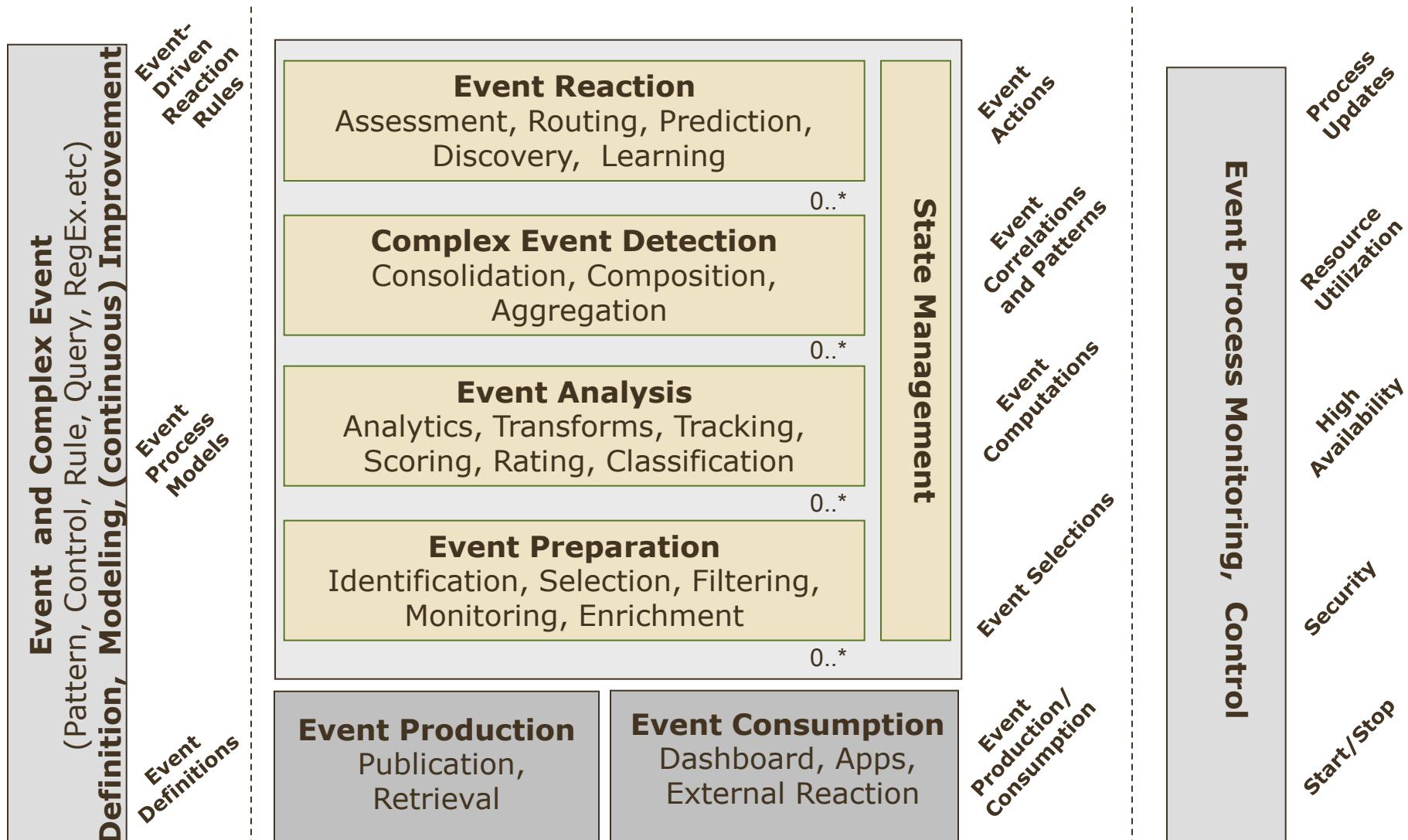
3. Consumption

- Defines which events are consumed after the detection of a complex event
- An event may contribute to the detection of several complex events, if it is not consumed
- Distinction in event messaging between “multiple receive” and “single receive”
- Events which can no longer contribute, e.g. are outdated, should be removed

4. Execution

- Actions might have an internal effect i.e. change the knowledge state leading to state transition from (pre-)condition state to post-condition state
- The effect might be hypothetical (e.g. a hypothetical state via a computation) or persistent (update of the knowledge base),
- Actions might have an external side effect

EPTS - Reference Architecture: Functional View



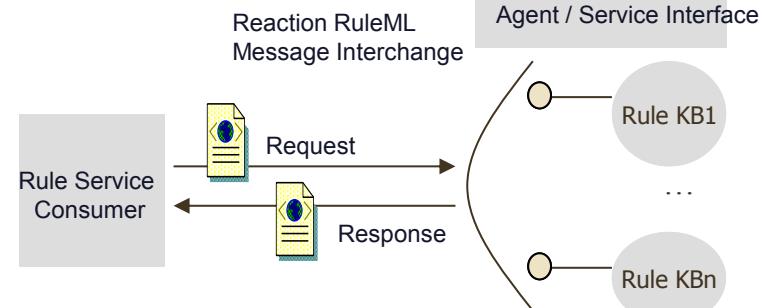
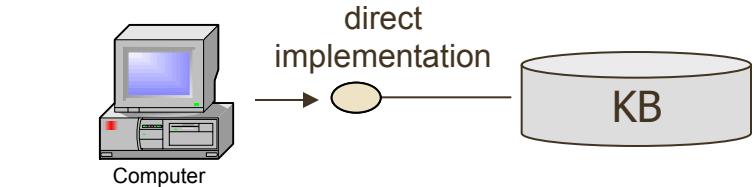
Design time

Run time

Administration

Coupling Approaches for (Distributed) Reaction RuleML Bases

- **Strong coupling**
 - Interaction through a stable interface
 - **API call is hard coded**
- **Loose coupling**
 - Resilient relationship between two or more systems or organizations with some kind of exchange relationship
 - Each end of the transaction makes its requirements explicit, e.g. as an **interface description**, and makes few assumptions about the other end
- **Decoupled**
 - **de-coupled in time using (event) messages** (e.g. via Message-oriented Middleware (MoM))
 - Often asynchronous stateless communication (e.g. publish-subscribe or CEP event detection)



Send and Receive Messaging

- **Send a message**
- ***Send @directive (oid,protocol,agent,content)***
- **Receive a message**
- ***Receive @directive (oid,protocol,agent,content)***

- *oid* is the conversation identifier (enabling also [subconverstations](#))
- *protocol*: protocol definition (high-level protocols and transport prot.)
- *agent*: denotes the target or sender of the message
- *@directive*: pragmatic context, e.g. FIPA Agent Communication Language (ACL) primitives
- *content*: Message payload

Messaging Reaction Rules

```
<Rule>
```

...

```
<do><Send><oid>xid1</oid> ...query1 </Send></do>
```

```
<do><Send><oid>xid2</oid> ...query2 </Send></do>
```

```
<on><Receive><oid>xid2</oid> ...response2 </Receive></on>
```

<if> prove some conditions, e.g. make decisions on the received answers **</if>**

```
<on><Receive><oid>xid1</oid> ...response1 </Receive></on>
```

....

```
</Rule>
```

Note: The „on“, „do“, „if“ parts can be in arbitrary combinations to allow for a flexible workflow-style logic with subconversations and parallel branching logic

Reaction - Routing

Prova: Example with Agent (Sub-) Conversations

```
rcvMsg(XID, esb, From, query-ref, buy(Product) :-  
    routeTo(Agent, Product), % derive processing agent  
    % send order to Agent in new subconversation SID2  
    sendMsg(SID2, esb, Agent, query-ref, order(From, Product)),  
    % receive confirmation from Agent for Product order  
    rcvMsg(SID2, esb, Agent, inform-ref, oder(From, Product)).  
  
    % route to event processing agent 1 if Product is luxury  
routeTo(epa1, Product) :- luxury(Product).  
    % route to epa 2 if Product is regular  
routeTo(epa2, Product) :- regular(Product).  
  
    % a Product is luxury if the Product has a value over ...  
luxury(Product) :- price(Product, Value), Value >= 10000.  
    % a Product is regular if the Product ha a value below ...  
regular(Product) :- price(Product, Value), Value < 10000.
```

corresponding serialization with Reaction RuleML <Send> and <Receive>

Reaction RuleML Messaging

```
<Message directive="<!-- pragmatic context -->" >  
  <oid>  <!-- conversation ID-->  </oid>  
  <protocol> <!-- transport protocol --> </protocol>  
  <sender> <!-- sender agent/service --> </sender>  
  <receiver> <!-- receiver agent/service --> </receiver>  
  <content> <!-- message payload --> </content>  
</Message>
```

- *oid* is the conversation identifier (enabling also subconverstations)
- *protocol*: protocol definition (high-level protocols and transport prot.)
- *agent (send, receiver)*: denotes the target or sender of the message
- *@directive*: pragmatic context, e.g. FIPA ACL primitives
- *content*: Message payload

Loosley-Coupled Communication via Messages to Agent Interface

```
<Message directive="acl:query-ref">
  <oid> <Ind>conversation1</Ind> </oid>
  <protocol> <Ind>esb</Ind> </protocol>
  <sender> <Ind>Agent1</Ind> </sender>
  <receiver> <Ind>Agent2</Ind> </receiver>
  <content>
    <Atom>
      <Rel>likes</Rel>
      <Ind>John</Ind>
      <Ind>Mary</Ind>
    </Atom>
  </content>
</Message>
```

FIPA ACL directive

Interpreted by Agent 2 as
query according to
ACL:query-ref

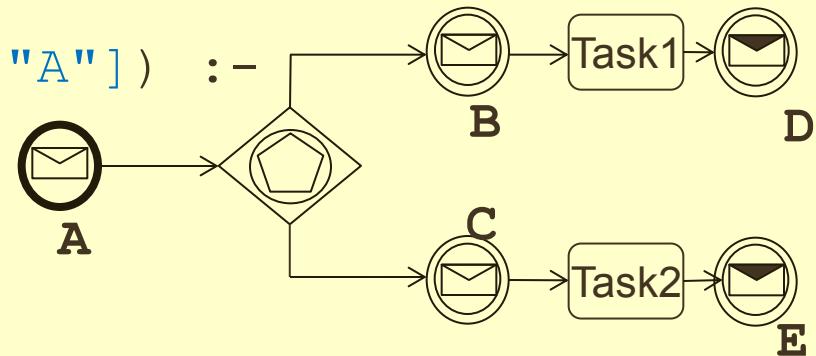
Note: the receiver „Agent2“
needs to specify an
appropriate <signature> for
„likes“

- Event Message is local to the conversation state (oid) and pragmatic context (directive)

Message Driven Routing

Prova : Event Routing in Event-Driven Workflows

```
rcvMsg(XID, Process, From, event, ["A"]) :-  
    fork_b_c(XID, Process).
```



```
fork_b_c(XID, Process) :-  
    @group(p1) rcvMsg(XID, Process, From, event, ["B"] ),  
    execute(Task1), sendMsg(XID, self, 0, event, ["D"] ).
```

```
fork_b_c(XID, Process) :-  
    @group(p1) rcvMsg(XID, Process, From, event, ["C"] ),  
    execute(Task2), sendMsg(XID, self, 0, event, ["E"] ).
```

```
fork_b_c(XID, Process) :-  
    % OR reaction group "p1" waits for either of the two  
    event message handlers "B" or "C" and terminates the  
    alternative reaction if one arrives  
    @or(p1) rcvMsg(XID, Process, From, or, _ ).
```

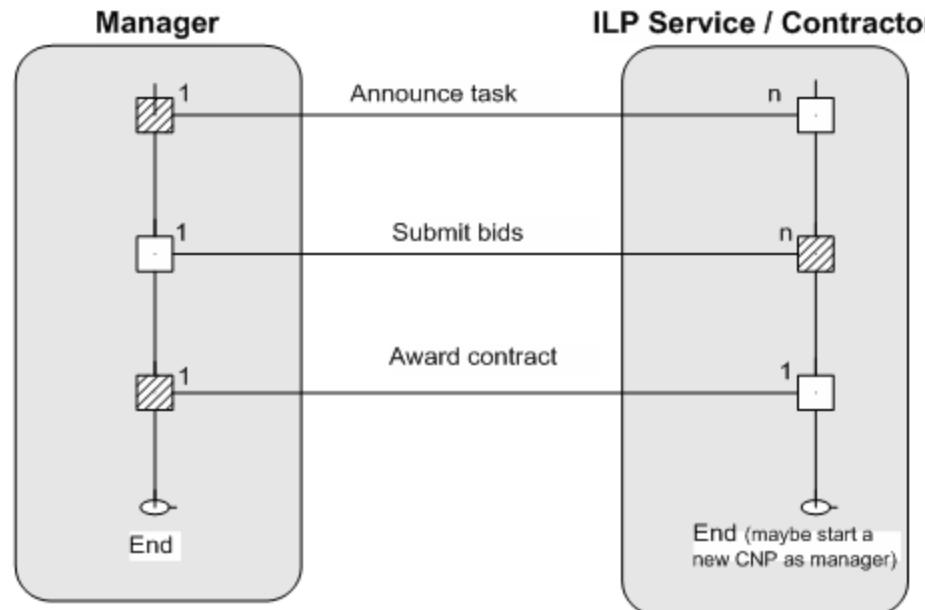
Distributed Rule Base Interchange Prova: Mobile Code

```
% Manager
```

```
upload_mobile_code(Remote,File) :  
    Writer = java.io.StringWriter(), % Opening a file fopen(File,Reader),  
    copy(Reader,Writer),  
    Text = Writer.toString(),  
    SB = StringBuffer(Text),  
    sendMsg(XID,esb,Remote,eval,consult(SB)).
```

```
% Service (Contractor)
```

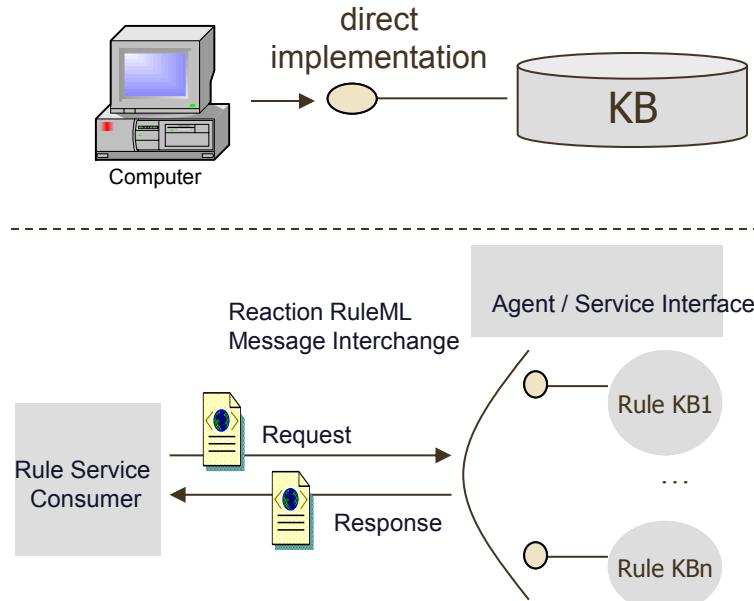
```
rcvMsg(XID,esb,Sender,eval,[Predicate|Args]) :- derive([Predicate|Args]).
```



Contract Net Protocol

Coupling Approaches for (Distributed) Reaction RuleML Bases

- **Strong coupling**
 - Interaction through a stable interface
 - **API call is hard coded**
- **Loose coupling**
 - Resilient relationship between two or more systems or organizations with some kind of exchange relationship
 - Each end of the transaction makes its requirements explicit, e.g. as an **interface description**, and makes few assumptions about the other end
- **Decoupled**
 - **de-coupled in time using (event) messages** (e.g. via Message-oriented Middleware (MoM))
 - Often asynchronous stateless communication (e.g. publish-subscribe or CEP event detection)

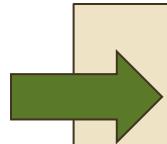


Decoupled Interaction: Event Messaging Reaction Rule

```
... sendMsg(SID, stream, "S&P500", inform, tick( (name, N), (price, P),  
(time, T)) ...
```

publish

decoupled



Event Stream

```
{(Name, "OPEL")(Price, 45)(Volume, 2000)(Time, 1) }  
{(Name, "SAP")(Price, 65)(Volume, 1000) (Time, 2)}
```

consum / subscribe

```
rcvMult(SID, stream, "S&P500", inform,  
        tick(Name^^car:Major_corporation, P^^currency:Dollar,  
              T^^time:Timepoint) ) :- ...<further processing of semantic event> .
```



```
{(OPEL, is_a, car_manufacturer),  
(car_manufacturer, build, Cars),  
(Cars, are_build_from, Metall),
```

```
(OPEL, hat_production_facilities_in, Germany),
```

```
(Germany, is_in, Europe)
```

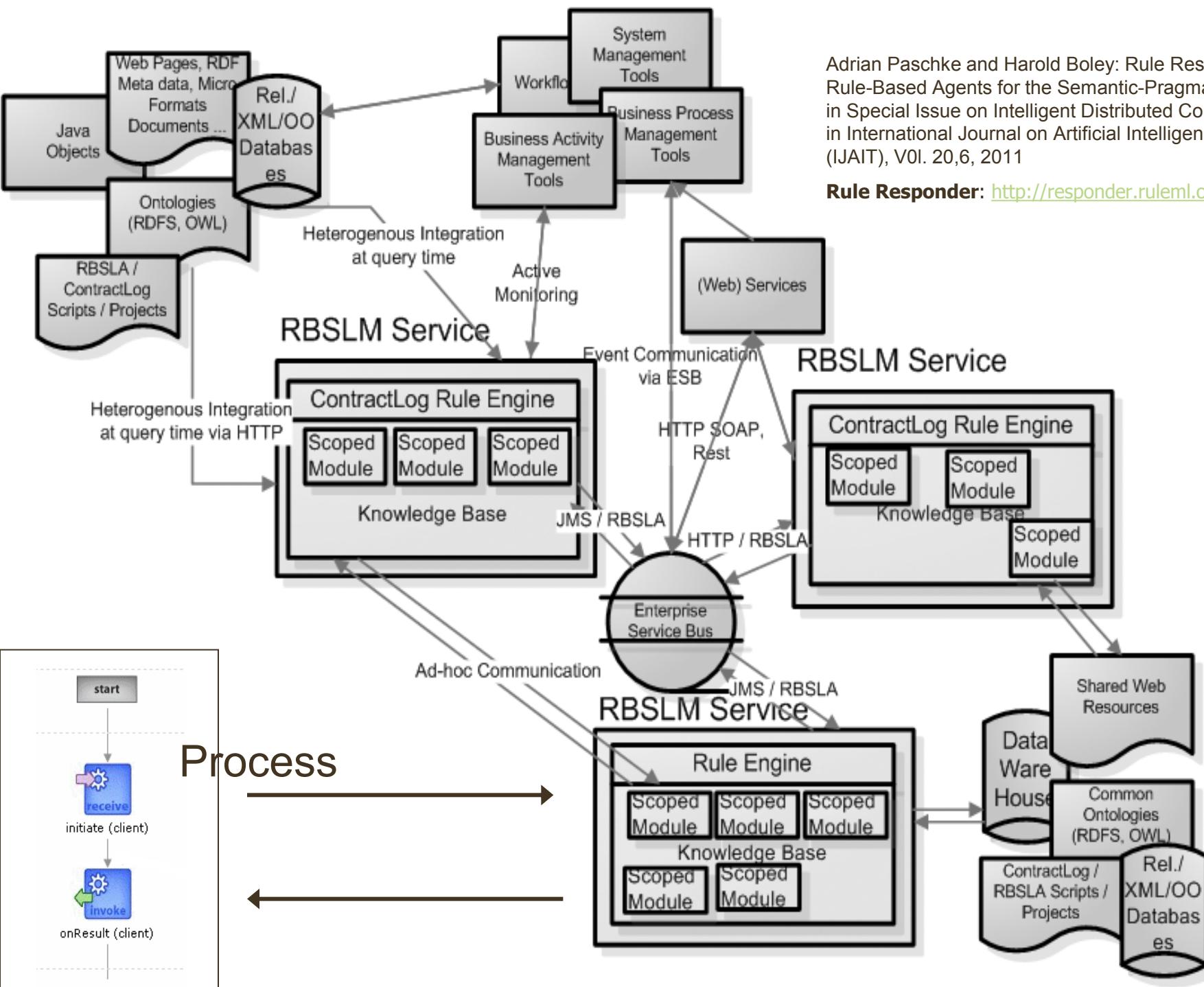
```
(OPEL, is_a, Major_corporation),
```

```
(Major_corporation, have, over_10,000_employees)}
```

Semantic
Knowledge Base

Adrian Paschke and Harold Boley: Rule Responder: Rule-Based Agents for the Semantic-Pragmatic Web, in Special Issue on Intelligent Distributed Computing in International Journal on Artificial Intelligence Tools (IJAIT), V0l. 20,6, 2011

Rule Responder: <http://responder.ruleml.org>



Summary of Selected Reaction RuleML Features

- Different Rule Families and Types (DR, KR, PR, ECA, CEP)
- Support for Event/Action/Situation... Reasoning and Processing
- Rule Interface and Implementation for Distributed Knowledge
- Test Cases for Verification, Validation and Integrity (VVI Testing)
- Knowledge Life Cycle Management, Modularization and Dynamic Views (metadata, scopes, guards, situations/fluents, update actions)
- Decoupled event messaging and loosely-coupled send/receive interaction against rule (KB) interface within conversations, coordination/negotiation protocols and pragmatic directives
- Different detection, selection and consumption semantics (**profiles**)
- Different algebras (Event, Action, Temporal,...)
- Specialized Language Constructs
 - Intervals (Time, Event), situations (States, Fluents)
- Top Level Meta Model for semantic Type Definitions and predefined Types in Top-Level Ontologies
- External (event) query languages, external data and ontology models
- ...

Agenda

- **Introduction to Event Processing**
- **Semantic Complex Event Processing (SCEP)**
- **Event Processing Technical Society (EPTS)**
 - Event Processing Standards Reference Model
 - Event Processing Reference Architecture
- **Event Processing Function Patterns - Examples**
 - Implementation Examples in the Prova Rule Engine (**Platform Specific**)
- **Reaction RuleML Standard**
 - Standardized Semantic Reaction Rules (**Platform Independent**)
- **Summary**

Summary

- **Semantic Rule-Based Event Processing (SCEP)**
 - complex event processing + semantic technologies
- **SCEP rule agents will be able to**
 - understand what is happening in terms of "real-time" events.
 - know which reactions and processes it can invoke and what events it can signal.
- **Reference Architecture and Metamodel provides a common set of EP functions that may be mapped to other EP ontologies / taxonomies**
 - Provides the basis for EP Patterns
 - EP Patterns can be semantic, design or runtime focused

Questions ?



Acknowledgment to the EPTS Reference Architecture working group members

Acknowledgement to the members of the Reaction RuleML technical group

Acknowledgement to the members of the Corporate Semantic Web group at FU Berlin

Further Reading – Surveys and Tutorials

- **Paschke, A., Boley, H.: Rules Capturing Event and Reactivity, in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, ISBN:1-60566-402-2, 2009**
<http://www.igi-global.com/book/handbook-research-emerging-rule-based/465>
Sample Chapter: <http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=16435934&article=0&fd=pdf>
- **Adrian Paschke, Alexander Kozlenkov: Rule-Based Event Processing and Reaction Rules. RuleML 2009: 53-66**
http://link.springer.com/chapter/10.1007%2F978-3-642-04985-9_8
- **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>
- **Adrian Paschke: The Reaction RuleML Classification of the Event / Action / State Processing and Reasoning Space. CoRR abs/cs/0611047 (2006)**
<http://arxiv.org/ftp/cs/papers/0611/0611047.pdf>
- **Jon Riecke, Opher Etzion, François Bry, Michael Eckert, Adrian Paschke, Event Processing Languages, Tutorial at 3rd ACM International Conference on Distributed Event-Based Systems. July 6-9, 2009 - Nashville, TN**
<http://www.slideshare.net/opher.etzion/debs2009-event-processing-languages-tutorial>
- **Paschke, A., Boley, H.: Rule Markup Languages and Semantic Web Rule Languages, in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, IGI Publishing, ISBN:1-60566-402-2, 2009**
<http://www.igi-global.com/chapter/rule-markup-languages-semantic-web/35852>

Further Reading – RuleML and Reaction RuleML

- **Harold Boley, Adrian Paschke, Omair Shafiq: RuleML 1.0: The Overarching Specification of Web Rules.** RuleML 2010: 162-178
[http://dx.doi.org/10.1007/978-3-642-16289-3 15](http://dx.doi.org/10.1007/978-3-642-16289-3_15)
<http://www.cs.unb.ca/~boley/talks/RuleML-Overarching-Talk.pdf>
- **Adrian Paschke, Harold Boley, Zhili Zhao, Kia Teymourian and Tara Athan: Reaction RuleML 1.0: Standardized Semantic Reaction Rules,** 6th International Conference on Rules (RuleML 2012), Montpellier, France, August 27-31, 2012
[http://link.springer.com/chapter/10.1007%2F978-3-642-32689-9 9](http://link.springer.com/chapter/10.1007%2F978-3-642-32689-9_9)
<http://www.slideshare.net/swadpasc/reaction-ruleml-ruleml2012paschketutorial>
- **Paschke, A., Boley, H.: Rule Markup Languages and Semantic Web Rule Languages, in Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches,** IGI Publishing, ISBN:1-60566-402-2, 2009
<http://www.igi-global.com/chapter/rule-markup-languages-semantic-web/35852>
Sample chapter: <http://www.csw.inf.fu-berlin.de/teaching/HandbookSemanticWebRules.pdf>
- **Paschke, A., Boley, H.: Rules Capturing Event and Reactivity,** in **Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches,** IGI Publishing, ISBN:1-60566-402-2, 2009
<http://www.igi-global.com/book/handbook-research-emerging-rule-based/465>
Sample Chapter: <http://nparc.cisti-icist.nrc-cnrc.gc.ca/npsi/ctrl?action=rtdoc&an=16435934&article=0&fd=pdf>
- **Adrian Paschke and Harold Boley: Rule Responder: Rule-Based Agents for the Semantic-Pragmatic Web,** in Special Issue on Intelligent Distributed Computing in International Journal on Artificial Intelligence Tools (IJAIT), Vol. 20,6, 2011

Further Reading – EPTS Event Processing Reference Architecture and Event Processing Patterns

- Adrian Paschke, Paul Vincent, Alexandre Alves, Catherine Moxey: Advanced design patterns in event processing. DEBS 2012: 324-334;
<http://www.slideshare.net/isvana/epts-debs2012-event-processing-reference-architecture-design-patterns-v204b>
- Adrian Paschke, Paul Vincent, Alexandre Alves, Catherine Moxey: Architectural and functional design patterns for event processing. DEBS 2011: 363-364
<http://www.slideshare.net/isvana/epts-debs2011-event-processing-reference-architecture-and-patterns-tutorial-v1-2>
- Paschke, A., Vincent, P., and Catherine Moxey: Event Processing Architectures, *Fourth ACM International Conference on Distributed Event-Based Systems* (DEBS '10). ACM, Cambridge, UK, 2010.
<http://www.slideshare.net/isvana/debs2010-tutorial-on-epts-reference-architecture-v11c>
- Paschke, A. and Vincent, P.: A reference architecture for Event Processing. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems* (DEBS '09). ACM, New York, NY, USA, 2009.
- Francois Bry, Michael Eckert, Opher Etzion, Adrian Paschke, Jon Riche: Event Processing Language Tutorial, DEBS 2009, ACM, Nashville, 2009
<http://www.slideshare.net/opher.etzion/debs2009-event-processing-languages-tutorial>
- Adrian Paschke. A Semantic Design Pattern Language for Complex Event Processing, AAAI Spring Symposium "Intelligent Event Processing", March 2009, Stanford, USA
<http://www.aaai.org/Papers/Symposia/Spring/2009/SS-09-05/SS09-05-010.pdf>
- Paschke, A.: Design Patterns for Complex Event Processing, 2nd International Conference on Distributed Event-Based Systems (DEBS'08), Rome, Italy, 2008.
<http://arxiv.org/abs/0806.1100v1>

Further Reading – Rule-Based Semantic CEP

- Corporate Semantic Web – Semantic Complex Event Processing
 - <http://www.corporate-semantic-web.de/semantic-complex-event-processing.html>
- Adrian Paschke: ECA-RuleML: An Approach combining ECA Rules with temporal interval-based KR Event/Action Logics and Transactional Update Logics CoRR abs/cs/0610167: (2006); <http://arxiv.org/ftp/cs/papers/0610/0610167.pdf>
- Adrian Paschke: Semantic Rule-Based Complex Event Processing. RuleML 2009: 82-92 [http://link.springer.com/chapter/10.1007%2F978-3-642-04985-9 10](http://link.springer.com/chapter/10.1007%2F978-3-642-04985-9_10)
- Adrian Paschke, Alexander Kozlenkov, Harold Boley: A Homogeneous Reaction Rule Language for Complex Event Processing, In Proc. 2nd International Workshop on Event Drive Architecture and Event Processing Systems (EDA-PS 2007) at VLDB 2007; <http://arxiv.org/pdf/1008.0823v1.pdf>
- Adrian Paschke: ECA-LP / ECA-RuleML: A Homogeneous Event-Condition-Action Logic Programming Language CoRR abs/cs/0609143: (2006); <http://arxiv.org/ftp/cs/papers/0609/0609143.pdf>
- Teymourian, K., Coskun, G., and Paschke, A. 2010. Modular Upper-Level Ontologies for Semantic Complex Event Processing. In Proceeding of the 2010 Conference on Modular ontologies: Proceedings of the Fourth international Workshop (WoMO 2010) O. Kutz, J. Hois, J. Bao, and B. Cuenca Grau, Eds. Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, 81-93. <http://dl.acm.org/citation.cfm?id=1804769>
- Adrian Paschke and Harold Boley: Rule Responder: Rule-Based Agents for the Semantic-Pragmatic Web, in Special Issue on Intelligent Distributed Computing in International Journal on Artificial Intelligence Tools (IJAIT), Vol. 20,6, 2011
- Kia Teymourian, Adrian Paschke: Towards semantic event processing. DEBS 2009

Further Reading –Rules and Logic Programming, Prova

- **Adrian Paschke: Rules and Logic Programming for the Web.** A. Polleres et al. (Eds.): Reasoning Web 2011, LNCS 6848, pp. 326–381, 2011.
http://dx.doi.org/10.1007/978-3-642-23032-5_6
http://rw2011.deri.ie/lectures/05Paschke_Rules_and%20Logic_Programming_for_the_Web.pdf
- **Prova Rule Engine** <http://www.prova.ws/>
 - Prova 3 documentation <http://www.prova.ws/index.html?page=documentation.php>
 - Journal: Adrian Paschke and Harold Boley: Rule Responder: Rule-Based Agents for the Semantic-Pragmatic Web, in Special Issue on Intelligent Distributed Computing in International Journal on Artificial Intelligence Tools (IJAIT), Vol. 20,6, 2011
 - **Prova 3 Semantic Web Branch**
 - Paper: Paschke, A.: A Typed Hybrid Description Logic Programming Language with Polymorphic Order-Sorted DL-Typed Unification for Semantic Web Type Systems, OWL-2006 (OWLED'06), Athens, Georgia, USA.
 - pre-compiled Prova 3 version with Semantic Web support from <http://www.cs.wiwi.fu-berlin.de/teaching/ws1213/prova-sw.zip>.
(The Java sources of the Prova 3 Semantic Web are managed on GitHub (<https://github.com/prova/prova/tree/prova3-sw>)
 - Prova ContractLog KR: <http://www.rbsla.ruleml.org/ContractLog.html>
 - Paschke, A.: Rule-Based Service Level Agreements - Knowledge Representation for Automated e-Contract, SLA and Policy Management, Book ISBN 978-3-88793-221-3, Idea Verlag GmbH, Munich, Germany, 2007.
<http://rbsla.ruleml.org>
 - Prova CEP examples: <http://www.slideshare.net/isvana/epts-debs2012-event-processing-reference-architecture-design-patterns-v204b>