Event Processing Reference Architecture - Design Patterns

Adrian Paschke  
(Freie Universitaet Berlin)

Paul Vincent  
(TIBCO Software)

Catherine Moxey, Martin Hirzel,  
Dave Rushall, Richard Jacks (IBM)

Alex Alves  
(Oracle)
Agenda

• EPTS Reference Architecture Introduction

• EP Patterns:
  – EPTS RA patterns
  – Some Use Cases
  – EPIA patterns
Introduction to Event Processing

Event Processing,
Complex Event Processing,
Event Stream Processing

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

Event Producer
(Event Source, Event Emitter)

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

Events

Derived Events

Event Management

Design time
Run time
Administration
Reference Architecture: Functional View

**Event Production**
- Publication, Retrieval

**Event Consumption**
- Dashboard, Apps, External Reaction

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

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

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

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

**State Management**

- Event Actions
- Event Correlations and Patterns
- Event Computations
- Event Selections
- Event Production/Consumption

**Event Process Monitoring, Control**
- Process Updates
- Resource Utilization
- High Availability
- Security
- Start/Stop

**Design time**

**Run time**

**Administration**
Comparison of EPTS RA Fns and EPIA EP Agents

**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

---

EPA

- Filter
- Transformation
- Pattern detect

- Translate
- Aggregate
- Split
- Compose

- Enrich
- Project
Agenda

• EP Meta Pattern / Pattern Structure
What do we want to cover in an EP Pattern?

**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.
EP MetaPattern / Pattern Description (version 1)

• **Name & Alternative Names**

• **Classification versus other references**
  – EPTS RA and other references e.g. “EP in Action” book

• **Description**
  – Role, and Business Rule type specification / requirement

• **Structure**
  – EPTS Glossary terms

• **Implementations**
EP MetaPattern future versions?

• **Name & Alternative Names**

• **Classification versus other references**
  – EPTS RA and other references e.g. “EP in Action” book

• **Description**
  – Role, and Business Rule type

• **Structure**
  – EPTS Glossary terms

• **Implementations**
  
  - may revise/improve EP Fns classification / ontology
  - compare/extend other “event patterns”
  - Extend examples pseudocode – real example
EP Pattern Implementations

• Differ in Implementation type (Event Processing Language, Execution semantics, etc)

• Sampled systems we cover:

1. **TIBCO BusinessEvents** Rete-production-rule-engine (with continuous query and event-expression options)

2. **Oracle Event Processing** Stream-processing engine

3. **Prova** Logic Programming Semantic CEP Rule Engine

4. **IBM Decision Server Events** component of WebSphere Operational Decision Management & **IBM InfoSphere Streams** stream processing engine
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)

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

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

Information
- Stored Events and Data

Continuously process events using procedural and declarative event processing elements
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

Event Production
Publication, Retrieval, Identification, Selection, Filtering, Enrichment, Transforms, Routing

Event Preparation
Identification, Selection, Filtering, Monitoring, Enrichment

Complex Event Detection
Consolidation, Composition, Aggregation

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

Event Reaction
Assessment, Routing, Prediction, Discovery, Learning

Event Consumption
Dashboard, Apps, External Reaction
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
Agenda

• **Functional Reference Architecture-based EP Patterns**
Patterns Coverage

Event Reaction
Assessment, Routing, Prediction, Discovery, Learning

Complex Event Detection
Consolidation, Composition, Aggregation

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

Event Preparation
Identification, Selection, Filtering, Monitoring, Enrichment

Event Production
Application Time, Publication, Retrieval

Event Consumption
Dashboard, Apps, External Reaction

State Management
Event Actions
Event Correlations and Patterns
Event Computations
Event Selections
Event Production/Consumption
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.*
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)”

• **EPIA:**
  – Transform, Translate, Enrich / Project
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):
    \[
    \text{All <event entities> whose <attribute1> matches the <attribute2> of <some other event or data or type> is related by <relationship> to that <some other event or data or type>.}
    \]
• 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

- TIBCO, Prova, Oracle and IBM implementations for this pattern

• **General pattern:**
  <declare> event, data
  <if> event-data-match
  <then> create-event-association-with-data

• **Effect:** find appropriate data that matches the incoming event* and create a reference between the event and the data

* note: event object = concept: input events are immutable**
** note: except in PreProcessor 😊

• **General pattern:**

\[
\text{SELECT } \langle \text{property} \rangle \\
\text{FROM } \langle \text{SOURCE} \rangle \\
\text{WHERE } \langle \text{predicate-condition} \rangle
\]

• **Effect:** matches events coming from \langle \text{SOURCE} \rangle with pre-defined event-type
Preparation: Identification: Implementations: Prova

Semantic Identification

- General pattern:
  \(<rcvMsg / rcvMult> \text{ Event } ^{\text{ Semantic Type }} :-\)

- 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

Event Stream

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

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

Identify event and associate with semantic background knowledge

Semantic Knowledge Base

\{(\text{OPEL, is_a, car_manufacturer}),
(\text{car_manufacturer, build, Cars}),
(\text{Cars, are_build_from, Metall}),
(\text{OPEL, hat_production_facilities_in, Germany}),
(\text{Germany, is_in, Europe})
(\text{OPEL, is_a, Major_corporation}),
(\text{Major_corporation, have, over_10,000_employees})\}
Preparation: Identification: Implementations: IBM WODM Decision Server Events

- **Example**: event rule relates to ‘new purchase order’ events, and the events are processed in the context of the customer number.

- **Effect**: presence of ‘new purchase order’ in the rule identifies that event type, and the events are processed in the context of the customer number field – the events of interest could be said to be **identified** by ‘new purchase order’ type and customer number, *or* the latter could be regarded as selection.
Preparation:Identification:Implementations: IBM InfoSphere Streams (SPL)

• Example:

```c

type Kind = enum { knownKind1, knownKind2, knownKind3 };

arseille...

stream<rstring key, int32 payload> Events = PrepId_EventsSource() {}  
stream<rstring key, Kind kind> Mappings = PrepId_MappingsSource() {}  
stream<Kind kind, int32 payload> KnownEvents = Custom(Events; Mappings) {
  logic
  state:
    mutable map<rstring, Kind> _keyToKind = {};
  onTuple Events:
    if (key in _keyToKind)
      submit({ Kind=_keyToKind[key], payload=payload }, KnownEvents);
  onTuple Mappings:
    _keyToKind[key] = kind;
}
```

**Example:**

- **Stream Preparation**
  - **Event Preparation** (Preparation)
    - Identification, Selection, Filtering, Monitoring, Enrichment
  - **Stream Mapping**
    - Custom function
      - Filter to map Event key to known kind
  - **Known Events**
    - Identified known events

---

**Diagram:**

- Events stream
- Mapping stream
- Custom function
- Known Events

---

**Example Code:**

```c

type Kind = enum { knownKind1, knownKind2, knownKind3 };

arseille...

stream<rstring key, int32 payload> Events = PrepId_EventsSource() {}  
stream<rstring key, Kind kind> Mappings = PrepId_MappingsSource() {}  
stream<Kind kind, int32 payload> KnownEvents = Custom(Events; Mappings) {
  logic
  state:
    mutable map<rstring, Kind> _keyToKind = {};
  onTuple Events:
    if (key in _keyToKind)
      submit({ Kind=_keyToKind[key], payload=payload }, KnownEvents);
  onTuple Mappings:
    _keyToKind[key] = kind;
}
```
RA: Preparation: Selection:

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

**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
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."

• **EPIA:**
  
  Filter (EPA or Event Processing Agent) performs filtering only and has no matching or derivation steps, so it does not transform the input event.
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>.

<table>
<thead>
<tr>
<th>event</th>
<th>Filter f(event)</th>
<th>event</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>discarded event</td>
<td></td>
</tr>
</tbody>
</table>
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

- TIBCO, Oracle, Prova and IBM implementations for this pattern

• **General pattern:**
  
  `<declare> event`

  `<if> event-boolean-expression`

  `<then> update-event-with-fact OR action-for-filtered-event`

• **Effect:** ignore filtered events on operations on non-filtered events

- **Implementation type:** Rete-based rule-driven system

- **Alternate pattern:**
  
  `<declare> event`
  
  `<if>`
  NOT event-boolean-expression
  
  `<then>`
  consume-event

- **Effect:** removed filtered events from system (and ALL subsequent operations!)
Preparation: Filter: Implementations: TIBCO Business Events: State transitions

- **General pattern:**
  
  \(<\text{state models for concept C}>\>
  
  \(<\text{route from State n to n+1}>\>
  
  \(<\text{on}>\text{ event}\>
  
  \(<\text{where}>\text{ event-filter}\>

- **Effect:** change state for some concept

- **Notes:** this is a specialisation of the general rule pattern!
Preparation: Filter: Implementations: TIBCO Business Events: Queries

• **General pattern:**
  `<select> output  
  <from> event  
  <where> event-filter

  <policy> notification

• **Effect:** collect filtered events for event operations per some policy

• **Note:** the pattern matching framework has a similar notion but a different surface syntax

• **General pattern:**

  SELECT <property>
  FROM <SOURCE>
  WHERE <predicate-condition>

• **Effect:** discard events whose predicate evaluate to false

• **Scenario:**
  - Select only stocks from the stock tick stream whose symbols are ‘AAA’.

• **Application Model**

Specify STREAM source

```sql
SELECT *
FROM StockTickStream
WHERE symbol = 'AAA'
```

Define predicate for filtering

• **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,...,Argₙ): Message payload
% Filter for stocks starting with „A“ and price > 100

\[
\text{rcvMult}(\text{SID}, \text{stream}, "S&P500", \text{inform}, \text{tick}(S, P, T)) : - \\
S = "A.*", \\
P > 100, \\
\text{sendMsg}(\text{SID2}, \text{esb}, "epa1", \text{inform}, \text{happens}(\text{tick}(S, P), T)).
\]

Example with rule chaining

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

\[
\text{rcvMult}(\text{SID}, \text{stream}, "S&P500", \text{inform}, \text{tick}(S, P, T)) : - \\
\text{filter}(S, P), \\
\text{sendMsg}(\text{SID2}, \text{esb}, "epa1", \text{inform}, \text{happens}(\text{tick}(S, P), T)). \\
\text{filter}(\text{Symbol}, \text{Price}) : - \\
\text{Price} > 100, \\
\text{Symbol} = "A.*".
\]

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)}

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_10,000_employees)}

rcvMult(SID, stream, “S&P500“, inform,
tick(Name^car:Major_corporation, P^currency:Dollar,
T^time:Timepoint)) :- ... <semantic filter inference> .
Preparation:Filter:Implementations: IBM WODM Decision Server Events

• **Example:** Define a filter to only process events with a purchase value greater than £100

```plaintext
if the value of the purchase is more than 100
then send discount voucher;
```

• **Effect:** Only process purchase events which match the filter, taking the action to send a discount voucher
Preparation:Filter:Implementations: IBM CICS Transaction Server for z/OS application event capture (1)

- **Example:**
  Define a filter to only produce events for banking transactions with a value greater than $5000

- **Effect:** Only produce events which match the filtering predicates
Preparation: Filter: Implementations: IBM CICS Transaction Server for z/OS application event capture (2) Details

```
<eventCaptureSpecification>
  <name>LargeCashDepositCS</name>
  <eventIdentifier>LargeCashDeposit</eventIdentifier>
  <description>Capture details for event when large cash deposit is made</description>
  <filter>
    <contextFilter>...
    </contextFilter>
    <locationFilter filterType="PROGRAM_INIT">
      <programInit>
        <PROGRAM filterValue="CPPEPM22" filterOperator="EQ" keyword="PROGRAM"/>
        <CHANNEL filterValue="" filterOperator="OFF" keyword="CHANNEL"/>
      </programInit>
    </locationFilter>
    <dataFilter>
      <filterItem>
        <dataFilter filterValue="5000" filterOperator="GT" languageStructureFilename="" languageStructureName="" languageVariableName="" dataPrecision="0" dataType="PACKED" length="8" offset="5" container="APP&PX22" source="CHANNEL"/>
      </filterItem>
    </dataFilter>
  </filter>
  <dataCapture>...
  </dataCapture>
</eventCaptureSpecification>
```
Preparation:Filter:Implementations: IBM InfoSphere Streams (SPL)

**Example:**

```cpp
stream<rstring name, uint32 age> People = PrepFil_Source() {};

stream<rstring name, uint32 age> YoungPeople = Filter(People) {
    param filter : age < 30u;
}
```
**RA: Preparation: Monitoring:**

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

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

- E.g. Selecting a particular channel and event selector in some middleware subscription
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
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
RA: Analysis: Transforms

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

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.”

• **EPIA:**
  – Transformation (EPA): an EPA that includes a derivation step and optionally also a filtering step.
  • Derived event: an event generated as a result of EP in EPA
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

- TIBCO, Prova, Oracle, and IBM implementations for this pattern

- **General pattern:**
  
  \[
  \text{event} \\
  \text{if} \\
  \text{event-matches-selection} \\
  \text{then} \\
  \text{apply-derivation-function for event}
  \]

- **Effect:** (usually identified, filtered) events have some transformation applied to their data, used to update some event object (or create a new event)
Analysis: Transforms: Implementations: Prova: Rules

• **General pattern:**

  \[ \text{rcvMsg} / \text{rcvMult} \ <\text{Event Msg Pattern(s)}> : - \\
  \ <\text{transformation function (Output, Input)}, \\
  \ ... \\
  \ <\text{transformation rule 1}>(\text{Output, Input}) : - \\
  \ <\text{transformation logic} > \]

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

- **General pattern:**
  
  ```
  SELECT f(<property>) as <derived>
  FROM <SOURCE>
  WHERE <predicate-condition>
  ```

- **Effect:** transforms selected properties into new properties, applying function if needed.
Analysis: Transforms: Implementations: IBM WODM Decision Server Events

**Example:** Use Javascript to transform an event e.g. transform order total from number of units to overall price in euros.

Transforming in the event runtime simplifies coding in the connectors.
Analysis:Transforms:Implementations:
IBM CICS Transaction Server for z/OS

• **Example:** Define an EP adapter for a CICS Event Binding (i.e. group of related events) which will transform events into a specified format.
Analysis:Transforms:Implementations: IBM InfoSphere Streams (SPL)

• Example:

```c
stream<rstring sym, float32 price, float32 vol> PreVwaps=Aggregate(Trades) {
    /* (see Detection:Aggregation) */
}

/* VWAP = volume-weighted average price */
stream<rstring sym, float32 vwap> Vwaps = Functor(PreVwaps) {
    output Vwaps : vwap = priceVol / vol;
}
```

![Diagram of event stream, Functor, and transformed event stream with trades weighted by volume]

- Trades
- Weighting transformation
- Trades with price transformed to volume-weighted average price
RA: Analysis: Tracking

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

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
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
RA: Analysis: Rating

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

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”.

• EPIA:
  – Transformation (EPA): an EPA that includes a derivation step and optionally also a filtering step.
    • Derived event: an event generated as a result of EP in EPA
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: Rating: Implementations

- TIBCO, Prova, Oracle, IBM implementations for this pattern

- **General pattern:**
  
  `<declare> event`  
  `<if>`  
  `event-matches-entity-selection`  
  `<then>`  
  `rating = rating-model(event)`

- **Effect:** events’ criteria (and possibly enrichment data) are applied to the rating function to create a rating classification or ordering

- **Note:** rating typically applied in a separate rule (if rating = “First” then …)
Analysis: Scoring: Implementations: Prova: Rules

- **General pattern:** see scoring
- **Effect:** rate events and select rated events
% 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)

- **General pattern:**
  
  ```sql
  SELECT rate@model(<property>)
  FROM <SOURCE>
  WHERE <predicate-condition>
  ```

- **Effect:** rates properties of events using model.
Analysis:Rating:Implementations: IBM WODM Decision Server Events

• **General Pattern:** Rating in WODM is typically carried out by the Decision Server Rules component – the output events indicate something interesting happened, and then business rules rate the type of response depending on details in the events.
Analysis: Rating: Implementations: IBM InfoSphere Streams (SPL)

• Example:

```cpp
typedef Bracket = enum { YOUNG, OLD, OTHER };

stream<rstring name, int32 age> Folks = /*...*/

stream<rstring name, Bracket bracket> Rated = Functor(Folks) {
}
```

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

---

Event stream
Folks

Functor
Rating functor to rate by age

Event stream
Events rated as YOUNG, OLD or OTHER
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
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: Discussion

- Are consolidation, composition and aggregation sufficiently defined or different?

- Original EPTS RA definitions
  - Consolidation: combining disparate events together into a "main" or "primary" event.
  - Composition: composing new, complex events from existing, possibly source, events
  - Aggregation: combining events to provide new or useful information, such as trend information and event statistics.
RA:Detection:Discussion(2)

**EPTS Glossary (v2) Terms**

- 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.
- Composite event
  - A (type of) 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. A composite event always includes the member (base)events from which it is derived.
  - A derived event is not a composite if its method of derivation lies outside a specified set of allowed constructors.
RA:Detection:Discussion(3)

• Dictionary definitions
  – Consolidation: bring together (separate parts) into a single or unified whole
  – Composition: combining parts or elements to form a whole
  – Aggregation: a group or mass of distinct or varied things

• From these we can infer: composition is a generic term, consolidation joins events of a similar or related type, and aggregation joins events of differing types
RA:Detection:Discussion(4)

• Other interesting pattern classifications?
  – Composition from an ordered or partially ordered sequence of events
  – Composition from missing events or non-events
  – Composition from out of order events in some sequence of events

• Other types of complex event detection?
  – Situations or States, indicating some prior sequence of events, significant to be classified or specified
RA:Detection:Discussion(5)

• **Summary**
  
  – Original EPTS RA classification can be safely revised / extended …
  
  – Here we look at
    Consolidation and Aggregation (i.e. composition of similar and different event types)
    + Composition of events including a non-event
RA:Detection:Discussion(6)

• Possible new EP RA definitions:
  – **Consolidation**: reinforcing a primary event with evidence from new events of the same type
  – **Composition**: composing a new event from existing events of similar types + all contributing events are included
  – **Aggregation**: combining event data from disparate event types to create a new event
Consolidation: combine events of similar type together into an 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

• EPIA:
  – Aggregate (EPA): a transformation EPA that takes as input a collection of events and creates a single derived event by applying a function over the input 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>.

Complex Event Detection
Consolidation, Composition, Aggregation
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

- TIBCO, Prova, Oracle, IBM implementations for this pattern
Detection: Consolidation: Implementations: TIBCO BusinessEvents: Rules

• **General pattern:**
  `<declare> event, consolidated-event-object</declare>`

  `<if>`
  `event-supports-event-object`

  `<then>`
  `update-consolidated-object with event-data`

• **Effect:** construct evidence for (main event) object for n incoming events
Detection:Consolidation:Implementations: Prova: Consolidation

• General pattern:

\[
\text{<rcvMult>} \quad \text{Event} :- \\
\quad \text{DerivedEvent} = \text{<consolidate>} ( \\
\quad \text{<compose/aggregate>} \quad \text{Events} ),
\]

\[
\text{<initiate} / \text{terminate}> \ ( \\
\quad \text{DerivedEvent, Situation}).
\]

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

\[ I: T_{interval} \times Fl \rightarrow \{true, false\} \]

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

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

\[ \text{initiates}(D, \text{situation1}, T). \]
\[ \text{holdsAt}(\text{situation1}, t5)? \Rightarrow \text{yes} \]
Dection: Consolidation: Implementations: Oracle EP: pattern matching

• **General pattern:**

```
SELECT <property>
FROM <SOURCE>
MATCH_RECOGNIZE (  
  MEASURES <property>
  PATTERN (<regular-expression>)
  DEFINE <expression>
)
```

• **Effect:** consolidates events from source into measures.
**Example:**

```sql
SELECT M.goingUpPrice
FROM marketFeed
MATCH_RECOGNIZE (  
    MEASURES B.price as goingUpPrice  
    PATTERN (A B)  
    DEFINEA as price < 29.0,B as price > 30.0  
) as M
```

**Description:** consolidates stock ticks into a single ‘going up’ alert event.
Detection: Consolidation: Implementations: IBM WODM Decision Server Events

- **Example**: LargeTransaction events generated for any type of large banking transaction (deposit, withdrawal, transfer)

- RecentSignificantAccountActivity is consolidation of several recent LargeTransaction events, each such event further reinforces the consolidated event

  ![Filter Example](image)

  - Not: this screenshot is from an older version of the product

- **Effect of the example**: Detect customers with recent significant account activity

  ![Diagram](image)
Detection: Consolidation: Implementations: IBM InfoSphere Streams (SPL)

type Trade = tuple<rstring sym, int32 price, int32 vol>;
type Quote = tuple<rstring sym, int32 offer>;
type TradeOrQuote =
    tuple<rstring sym, int32 price, int32 vol, enum{TRADE, QUOTE} kind>;

/*...*/

stream<Trade> Trades = /*...*/

stream<Quote> Quotes = /*...*/

stream<TradeOrQuote> TQs = Custom(Trades; Quotes) {
    logic
    onTuple Trades:
        submit({sym=sym, price=price, vol=vol, kind=TRADE}, TQs);
    onTuple Quotes:
        submit({sym=sym, price=offer, vol=-1, kind=QUOTE}, TQs);
}
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: Aggregation

Complex Event Detection
Consolidation, Composition, 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"

• EPIA:
  – Aggregate (EPA): a transformation EPA that takes as input a collection of events and creates a single derived event by applying a function over the input events.
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

- TIBCO, Oracle, Prova and IBM implementations for this pattern

- **General pattern:**
  `<declare> event, aggregation-object`

  `<if>
  event-member-of-aggregation`

  `<then>
  update-aggregation-object with event-data`

- **Effect:** construct aggregation (event) object for each incoming event

• **General pattern template:**

```
SELECT <summary-property>
FROM <STREAM>[WIN]
WHERE <predicate>
GROUP BY <aggregation-identity>
```

• **Effect:** summarizes the properties of several simple event into a new complex event

- **Scenario:**
  - Output the average bid and ask price of a stock in the last 10 seconds.

- **Application Model**
SELECT symbol, AVG(bid), AVG(ask)
FROM
    StockTickStream [RANGE 10 SECONDS]
GROUP BY
    symbol
Detection:Aggregation:Implementations: Oracle
EP: table metaphor

<table>
<thead>
<tr>
<th>Time</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1s</td>
<td>{“AAA”, 10.0, 12.0}</td>
<td>{“AAA”, 10.0, 12.0}</td>
</tr>
<tr>
<td>4s</td>
<td>{“AAA”, 12.0, 14.0}</td>
<td>{“AAA”, 11.0, 13.0}</td>
</tr>
<tr>
<td>9s</td>
<td>{“BBB”, 4.0, 5.0}</td>
<td>{“AAA”, 11.0, 13}, {“BBB”, 4.0, 5.0}</td>
</tr>
<tr>
<td>15s</td>
<td>{“BBB”, 8.0, 10.0}</td>
<td>{“BBB”, 6.0, 7.5}</td>
</tr>
<tr>
<td>20s</td>
<td></td>
<td>{“BBB”, 8.0, 10.0}</td>
</tr>
</tbody>
</table>

- General pattern template:

  \[ \text{<rule\_head>} \quad :- \\
  \text{<Create Aggregator>}, \\
  \text{@group(<reaction\_group>)} \\
  \text{@timer\_size(Start, Interval, Aggregator)} \\
  \text{rcvMsg <EventPattern> [<Aggregate\_Operation>] } \]

  \[ \text{<rule\_head>} \quad :- \\
  \text{@or(<reaction\_group>)} \\
  \text{rcvMsg <Aggregator>,} \\
  \text{... <consume\_Aggregator>}. \]

- 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
    rcvMsg(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) rcvMsg(XID,self,From,or,[Counter]),
    ... <consume the Counter aggregation object>.
```
Detection:Aggregation:Implementations: IBM WODM Decision Server Events

- **General Pattern**: Share data across multiple events by appending the data to be aggregated into a business object array... then when processing the events, use relevant aggregator.

  Add to array

  Business Object Array

  Operation on aggregate

- **Effect of the example**: sum up the value of a customer’s purchases over a week.

  ```
  if the sum of the values of the purchases is more than 100
  then send discount voucher ;
  ```
Detection:Aggregation:Implementations: IBM InfoSphere Streams (SPL)

• Example:

```plaintext
type Trade = rstring sym, timestamp ts, decimal64 price, decimal64 vol;
/* define PreVwap to subsume attributes of Trade */
type PreVwap = Trade, tuple<decimal64 priceVol>;

/* ... */

/* VWAP = volume-weighted average price */
stream<PreVwap> PreVwaps = Aggregate(Trades) {
  window Trades : sliding, delta(ts, 60.0), count(1), partitioned;
  param partitionBy : sym;
  output PreVwaps : priceVol = Sum(price*vol), vol = Sum(vol);
}
```

---

**Diagram:**

- **Aggregation:**
  - **Trades:**
  - **Sliding Window:**
  - **Aggregate:**
  - **Stream:**
    - **Summed trades:**
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
RA: Reaction: Routing:

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

**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."

- **EPIA:** no direct analogy, possibly maps to split / compose / project
  - Project (EPA): a translate EPA that takes an input event and creates a single derived event containing a subset of the attributes of the input event.
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>.

\[
\text{dest} = f(\text{event} + \text{data})
\]
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

- TIBCO, Oracle, Prova and IBM implementations for this pattern
Reaction:Routing:Implementations: TIBCO BusinessEvents: Rules

• **General pattern:**
  <declare> event, routing-data

  <if>
  event-matches-routing-data

  <then>
  create-new-routed-event and send-new-routed-event

• **Effect:** match events to routing rule and construct / send new event

- **General pattern template:**
  - Route 1: `<filtering pattern>` query
  - Route 2: `<filtering pattern>` query
  - Route n: `<filtering pattern>` query

- **Effect:** Different predicates associated to different queries evaluate and select appropriate destinations

• Scenario:
  – Output stocks whose symbols start with ‘a-m’ to destination 1, and whose symbols start with ‘n-z’ to destination 2.

• Application Model

Destination-1:
SELECT *
FROM StockTickStream
WHERE symbol.matches("^[a-m]")

Destination-2:
SELECT *
FROM StockTickStream
WHERE symbol.matches("^[n-z]")
**Reaction: Routing: Implementations: Prova: rules**

- **General pattern template:**

  \[
  \text{rcvMsg} \ <\text{Event Msg Pattern}> : -
  \text{<routing decisions>},
  \text{sendMsg} \ <\text{route event}>.
  \]

  \[
  \text{<routing decision rule 1> : -}
  \text{<decision conditions>},
  \]

  \[
  \text{<routing decision rule 2> : -}
  \text{<decision conditions>},
  \]

  ...

- **Effect:** Events are routed according to the routing decision rules
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, order(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 has a value below ...
regular(Product) :- price(Product, Value), Value < 10000.
Prova: Event Routing in Event-Driven Workflow

```prolog
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, _).
```
Reaction:Routing:Implementations: IBM WODM Decision Server Events

• Example: Define an event rule which will send the request to supplier A when the purchase indicates supply from Supplier A

if the supplier of the purchase is "Supplier A" then provision from Supplier A;

• Effect of example: Events are routed as required
Reaction:Routing:Implementations: IBM CICS Transaction Server for z/OS

**Example:** Define a EP adapter for a CICS Event Binding (i.e. group of related events) which will route (and format) the event for delivery to a consumer

**Effect of example:** Events routed as required
Reaction:Routing:Implementations: IBM InfoSphere Streams (SPL)

• Example:

```cpp
type Bracket = enum { YOUNG, OLD, OTHER };

/*...*/

stream<rstring name, int32 age> Folks = /*...*/

stream<rstring name, Bracket bracket> Rated = /*...*/

(stream<Rated> Young; stream<Rated> Old) = Custom(Rated) {
  logic
  onTuple Rated: {
    if (bracket == YOUNG)
      submit(Rated, Young);
    else if (bracket == OLD)
      submit(Rated, Old);
  }
}
```

• Effect: Route based on age rating
RA: Reaction: Prediction:

Prediction: an output of some event is the prediction that a new event will occur (or will have occurred)

- E.g. based on some sequence of sensor events, an earthquake event is predicted in location L and within time period P
Reaction: Prediction: Classification

• **Alternative Names:**
  – Event Anticipation; subtype of Event Assessment

• **EPTS Reference Architecture:**
  – *Event Prediction is where the reaction to some event processing is that some new event is predicted to occur.*
    • Example, a complex event detection may predict some future event that should then be assessed and processed further.

• **EPIA:** no direct analogy, possibly maps to split / compose / project
  – *Project (EPA): a translate EPA that takes an input event and creates a single derived event containing a subset of the attributes of the input event.*
Reaction: Prediction: Description

• **Role / Explanation**
  – Prediction is used to describe the process of evaluating the state of a set of events to predict some future event.
  – For example: a new purchase event of an airline ticket may cause a prediction that a business trip event will occur.

• **Associated Business Rule Specification**
  – a constraint (to be enforced during event processing): *The* `<events>` *that satisfies* `<event relationships>` *causes a prediction of* `<anticipated event>`

\[ \text{Prediction} = f(\text{event} + \text{data}) \]

**Diagram:**
- Event
- Prediction = f(event + data)
- Predicted event
- Other data or events
EPTS Glossary comparison

- Virtual event: An event that does not happen in the physical world but is imagined, modeled or simulated.
  - Example: Events predicted by a weather simulation
  - Note: A virtual event can refer to either an event object or a thing that happens.
Reaction:Prediction:Implementations

- TIBCO, IBM, Oracle implementations for this pattern (Prova – Prova Abductive and Inductive Logic Programming or Java API Call to external functionality)

\[
\text{Prediction} = f(\text{event} + \text{data})
\]

Other data or events

\begin{align*}
\text{event} & \quad \text{Prediction} = f(\text{event} + \text{data}) \\
& \quad \text{predicted event}
\end{align*}

• **General pattern:**
  <declare> event, prediction-data

  <if>
  event-matches-prediction-data

  <then>
  create <predicted event object>

• **Effect:** match event to prediction criteria and create predicted event or object
Reaction:Prediction:Implementations: TIBCO BusinessEvents: State Model

• **General pattern:**
  
  `<on> event`  
  `<condition>`  
  `event-matches-prediction-data`  
  
  `<then change state to>`  
  `predicted-event-state`

• **Effect:** match event to prediction criteria and change to predicted state

• Note that there may be a set of prior states that form part of the state model for predicting events

• **Effect:** integration with ODM (Oracle Data Mining) to import model and do predictive score in an online fashion.

```sql
SELECT price
FROM ExchangeStream [NOW]
WHERE score(model, price) > 0.8
```
Reaction:Prediction:Implementations: IBM WODM Decision Server Events

• **General Pattern:** Prediction in WODM is provided by the Decision Server Rules component – the Rule Designer API provides classes and interfaces for a BRL prediction engine
**Example:**

```c
use com.ibm.streams.mining.scoring::*;

type Person = tuple<rstring clientId, int32 age, rstring gender>;
type Predict = Person,
    tuple<float64 predictedVal, float64 predictedStdDev>;

/*...*/

stream<Person> People = /*...*/

stream<Predict> Predictions = Regression(People) {
    param
        //PMML (predictive model markup language) generated e.g. by SPSS
        model    : "../models/linreg.pmml";
        clientId : "CLIENT_ID";
        age      : "AGE";
        gender   : "GENDER";
}
```

**Effect:** Regression operator calculates predicted values and standard deviation for each tuple in the input stream of People, and assigns to the output stream (giving predictions about age, gender)
Discovery: in event processing some new type, schema, state or classification of events is discovered

- An event pattern instance may be detected, where as an event pattern class is discovered
- E.g. a customer query event may indicate a combination of prior transaction events are a new discovered potential fraud event
Learning: uses new event information to refine the knowledge of event patterns, for example using statistics

- E.g. a neural network for fraud detection uses a learning mode using historic events versus results in order to train a new neural net for predicting possible fraud events
Agenda

• Non-RA EP Patterns
  – Other Patterns
  – Use Cases using Patterns
Other: Application-Time

• **Problem:**
  – Need to use application’s view of time, instead of CPU wall-clock.
  – This is particularly useful when events originate from different machines and need some way of synchronizing.

• **Scenario:**
  – Again, consider bid and ask stream…
  – However, bid and ask requests are time-stamped on trader at the time that the order is placed.
Other: Application-Time

• **Scenario:**
  
  – Seller places two ask requests respectively at time 8:00:00 and 8:00:12
  
  – Buyer places one bid request at time 8:00:11
Other: Application-Time

• **Scenario:**
  
  – Remember that we want to correlate using a 10 seconds window, as anything older is stale…
  
  – Hence first event from seller should not be considered, and we should correlate the ask price of 11.0 with the bid price of 9.5
Other: Application-Time

• **Scenario:**
  
  – However, the reality is that the first two events could arrive together in a burst in the exchange, and the third could be delayed in the cloud…
Other: Application - Time

- Scenario:
  - In this case, bid price of 9.5 would correlate to ask price of 10.0 and the exchange would lose money as the spread is lower…
Other: Application-Time: Implementations: Oracle EP

• **Solution:**
  
  – The query does not change…
  
  – However, STREAMS must be configured to use application time-stamps based upon some event property, instead of having events being system-timestamped…

![Diagram showing the integration of StockFeedAdapters, StockTickStream, CQL-processor, and OutputStream.](image-url)
**Example:** Define a correlation expression

\[
(orderPlacedEvent/dynamic/tns:Item_Ordered = itemID) \text{ and } (\text{fn:year-from-dateTime}(orderPlacedEvent/static/cics:context-info/cics:timestamp) = \text{Year}) \text{ and } (\text{fn:month-from-dateTime}(orderPlacedEvent/static/cics:context-info/cics:timestamp) = \text{Month})
\]

**Effect of example:**
The duration of a customer order can be obtained from the timestamps on start and end events

<table>
<thead>
<tr>
<th>orderID</th>
<th>itemID</th>
<th>customer name</th>
<th>item price</th>
<th>item quantity</th>
<th>order duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>+000012784</td>
<td>0001</td>
<td>Steve</td>
<td>20</td>
<td>20</td>
<td>24 m, 19.121 s</td>
</tr>
<tr>
<td>+000012785</td>
<td>0002</td>
<td>Lijia</td>
<td>5</td>
<td>40</td>
<td>23 m, 25.009 s</td>
</tr>
</tbody>
</table>
Other: Application-Time: Implementations: IBM WODM Decision Server Events

- **Example:** Events in WODM carry both the time the event occurred (a timestamp in the event payload) and the time when the event was received into Decision Server Events.

- The example shows event XML received by Decision Server Events (in one of the supported XML formats) with the application-time time stamp:

```xml
<?xml version="1.0" standalone="yes"?>
<connector name="OrderingPatterns" version="2.2">
    <connector-bundle name="About_to_place_order" type="Event">
        ...
    </connector-bundle>
    <system>GBIBMIYACICSAOR1</system>
    <timestamp>2012-04-19T15:41:12+00:00</timestamp>
</connector>
```
Other: Application-Time: Implementations: Prova: rules

• **General pattern template:**

\[
\text{rcvMsg} <\text{Event Msg Pattern}(s)> : - \\
\langle \text{add application timestamp to event} \rangle, \\
\text{sendMsg} <\text{Adjusted Event} >. \\
\]

• **Effect:** Any timestamp can be adjusted (for time computations)
Other: Application-Time: Implementations: TIBCO BusinessEvents: PreProcessor Pattern

- **General Pattern**: adjust Event metadata (e.g. Timestamp) in preprocessor for that event
  - events are generally immutable!
  - PreProcessor applies regardless of engine used!

- **Effect**: Any custom timestamp can be adjusted (for time computations)

**Note**: If the problem is “out of order” events (rather than synchronizing clocks) a different pattern may be better!
Agenda

• EP Patterns in Use Cases
IBM CICS and WebSphere Operational Decision Management Use Case

1. Customer Deposit and Withdrawal Transactions from CICS
2. Process Customer Deposit and Withdrawal Events and identify event patterns
3. Collect Customer’s profile
4. Determine appropriate customer promotion
5. Customer/RM SMS
6. Measure success of promotions

IBM CICS and WebSphere Operational Decision Management Use Case
Oracle Use-case: Capital Markets

• Low-Latency Trading
  – Algorithm Trading

• Trade Matching
  – High throughput
  – In-flight fraud

• Position Capture and Aggregation

• Risk and Analytics
  – Pre/post trade analytics
  – On-demand risk management
Oracle Use-case: Telemetry and Geo-fencing

• **Spatial extension in CQL**
  – Geometry objects (point, polygon)
  – Operations: within, inside, touches, etc.

• **Allows queries such as:**
  – Is streaming location of vehicle within pre-defined areas?
  – Is streaming customer close to shop?
TIBCO Use Case: Application Service Gateway

Telco high performance service policies using event processing

Mobile Services difficult to monitor end to end

Auto detect and fixing of issues for better user service

Excellent service quality needed to avoid customer churn

Service quality affects take-up of new services
Service Gateway architecture exploits event processing

- Standardized channel for monitoring subscribers
- Standardized access to services (e.g. Facebook, Google, etc)
- Consolidation of existing multiple gateways
- Allows analysis of service usage, subscriber behavior, service performance, marketing campaign success, etc
- Creates events for revenue generation and reconciliation
- Handles Common Telco Notifications - change-of-device, contract, service- allowing UpSell/Cross sell etc
Service Gateway architecture exploits event processing

- Standardized channel for monitoring subscribers
- Standardized access to services (e.g., Facebook, Google, etc)
- Consolidation of existing multiple gateways
- Allows analysis of service usage, subscriber behavior, service performance, marketing campaign success, etc
- Creates events for revenue generation and reconciliation
- Handles Common Telco Notifications - change-of-device, contract, service - allowing UpSell/Cross sell etc

**Patterns**

- Security Pattern
- Decision / Policy Pattern
- Routing Pattern
- Transformation Patterns
TIBCO Use Case: Service Performance Manager

SOA metrics generation using event processing
Agenda

• EP Pattern Classifications
• EP Patterns @EPIA
Categorization of Event Processing Patterns

Source: Paschke, A.: Design Patterns for Complex Event Processing, DEBS'08, Rome, Italy, 2008
http://arxiv.org/abs/0806.1100v1
Categorization of Patterns

• Categorization according to Good and Bad Solutions
  – CEP Patterns
  – CEP Anti-Patterns

• Categorization according to the Abstraction Level
  – Guidelines and Best Practices
  – Management patterns
  – Architecture patterns
  – Design patterns
  – Mapping patterns
  – Idioms / Realization patterns
  – Smells / Refactoring patterns

• Categorization according to the Intended Goal
  – Adoption patterns
  – Business patterns
  – Integration patterns
  – Composite patterns:
  – …

• Categorization according to the Management Level
  – Strategic patterns
  – Tactical patterns
  – Operational patterns

Pattern categorization from ‘Event Processing in Action’

1. **Filtering**
   - stateless or stateful

2. **Transformation**
   - Project
   - Translate
   - Enrich
   - Split
   - Aggregate
   - Compose

3. **Pattern Detection - the types of pattern that can be detected**
   - Basic patterns
     - Logical Operator patterns
       - ALL
       - ANY
       - ABSENCE
     - Threshold patterns
       - COUNT
       - VALUE MAX
       - VALUE MIN
       - VALUE AVERAGE
       - FUNCTOR
     - Subset selection patterns
       - RELATIVE n HIGHEST VALUES
       - RELATIVE n LOWEST VALUES
   - Modal patterns
     - ALWAYS
     - SOMETIMES
   - Dimensional patterns
     - Temporal order patterns
       - SEQUENCE
       - FIRST n
       - LAST n
     - Temporal Trend patterns
       - INCREASING
       - DECREASING
       - STABLE
       - NON-INCREASING
       - NON-DECREASING
       - MIXED
     - Spatial patterns
       - MAX DISTANCE
       - AVERAGE DISTANCE
       - RELATIVE MIN DISTANCE
       - RELATIVE MAX DISTANCE
       - RELATIVE AVERAGE DISTANCE
     - Spatiotemporal patterns
       - MOVING IN A CONSTANT DIRECTION
       - MOVING IN A MIXED DIRECTION
       - STATIONARY
       - MOVING TOWARD
Summary

• Functional Patterns v1.0 are “complete”

• Next: further analysis and refinement terminology
  + more examples
  + drill-down into operation level
References

• EPTS Glossary

• Event Processing in Action
  – Opher Etzion and Peter Niblett, Manning Publications Co. (2011)