Event Processing Architectures
leading to an EPTS Reference Architecture

Courtesy of:
Adrian Paschke (Freie Universitaet Berlin)
Paul Vincent (TIBCO Software)
Catherine Moxey (IBM)
Alex Alves (Oracle)
Themeis Palpanas (University of Trento)
Tutorial

- Event Processing is an increasingly important area in the field of IT
- Event Processing Architectures have evolved to handle the needs of low-latency / high-throughput event processing
- Event Processing Architecture diagrams are used to describe the functions and component layouts of event processing systems
- Various providers and suppliers use their own architectural descriptions, and EPTS has collated and refined these into a candidate “reference” architecture
Agenda

- Introduction to architectures, architecture methodologies, and event processing
- Member architectures and salient features
- Skeleton reference architecture from EPTS Reference Architecture Working Group
- Summary and future work of the EPTS Reference Architecture Working Group
Agenda

• Introduction to architectures, architecture methodologies, and event processing

• Member architectures and salient features

• Skeleton reference architecture from EPTS Reference Architecture Working Group

• Summary and future work of the EPTS Reference Architecture Working Group
Introduction to Event Processing

• **Event-centric view of IT**

• **Events are**
  - Sent and Received
  - Aggregated, Transformed into Data, Deleted
  - Processed in queries, rules etc
  - Cause actions like processes, service invocations, etc
Introduction to Event Processing

Event Processing, Complex Event Processing, Event Stream Processing

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

Events

Derived Events

Event Producer
(Event Source, Event Emitter)

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

Design time

Run time

Administration
About the EPTS Reference Architecture 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 *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

• **Current work is focused on**
  – Discovery of *existing Event Processing Architectures*
    • collected RAs from e.g. IBM, Oracle, Tibco, Streambase, Aleri, Microsoft…
  – Definition of *Terminology and Methodology* for comparing and describing Event Processing Reference Architectures
Relationship with the other EPTS Groups / Areas

• Application and Analysis
  – EPTS Use Case WG

• Terminology
  – EPTS Glossary WG

• Value and ROI
  – EPTS Business Value WG

• Architecture & Metamodels
  – EPTS Reference Architecture WG

• Features and Components
  – EPTS Language WG
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.
Motivation and Benefits

• Motivation
  – Event Processing is evolving from many existing technologies and creating or adapting different tools
  – Potential adopters (stakeholders) may have problems understanding and adequately defining EP-based architectures and solutions.

• Benefits
  – a Reference Architecture aids efficient Event Processing solution development, by predefining customizable abstract frames of reference for specific stakeholder concerns and application domains.
    • aids in reusability of successful EP architectures for frequently occurring EP design problems
    • enables easier comparison of proposed EP solutions
  – Underlying Reference Model defines / explains the terminology and components in Event Processing architectures
• **Recommended Practice for Architectural Description of Software-intensive Systems**
  
  – Now an **ISO/IEC 42010:2007 standard**
  
  – Includes 6 elements
    1. Architectural description
    2. System **stakeholders** and their concerns
    3. One or more architectural **views**
    4. **Viewpoints**
    5. A **record of** all known **inconsistencies** among the architectural description’s required constituents
    6. A **rationale for selection** of the architecture
ISO/IEC 42010:2007 Terminology (1)

• Architecture
  – The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

• Architectural Description
  – A collection of products that document the architecture.

• System
  – A collection of components organized to accomplish a specific function or set of functions.

• System Stakeholder
  – A system stakeholder is an individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.
ISO/IEC 42010:2007 Terminology (2)

- **View**
  - A representation of the whole system from the **perspective** of a related set of concerns.

- **Viewpoint**
  - A specification of the **conventions for constructing and using a view** - a pattern or template which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

- **Model**
  - A view may consist of one or more **models and a model may participate in one or more views**.
  - Each model is defined according to the **methods established in the corresponding viewpoint definition**.
Declaring a Viewpoint

- **Each viewpoint is specified by:**
  - Viewpoint name
  - The stakeholders addressed by the viewpoint
  - The stakeholder concerns to be addressed by the viewpoint
  - The viewpoint language, modeling techniques, or analytical methods used
  - The source, if any, of the viewpoint (e.g., author, literature citation)

- **A viewpoint may also include:**
  - Any consistency or completeness checks associated with the underlying method to be applied to models within the view
  - Any evaluation or analysis techniques to be applied to models within the view
  - Any heuristics, patterns, or other guidelines which aid in the synthesis of an associated view or its models
Architecture Views - Examples

• **Domain Architecture**
  domain models represent domain requirement and logic
  • e.g. process representation, use case analysis, free text

• **Application Architecture**
  structural, logic architecture
  – **Functional Layering**, e.g. functional multi-tier structure
  – **Functional Decomposition**, e.g. Component Architecture, UML (class/activity/sequence)

• **System Architecture**
  – Concrete implementation of an architecture in a target platform
Example UML Architectural Views and Diagrams

UML defines 13 diagram models that describe 4+1 architectural views.

4+1 architectural views model by Philippe Kruchten, IBM
### Event Processing Reference Architecture Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint Element</th>
<th>Viewpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concepts</td>
<td>How to implement?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Architects / Engineers</td>
</tr>
<tr>
<td>Concerns</td>
<td>Effective construction and deployment</td>
</tr>
<tr>
<td>Techniques / Languages</td>
<td>Modeling, Engineering</td>
</tr>
</tbody>
</table>
Agenda

- Introduction to architectures, architecture methodologies, and event processing
- **Member architectures and salient features**
- Skeleton reference architecture from EPTS Reference Architecture Working Group
- Summary and future work of the EPTS Reference Architecture Working Group
Overview – Sample Member Architectures

- **University of Trento** - Themis Palpanas
- **TIBCO** – Paul Vincent
- **Oracle** - Alex Alves
- **IBM** - Catherine Moxey
Example Use Case

The fast flower delivery use case

- **Flower Stores**
  - Location
  - Location Service

- **Pick-up**
  - Pick Up Alert
  - GPS

- **Van Driver**
  - Delivery Bid
  - Delivery confirmation and time-out alerts

- **Recipient**
  - Assignments
  - Delivery confirmation

- **Driver's Guild**
- **Monitoring System**
  - Ranking
  - Ranking System

- **EPTS**
  - Event Processing Technical Society

Courtesy of Opher Etzion & Peter Niblett, IBM
Use Case Detail: FFD delivery and bid requests

High level Description

Stakeholders vs Events (high level)

<table>
<thead>
<tr>
<th>Id</th>
<th>Precondition</th>
<th>Event</th>
<th>Stakeholder sending</th>
<th>Stakeholder receiving</th>
<th>Other actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td>Delivery Request</td>
<td>Florist</td>
<td>System</td>
<td>System also enriches event with &quot;minimum ranking&quot; based on florist</td>
</tr>
<tr>
<td>2.</td>
<td>Delivery Request matched to Drivers based on ranking, nearby region</td>
<td>GPS Location</td>
<td>Van</td>
<td>System</td>
<td>System maps GPS data to &quot;city region&quot; for van</td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>Bid Request</td>
<td>System</td>
<td>Driver</td>
<td></td>
</tr>
</tbody>
</table>

Overall Preconditions
None described.

Postconditions
Bid sent out OR Bid not sent out (an exception)
FDD Use Case: Stakeholders, Events, and Details

Detailed Description

Stakeholders vs Events (details)

<table>
<thead>
<tr>
<th>Step</th>
<th>Actor Actions</th>
<th>Requirement for System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Florist places Delivery Request</td>
<td>Accepts Delivery Request</td>
</tr>
<tr>
<td>2.</td>
<td>System accepts Delivery Request</td>
<td>Enriches Delivery Request, creates Bid List from Driver List, filters Bid List, creates Bid Request</td>
</tr>
<tr>
<td>3.</td>
<td>Van sends Location Event</td>
<td>Maps to Van Region Location, updates Driver List</td>
</tr>
</tbody>
</table>

Extensions
- branches from main flow for special conditions
  - NoBidderFound

Variations
- TBA

Frequency
- Estimates per event are:

<table>
<thead>
<tr>
<th>Event</th>
<th>Stakeholder sending</th>
<th>frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery Request</td>
<td>Florist</td>
<td># deliveries per day (max and average) per florist: 30 and 10; # florists: 20</td>
</tr>
<tr>
<td>GPS Location</td>
<td>Van</td>
<td># update events per GPS system per hour: 60; # vans: 20</td>
</tr>
<tr>
<td>Bid Request</td>
<td>System</td>
<td># bid requests ( \approx ) # deliveries per day</td>
</tr>
</tbody>
</table>
Overview – Sample Member Architectures

• **University of Trento** - Themis Palpanas
• TIBCO – Paul Vincent
• Oracle - Alex Alves
• IBM - Catherine Moxey
Entity Name System Service for CEP
Entity Identification

• what is the problem?
  – different pieces of data may refer to the same real world entity
    • entity: person, organization, location, conference, piece of hardware, etc
  – references to entities are often in an event’s payload
Entity Identification

• what is the problem?
  – different pieces of data may refer to the same real world entity
    • entity: person, organization, location, conference, piece of hardware, etc
  – references to entities are often in an event’s payload

• why is this problem important?
  – incorrect identification of entities may lead to
    • erroneous decisions in subsequent processing steps
    • reduced overall performance quality
Entity Identification

• what is the problem?
  – different pieces of data may refer to the same real world entity
    • entity: person, organization, location, conference, piece of hardware, etc
  – references to entities are often in an event’s payload

• why is this problem important?
  – incorrect identification of entities may lead to
    • erroneous decisions in subsequent processing steps
    • reduced overall performance quality

• why is this problem relevant?
  – events originate from several different sources
  – sources may refer to same real-world entity in different ways
  – if entity identification problem not tackled, event filtering/correlation/etc. will produce wrong results
Our solution

• the Entity Name System (see References [1][2])
  – scalable infrastructure for assigning and managing unique global identifiers for named entities

• basic ideas
  – any description of an entity is “resolved” into its global ID
  – not a universal knowledge base about entities
  • stores minimal amount of information about an entity for
    – distinguishing entities one from another
    – finding entities and their identifiers as a result of a query
  – similar to lightweight master data management system
Entity Name System (ENS)
Entity Name System (ENS)
Entity Name System (ENS)
Entity Name System (ENS): Reference Architecture

Event Processing System

ENS node

Access Services

Matching Component

Storage Component

entity repository

e1 id

e1

event source A

... e1 ...

event source B

... e1 ...

... event source C

32
Entity Name System Service

- Event consumer
- Event modeler
- Event processing medium
- Event-entity identification
- Event originator
- ENS node
Entity Name System Service: Use Case

correct entity identification: single driver dispatched for both events

customers X and Y are the same addresses

delivery location A(=B)

event entities from different data sources
Overview – Sample Member Architectures

- University of Trento - Themis Palpanas
- TIBCO – Paul Vincent
- Oracle - Alex Alves
- IBM - Catherine Moxey
TIBCO Reference Architecture(s)

Architect: Functional (abstract derivative)

End User: Functional (abstract)

Generalized C/EP Architecture

Architect: Component (abstract)

CEP Agent Architecture

Architect: Component (detailed agent-level)
TIBCO Example Architecture

Input Events + Content

Complex / Event Processor:
- Event relationships
- Complex Event patterns and sequences
  - (Optional) Inference Rules
  - (Optional) Business Decisions

Services

Input Data

Output Events (including UI)
TIBCO Example Architecture for FFD

Input Events + Content

Channels: TBA
Events: Delivery Request GPS Location

Complex / Event Processor:
- Event relationships

Operations:
- Enrich Delivery Request
- Create + filter Bid List
- Create Bid Request

Services

Input Data

Data:
- Florist Minimum Ranking
- Driver Ranking
- Van Region Location
- Van-Driver association
- Bid List, Driver List

Output Events (including UI)

Channels: TBA
Events: Bid Request Rejection
TIBCO Example Architecture detail

Input Channel
+ Destination
+ Payload
+ Metadata
+ Timestamp(s)

Events and Data
includes static relationships / hierarchies / ontology

Event Operations
components for detecting patterns via queries, rules (ECA, inference, constraint etc), orchestrations (process, state), with optional reactions and decisions

Complex / Event Processor

Services
+ Interfaces
+ Choreographies required

Data (Services)
+ Interfaces
+ Choreographies required

Event and Data State Local Store
+ Choreographies / locks required

Agent Arrangement + Choreography

Display

Outbound Channel
+ Destination
+ Payload
+ Metadata

Agent
TIBCO Architecture – FFD detail

**Input Channel**
- +Destination
- +Payload
- +Metadata
- +Timestamp(s)

**Event format:** depends on channel and source requirements etc

**TimeToLive:** depends on processing approach used

**Data / Event structures:** depends on operations needed

**Complex / Event Operations**
- components for detecting patterns via queries, rules (ECA, inference, constraint etc), operations (process, state), internal reactions and decisions

**Processing operations:** rules, states etc based on operations needed

**Event and Data State Local Store**

**Data location:** depends on latency, + external sharing requirements

**Services + Interfaces**
- + Choreographies required

**Services + Data** includes state

**Agent Arrangement**
- Agent structure: based on process load / freq requirements

**Display**

**Outbound Channel**
- + Destination
- + Payload
- + Metadata
Example Deployment Architecture

Distributed Event Processing Agents – see ref.
Overview – Sample Member Architectures

- University of Trento - Themis Palpanas
- TIBCO – Paul Vincent
- Oracle - Alex Alves
- IBM - Catherine Moxey
Conceptual View

Event Repository

Event Specification

catalogued in

stored in

described By

Event Processor

Event Channel

Event Consumer

Event

Event Producer

processes

uses

receives

receives through

publishes

sends through

© 2010 Oracle Corporation
EDA is composed of several processing steps intermingled with user logic.

This arrangement or network of event processing components is called an event processing network (EPN).
Fast-Flower-Delivery Use-case
Overview – Sample Member Architectures

• University of Trento - Themis Palpanas
• TIBCO – Paul Vincent
• Oracle - Alex Alves
• IBM - Catherine Moxey
IBM Conceptual Architecture for EP

Event Producers

Event Emitter
- Event Instantiator
- Event Processing Services (Simple)
- Event Adapters

Event Bus
- Publication Services
  - Event Processing Services
  - Event Security Services
  - Event Information Management and Query Services
- Event Channels
- Subscription Services
  - Event Adapters
- Notification Services
- Registries
- Repositories

Event Handler
- Event Adapters
- Event Processing Services (Simple)
- Event Orchestration Services

Event Consumers
**IBM EP Architecture – mapping to methodology**

<table>
<thead>
<tr>
<th>Stakeholders</th>
<th>Concerns</th>
<th>Architectural View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event consumers – use events to get information, take action, detect problems, etc.</td>
<td>Receive correct events in consumable format, with relevant data</td>
<td>Event Handler (adapters, consumer-side EP services, orchestration), Event Processing Services, Repositories, Event Emitters, Publication, Subscription, Notification services to receive events</td>
</tr>
<tr>
<td>Event “Analysts” – create and understand event specifications and definitions, how events should be processed, etc.</td>
<td>Means to specify the events to be produced, the patterns to be identified, event enrichment etc.</td>
<td>Event Instantiator, Event Processing Services, Event Information Management and Query Services, Event Security Services, Event Registries, Event Repositories to do pattern-matching over time</td>
</tr>
<tr>
<td>Event System managers – manage and control the EP system</td>
<td>Security, Performance, Configurability</td>
<td>Event Security Services, Event Governance and related Security Services</td>
</tr>
<tr>
<td>Event System operators – operate the EP system</td>
<td>Abilities to monitor and configure system, diagnose and solve problems</td>
<td>Event Monitoring and Analytic Infrastructure, Event Channels, Publication, Subscription, Notification Services</td>
</tr>
<tr>
<td>Event Producers</td>
<td>Generated events are made use of in some way</td>
<td>Event Instantiator, producer-side EP services, Event Adapters, Event Bus…</td>
</tr>
</tbody>
</table>
IBM Conceptual Architecture for Decision Maker stakeholder

Event Producers – produce useful events
- relevant data

Event Instantiator

Event Processing Services (Simple) – relevant data

Event Adapters

Event Bus

Publication Services

Event Processing Services – valuable events, relevant data

Event Channels

Event Security Services

Event Information Management and Query Services

Subscriptions Services

Notification Services

Event Adapters – consumable format

Event Processing Services (Simple) – valuable events

Event Orchestration Services

Event Consumers – receive valuable business events in consumable format, with relevant data

User Interface – define business events, event patterns, interact with registries & repositories, subscribe, etc.

Event Governance and Related Security Services

Event Monitoring & Analysis Infrastructure

Repositories – valuable events

Registries - Event patterns

Key: Components of interest – Concerns
IBM Conceptual Architecture for Architect stakeholder

**Event Producers**
- produce events when something (of potential interest) happens

**Event Emitter**
- convert, package events

**Event Bus**
- can create reduced volume of more informative events

**Event Handler**
- prepare events for consumption, determine appropriate consumer

**Event Consumers**
- respond to events

**Publication Services**
- Event Processing Services (Simple) – filtering, mediation, enrichment
- Event Security Services – control access & authority
- Event Orchestration Services – event flows, event distribution

**Subscription Services**
- Event Adapters – receive events from event bus, unwrap

**Notification Services**
- Event Adapters – formatting, protocol conversion

**Event Instantiator**
- create event instances, add metadata

**Event Monitoring and Analytics Infrastructure**
- Event Governance and Security Services – manage/control lifecycle of events & metadata
- Repositories – store events
- Registries – taxonomies, ontologies

**Key**
- Component View (select required components)
- Functional View
FFD Use Case mapped to IBM RA

Showing some of the Event Flows and Event Processing Agents

Event Emitters
- A1
- A2

Event Bus
- A3
- A4
- A6
- A7

Event Handler
- Event Processing Services (Simple)
  - Event Instantiator
  - Event Processors
  - Event Security Services
  - Event Information Management and Query Services
  - Repositories
  - Registries
  - Event spec, Event patterns

Event Orchestration Services

Event Consumers
- Flower Store
- Driver
- System Manager
- Drivers’ Guild
- Monitoring System

Event Producers
- Event Instantiator
- Event Processing Services (Simple)

Event Handling

DEBS 2010

Agent ID | Agent Name
--- | ---
A1 | Enrich Delivery Event
A2 | Translate Location (from GPS to region of city)
A3 | Filter Authorized drivers (who meet ranking)
A4(m) | Assign (manual)
A4(a) | Assign (automatic)
A5 | Generate pick-up alert
A6 | Generate Delivery alert
A7 | Adjust Ranking

Agent Properties
- Agent Name
- Agent Type
- Agent Contact
- Input Event(s)
- Output Event(s)
- Agent Specification
- Agent Implementation
Agenda

• Introduction to architectures, architecture methodologies, and event processing

• Member architectures and salient features

• Skeleton reference architecture from EPTS Reference Architecture Working Group

• Summary and future work of the EPTS Reference Architecture Working Group
Step 1: collect Event Processing Reference Architectures…
Step 2: identify methodology to abstract an EPTS version…
Step 3: distil the common EPTS Reference Architecture(s)
## Reference Architecture Viewpoints recap

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Viewpoint</th>
<th>Viewpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Element</strong></td>
<td><strong>Engineering EP Architecture</strong></td>
<td><strong>Managing EP Architecture</strong></td>
</tr>
<tr>
<td>Concepts</td>
<td>How to implement?</td>
<td>How to apply?</td>
</tr>
<tr>
<td>Stakeholders</td>
<td>Architects / Engineers</td>
<td>Project Manager</td>
</tr>
<tr>
<td>Concerns</td>
<td>Effective construction and deployment</td>
<td>Operational Management</td>
</tr>
<tr>
<td>Techniques / Languages</td>
<td>Modeling, Engineering</td>
<td>IT (service/appl) management, project management</td>
</tr>
</tbody>
</table>
Stakeholders and Viewpoints covered

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Component</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Maker / End User</td>
<td>[Not applicable]</td>
<td>Inputs, Outputs and Processing Requirements View</td>
</tr>
<tr>
<td>Architect / Designer</td>
<td>Solution Components View</td>
<td>Functions carried out in CEP View</td>
</tr>
</tbody>
</table>
### Stakeholders and Viewpoints covered

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Component</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Maker / End User</td>
<td>[Not applicable]</td>
<td>Inputs, Outputs and Processing Requirements View</td>
</tr>
<tr>
<td>Architect / Designer</td>
<td>Solution Components View</td>
<td>Functions carried out in CEP View</td>
</tr>
</tbody>
</table>
Logical View

• View of the Event Processing Network?
• A Network Diagram?
• *Currently under consideration!*
Example Logical View

Source: WestGlobal
### Stakeholders and Viewpoints covered

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Component</th>
<th>Functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Maker / End User</td>
<td>[Not applicable]</td>
<td>Inputs, Outputs and Processing Requirements View</td>
</tr>
<tr>
<td>Architect / Designer</td>
<td>Solution Components View</td>
<td>Functions carried out in CEP View</td>
</tr>
</tbody>
</table>
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.
Functional View source: definitions of EP

Event Processing, Complex Event Processing, Event Stream Processing

- Event Abstraction
- Event Pattern Detection
- Event Composition
- etc. etc.

Event Producer
(Event Source, Event Emitter)

Derived Events

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

Design time

Run time

Administration
Reference Architecture: Functional View

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

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

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

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

- **Event Production**
  - Publication, Retrieval

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

- **State Management**

- **Event Actions**
  - Event Correlations and patterns
  - Event Computations
  - Event Selections

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

- **Design time**
- **Run time**
- **Administration**
Reference Architecture: Functional View / Runtime

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

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

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

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

- **Event Production**
  - Publication, Retrieval

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

- **Run time**
Reference Architecture: Functional View / Runtime

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

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

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

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

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

Event Preparation: the process of preparing the event and associated payload and metadata for further stages of event processing.

- **Entity Identification**: incoming events will need to be identified relative to prior events, such as associating events with particular sources or sensors.
- **Event Selection**: particular events may be selected for further analysis. Different parts of event processing may require different selections of events. See also event filtering.
- **Event Filtering**: a stream or list of events may be filtered on some payload or metadata information such that some subset is selected for further processing.
- **Event Monitoring**: particular types of events may be monitored for selection for further processing. This may utilise specific mechanisms external to the event processing such as exploiting event production features.
- **Event Enrichment**: events may be "enriched" through knowledge gained through previous events or data.
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.
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.
Reference Architecture: Functional View / Design time

Covers the definition, modeling, improvement / maintenance of the artifacts 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)
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 architectures, architecture methodologies, and event processing
• Member architectures and salient features
• Skeleton reference architecture from EPTS Reference Architecture Working Group
• Summary and future work of the EPTS Reference Architecture Working Group
Current Works

• Terminology and Methodology
  – ANSI/IEEE Std 1471 :: ISO/IEC 42010 Terminology
  – EPTS-RA Terminology
  – Concepts from the EPTS Glossary
• Reference Architecture Discovery
• Reference Architecture Comparison
  – Identify commonalities in proposed architectures
  – Collect core and additional CEP system functions and components
• Member Reference Architecture Descriptions
  – Using the EPTS-RA Methodology and Terminology
• Application of RA Descriptions to Use Cases
Next Steps

• Generalize the EPTS Reference Architecture descriptions
  – addressing important stakeholders and their views
  – using rigorous RA methodology and terminology (+ glossary)

• Evaluate the EPTS-RA descriptions
  – apply on selected EPTS use cases and compare to member experience
  – compare with Best Practice Guidelines

• Review Logical Architecture requirements for Reference purposes

• Possible Outputs:
  – EPTS-RA Description document
  – EPTS-RA Design Patterns and Best Practice Guidelines document
  – Wiley Book "Pattern Oriented Software Architecture: Architectures, Models and Patterns for Event Processing" (already in preparation)
  – Input for Event Processing Metamodels and associated standards
Summary

• Reference Architecture provides a common set of EP Functions that may be included in
  – EP systems
  – EP-related tools

• Describes the mapping from Glossary of Terms to Implementation

• Provides the basis for EP agents, operators and languages
Thank you!

Acknowledgment to the Event Processing Technical Society
Reference Architecture working group: David Tucker (Event Zero), John Morrell (Coral8, Inc.), Baden Hughes (Event Zero), Dieter Gawlick (Oracle), Alex Alves (Oracle), Anand Srinivasan (Oracle), Shailendra Mishra (Oracle), Zbigniew Jerzak (TU Dresden), Hans-Arno Jacobsen (University Toronto), Albert Mavashev (Nastel Technologies Inc.), Simon Courtenage (University of Westminster), Catherine Moxey (IBM United Kingdom Limited), Richard Tibbetts (Streambase), Brian Connell (WestGlobal), Guy Sharon (IBM), Themis Palpanas (University of Trento)
References

- EPTS Glossary
  http://www.ep-ts.com/component/option,com_docman/task,doc_download/gid,66/Itemid,84/

- University of Trento Architecture section:
  1. Heiko Stoermer, Themis Palpanas, George Giannakopoulos. *The Entity Name System: Enabling the Web of Entities*. International Workshop on Data Engineering meets the Semantic Web (DESWeb), in conjunction with the IEEE International Conference on Data Engineering (ICDE), Long Beach, CA, USA, March 2010
  2. Zoltan Miklos, Nicolas Bonvin, Paolo Bouquet, Michele Catasta, Daniele Cordioli, Peter Fankhauser, Julien Gaugaz, Ekaterini Ioannou, Hristo Koshutanski, Antonio Mana, Claudia Niederee, Themis Palpanas, Heiko Stoermer. *From Web Data to Entities and Back*. International Conference on Advanced Information Systems Engineering (CAiSE), Hammamet, Tunisia, June 2010

- IBM Architecture section:

- TIBCO Architecture section: