Event-driven Process Engines

Background

Matthias Weidlich
Setting

Permits visual modeling of processes including the annotation of model elements with produced and consumed event types.

**TASK: MODELING**
- **Process Modeler**
  - consumes: PackageScanned
  - produces: PackageScanned

**TASK: EXECUTION**
- **Process Execution Engine**
  - events occurring during process execution
  - matched complex events

**Event Type Repository**
- **Simple Event Type**
  - read / write
- **Complex Event Type**
  - matched complex events based on queries over incoming events

**Environment**
- defined by data attributes
- defined by a query to be evaluated over event streams, based on data attributes, time frame, and sequence patterns

Sensors and monitoring devices noticing and producing events.
BPM/BPMN Primer
Process Modelling - The Why
Business Process Management

Goals

... get holistic view on how an organisation works
... understand activities of an organisation and their relations
... understand embedding of activities within an organisational and technical context

Potential for improving the business process
Starting point

- Radical changes work out only under specific conditions
- Re-engineering neglects continuous changes of the environment

BPM Lifecycle

- Continuous evaluation and monitoring of a process
- Incremental improvements

“Business process management includes concepts, methods, and techniques to support the design, administration, configuration, enactment, and analysis of business processes”

[Weske]
BPM Lifecycle and Models

1. Process identification
   - Process architecture
   - Process discovery
     - Conformance and performance insights
     - As-is process model
   - Process analysis
     - Insights on weaknesses and their impact
   - Process redesign
     - To-be process model
   - Process implementation
     - Executable process model
Purposes of Modelling

Large variety of modelling purposes

• Business purposes
• Information systems purposes

Business purposes

• Documentation, guidelines, work instructions
• Process redesign, from as-is to to-be
• Staff planning, often using statistical annotations
• Quality certification
Purposes of Modelling cont.

Information systems purposes

• Enterprise Resource Planning (ERP) system selection
  • ERP systems provide business functionality
  • System selection based on delta-analysis of own processes and implemented process

• Software development
  • Process models as requirement documents

• Process implementation
  • Workflow system supports execution of cases
  • Different degrees of automation of activities
Process-oriented Information System

Process-oriented Information System (POIS)

• "a generic software system that is driven by explicit process representations to coordinate the enactment of business processes"
  [Weske 2007]

Process-orchestration

• "a system acts as a central agent that controls the execution of the process activities, very similar to a conductor centrally controlling the musicians in an orchestra"
BPM Lifecycle and POIS
Beyond System Workflows

Human Interaction Workflows
- User interaction during process execution
- Combination of manual and fully automated activities
- Active control of process by interaction with process participants

Human workflow systems typically also include:
- Modelling and integration of process participants (roles, capabilities)
- Provisioning of specific interfaces (work lists)
Example of a Human Interaction Workflow

- Store Order
- Check Inventory
- Prepare Invoice
- Send Invoice
- Handle Shipment
- Prepare Shipment

- Order Management System
- Inventory Management System
- Office Application
- Archiving System

- Archive

Process Modelling - The How
Process Models

Model Level:
- Take Call
- Customer Entry
- Get Contact Details
- Known Customer?
- File Quote
- Approval needed?
- Submit Quote
- Close Deal
- Approve Quote

Model of:
- Contact from Marketing
- Get Past Counter-Offer
- Schedule Customer Presentation
- Enter Quote Details
- Send Quote
- Pos. Response
- File Contract
- Enter Negotiation Report
- Attach Quote Appendix
- Enter Loss Report
- 2 weeks
- Schedule Call

Abstraction:
- Original
- Model of

© Stiftung Deutsches Technikmuseum Berlin
Mapping Business Processes

What is mapped to a process model?

• Activities
  Building blocks that describe elementary pieces of work
• Routing conditions
  Describe temporal and logical constraints on the execution of activities
• Inputs, Outputs
  Informational or physical artefacts processed by activities
• Events
  How time, messages, exception influence the execution
• Resources
  Persons, organisational units, systems that execute activities
Abstraction Overview

Levels of Abstraction

- Meta-model
  - process \( \rightarrow \) activity
  - process landscape

Levels of Granularity

- Level 1: Models
  - accounting
  - production
  - sales

- Level 2: Cases
  - make pin
  - request credit
  - hire staff
  - book trip

- Level 3: Cases
  - pin made by Adam Smith, 1776
  - pin made by Peter Smith, 1776
  - pin made by John Smith, 1776
  - credit request by Hammer & Champy, 1990
  - credit request by Davenport, 1990
  - credit request by Kettinger, 1997
  - hiring of Taylor, 1911
  - hiring of Ford, 1925
  - hiring of Rosemann, 2003
Process Modelling – How?

Modeling Technique

Modeling Language
Notation  Syntax  Semantics

Modeling Tool
Modeling Method
Business Process Model and Notation (BPMN)
Business Process Model and Notation

BPMN, version 2.0

• Standardised by Object Management Group (OMG)
• Before, version 1.X: Business Process *Modeling* Notation

Very expressive modelling language, mainly for modelling functional view of business processes

• MOF conformant meta-model
• Informal, but rather precise execution semantics
Check that...
How is it defined?

A **Pool** is the graphical representation of a **Participant** in a **Collaboration**. A **Participant** (see page 114) can be a specific **PartnerEntity** (e.g., a company) or can be a more general **PartnerRole** (e.g., a buyer, seller, or manufacturer). A **Pool** **MAY** or **MAY NOT** reference a **Process**. A **Pool** is **NOT REQUIRED** to contain a **Process**, i.e., it can be a “black box.”

- A **Pool** is a square-cornered rectangle that **MUST** be drawn with a solid single line (see Figure 9.2).
  - The label for the **Pool** **MAY** be placed in any location and direction within the **Pool**, but **MUST** be separated from the contents of the **Pool** by a single line.
  - If the **Pool** is a black box (i.e., does not contain a **Process**), then the label for the **Pool** **MAY** be placed anywhere within the **Pool** without a single line separator.
  - One, and only one, **Pool** in a diagram **MAY** be presented without a boundary. If there is more than one **Pool** in the diagram, then the remaining **Pools** **MUST** have a boundary.

The use of text, color, size, and lines for a **Pool** **MUST** follow the rules defined in Section “Use of Text, Color, Size, and Lines in a Diagram” on page 41.

<table>
<thead>
<tr>
<th>Name</th>
</tr>
</thead>
</table>

**Figure 9.2 - A Pool**
### Participant attributes and model associations

<table>
<thead>
<tr>
<th>Attribute Name</th>
<th>Description/Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>name: string [0..1]</td>
<td>Name is a text description of the Participant. The name of the Participant can be displayed directly or it can be substituted by the associated PartnerRole or PartnerEntity. Potentially, both the PartnerEntity name and PartnerRole name can be displayed for the Participant.</td>
</tr>
<tr>
<td>processRef: Process [0..1]</td>
<td>The processRef attribute identifies the Process that the Participant uses in the Collaboration. The Process will be displayed within the Participant’s Pool.</td>
</tr>
<tr>
<td>partnerRoleRef: PartnerRole [0..*]</td>
<td>The partnerRoleRef attribute identifies a PartnerRole that the Participant plays in the Collaboration. Both a PartnerRole and a PartnerEntity MAY be defined for the Participant. This attribute is derived from the participantRefs of PartnerRole.</td>
</tr>
<tr>
<td>partnerEntityRef: PartnerEntity [0..*]</td>
<td>The partnerEntityRef attribute identifies a PartnerEntity that the Participant plays in the Collaboration. Both a PartnerRole and a PartnerEntity MAY be defined for the Participant. This attribute is derived from the participantRefs of PartnerEntity.</td>
</tr>
</tbody>
</table>
Attributes

Attributes enrich the graphical representation
• Only some attributes are represented graphically
• Hence, graphical representation is not complete

Attributes of Business Process Diagrams
• Technical details (id, name, version, author, language)
• Reference to expression language
And semantics?

![Diagram](image)

**Figure 13.3 - Merging and Branching Sequence Flows for a Parallel Gateway**

On the one hand, the **Parallel Gateway** is used to synchronize multiple concurrent branches (merging behavior). On the other hand, it is used to spawn new concurrent threads on parallel branches (branching behavior).

**Table 13.1 – Parallel Gateway Execution Semantics**

| Operational Semantics | The **Parallel Gateway** is activated if there is at least one *token* on each incoming **Sequence Flow**.
| | The **Parallel Gateway** consumes exactly one *token* from each incoming **Sequence Flow** and produces exactly one *token* at each outgoing **Sequence Flow**.
| | If there are excess *tokens* at an incoming **Sequence Flow**, these *tokens* remain at this **Sequence Flow** after execution of the **Gateway**.
| Exception Issues | The **Parallel Gateway** cannot throw any exception.
| Workflow Patterns Support | Parallel Split (WCP-2)
| | Synchronization (WCP-3)
Activities

Activities represent *pieces of work*

- Activities take time
- Activities are atomic (task) or subprocesses
- Subprocesses can be collapsed, if contained process is not relevant in current model
Activities cont.

Multiple instances

- Compact representation of activities that are executed multiple times
- Example: Activity is executed for each position of an order
- Resembles For-loop in common programming languages if executed sequentially

- Important attributes:
  LoopCharacteristics is of type MultiInstanceLoopCharacteristics,
Activities cont.

Loop activities

- Repeated execution of activity is represented by loop activity
- Condition determines whether execution is repeated
- Resembles While-loop or Repeat-Until-loop in common programming languages (depends on `testBefore = {true, false}`)
- Important attributes:
  - `LoopCharacteristics` is of type `StandardLoopCharacteristics`
Sequence Flow

Execution order is defined by sequence flow

Execution semantics of $A \rightarrow B$

- Activity $B$ can be started only once activity $A$ has ended

Realised by *signaling of flows*

- Once $A$ ends, a token is sent on the edge
- Once $B$ receives this token, it can start execution
- Tokens in BPMN cannot be distinguished (*black tokens*)
Execution Order

• Sequence flow allows for specifying sequential behaviour
• Complex logic is expressed by gateways
• Gateways have base form (diamond)
• Different symbols in the base form indicate gateway type
• Most commonly used
  • Data-based exclusive gateway (XOR gateway)
  • Parallel gateway (AND gateway)
Uncontrolled Flow

Tasks can have multiple incoming / outgoing sequence flows

“Uncontrolled“ flow semantics
• A token is sent on every outgoing flow
• Every token on an incoming flow results in execution

Best Practice:
Gateways

Data-based XOR gateway as split

- Select one out of a set of alternatives based on internal data
- Every flow leaving the gateway has attribute `ConditionType` set to "Expression" and a `ConditionExpression`
- A token is sent to first flow that evaluates to true

Data-based XOR gateway as join

- Merge alternative branches
- Sent token as soon as one token arrives at incoming flows
Gateways cont.

Parallel gateway, as split

- Sent token on each outgoing flow
- Allows for modelling concurrent execution

Parallel gateway, as join

- Gateway synchronises once a token has been received on all incoming flows
- Paths of parallel execution are joint
Gateways cont.

Inclusive OR gateway, as split
- A non-empty subset of outgoing flows is selected and a token is sent on those flows, at least one, at most all
- Can be seen as generalisation of the other two types

Inclusive OR gateway, as join
- Gateway waits until it received a token on all flows for which a token has been produced “upstream”
- Often used because of flexibility
- But: complex execution semantics (loops!)
Gateways cont.

Event-based gateway, as split
- Gateway is followed by catching intermediate events or receive tasks
- A token is sent on the flow of the first event to occur (or task to receive a message)

Event-based gateway, as join
- Same as XOR gateway
Events, Types and Triggers

Characteristics

- Events do not take time
- Can be catching or throwing
- Have type: Start, intermediate, end event

Start event (catching)
- Commonly, it leads to the creation of a new process instance

End event (throwing)
- Is triggered once a token arrives
- Commonly, it signals completion of a process instance

Intermediate event (catching / throwing)
- May occur in the course of processing

Event triggers
- Define business semantics (reception of a message) for increasing understandability
- Events have type and trigger, but not all combinations are valid
Start Events

- **Blank**
  - No concrete trigger
  - E.g., manual instantiation of a process

- **Conditional**
  - If condition becomes true, instantiate process

- **Message**
  - Receive a message

- **Signal**
  - Observe a milestone

- **Multiple**
  - Different alternatives to instantiate a process
End Events

- **Blank**
  - Ends execution path, not necessarily the process instance

- **Termination**
  - Ends process instance immediately

- **Message**
  - Sends message

- **Signal**
  - Signals a flag that may be reacted upon by the same or other instances
Intermediate Events

Intermediate events may be catching or throwing
- Catching: process waits for event to occur
- Throwing: process triggers events and continues

Intermediate events are connected to the process
- By sequence flow (catching or throwing)
- Attached to boundary of activity (only catching)

Message intermediate event
- Sending is done immediately
- Process is blocked until message is received
Intermediate Events cont.

Timer intermediate event: wait for trigger

- Boundary event: a token is sent on the flow leaving the event if timer is triggered before activity finished execution

- Time for pausing the process may be:
  - Duration (10 min, 15 days, …)
  - Point in time, absolute/relative (8:00 h, 2 days before travel, …)
Intermediate Events cont.

Error intermediate event: react to exceptions

- Must be catching and boundary event
- Used to define exception handling
Intermediate Events cont.

Condition intermediate event: react to changing conditions

- Processing continues only if condition is true
Signal Events

Send a signal inside a process (instance) or even beyond the boundaries of a process (instance)

- Broadcasting: a signal may be processed at different places
- Model complex control flow, like synchronisation between subprocesses
Link Events

Link events allow for connecting different parts of a process model without sequence flow.
Interrupting vs. Non-Interrupting

Catching boundary events catch event during execution of the parent activity

- Two ways to react (not valid for all event trigger)
  - *Interrupting*: activity is aborted
  - *Non-interrupting*: execution of activity continues, flow of boundary is activated concurrently
Example

Have friends over for dinner → Choose recipe → Prepare dinner → Enjoy dinner

Preparation

Further friend wants to join → Order pizza

Dinner is burnt → Invite further friend

Order pizza

 Invite further friend

Enjoy dinner
<table>
<thead>
<tr>
<th>Events</th>
<th>Start</th>
<th>Intermediate</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>None: Untyped events,</td>
<td>□</td>
<td><img src="image1" alt="Symbol" /></td>
<td><img src="image2" alt="Symbol" /></td>
</tr>
<tr>
<td>indicate start point, state</td>
<td><img src="image3" alt="Symbol" /></td>
<td><img src="image4" alt="Symbol" /></td>
<td><img src="image5" alt="Symbol" /></td>
</tr>
<tr>
<td>changes or final states.</td>
<td><img src="image6" alt="Symbol" /></td>
<td><img src="image7" alt="Symbol" /></td>
<td><img src="image8" alt="Symbol" /></td>
</tr>
<tr>
<td>Message: Receiving and sending</td>
<td><img src="image9" alt="Symbol" /></td>
<td><img src="image10" alt="Symbol" /></td>
<td><img src="image11" alt="Symbol" /></td>
</tr>
<tr>
<td>messages.</td>
<td><img src="image12" alt="Symbol" /></td>
<td><img src="image13" alt="Symbol" /></td>
<td><img src="image14" alt="Symbol" /></td>
</tr>
<tr>
<td>Timer: Cyclic timer events,</td>
<td><img src="image15" alt="Symbol" /></td>
<td><img src="image16" alt="Symbol" /></td>
<td><img src="image17" alt="Symbol" /></td>
</tr>
<tr>
<td>points in time, time spans or</td>
<td><img src="image18" alt="Symbol" /></td>
<td><img src="image19" alt="Symbol" /></td>
<td><img src="image20" alt="Symbol" /></td>
</tr>
<tr>
<td>timeouts.</td>
<td><img src="image21" alt="Symbol" /></td>
<td><img src="image22" alt="Symbol" /></td>
<td><img src="image23" alt="Symbol" /></td>
</tr>
<tr>
<td>Escalation: Escalating to</td>
<td><img src="image24" alt="Symbol" /></td>
<td><img src="image25" alt="Symbol" /></td>
<td><img src="image26" alt="Symbol" /></td>
</tr>
<tr>
<td>an higher level of responsibility.</td>
<td><img src="image27" alt="Symbol" /></td>
<td><img src="image28" alt="Symbol" /></td>
<td><img src="image29" alt="Symbol" /></td>
</tr>
<tr>
<td>Conditional: Reacting to</td>
<td><img src="image30" alt="Symbol" /></td>
<td><img src="image31" alt="Symbol" /></td>
<td><img src="image32" alt="Symbol" /></td>
</tr>
<tr>
<td>changed business conditions or</td>
<td><img src="image33" alt="Symbol" /></td>
<td><img src="image34" alt="Symbol" /></td>
<td><img src="image35" alt="Symbol" /></td>
</tr>
<tr>
<td>integrating business rules.</td>
<td><img src="image36" alt="Symbol" /></td>
<td><img src="image37" alt="Symbol" /></td>
<td><img src="image38" alt="Symbol" /></td>
</tr>
<tr>
<td>Link: Off-page connectors.</td>
<td><img src="image39" alt="Symbol" /></td>
<td><img src="image40" alt="Symbol" /></td>
<td><img src="image41" alt="Symbol" /></td>
</tr>
<tr>
<td>Two corresponding link events</td>
<td><img src="image42" alt="Symbol" /></td>
<td><img src="image43" alt="Symbol" /></td>
<td><img src="image44" alt="Symbol" /></td>
</tr>
<tr>
<td>equal a sequence flow.</td>
<td><img src="image45" alt="Symbol" /></td>
<td><img src="image46" alt="Symbol" /></td>
<td><img src="image47" alt="Symbol" /></td>
</tr>
<tr>
<td>Error: Catching or throwing</td>
<td><img src="image48" alt="Symbol" /></td>
<td><img src="image49" alt="Symbol" /></td>
<td><img src="image50" alt="Symbol" /></td>
</tr>
<tr>
<td>named errors.</td>
<td><img src="image51" alt="Symbol" /></td>
<td><img src="image52" alt="Symbol" /></td>
<td><img src="image53" alt="Symbol" /></td>
</tr>
<tr>
<td>Cancel: Reacting to cancelled</td>
<td><img src="image54" alt="Symbol" /></td>
<td><img src="image55" alt="Symbol" /></td>
<td><img src="image56" alt="Symbol" /></td>
</tr>
<tr>
<td>transactions or triggering</td>
<td><img src="image57" alt="Symbol" /></td>
<td><img src="image58" alt="Symbol" /></td>
<td><img src="image59" alt="Symbol" /></td>
</tr>
<tr>
<td>cancellation.</td>
<td><img src="image60" alt="Symbol" /></td>
<td><img src="image61" alt="Symbol" /></td>
<td><img src="image62" alt="Symbol" /></td>
</tr>
<tr>
<td>Compensation: Handling or</td>
<td><img src="image63" alt="Symbol" /></td>
<td><img src="image64" alt="Symbol" /></td>
<td><img src="image65" alt="Symbol" /></td>
</tr>
<tr>
<td>triggering compensation.</td>
<td><img src="image66" alt="Symbol" /></td>
<td><img src="image67" alt="Symbol" /></td>
<td><img src="image68" alt="Symbol" /></td>
</tr>
<tr>
<td>Signal: Signalling across</td>
<td><img src="image69" alt="Symbol" /></td>
<td><img src="image70" alt="Symbol" /></td>
<td><img src="image71" alt="Symbol" /></td>
</tr>
<tr>
<td>different processes. A signal</td>
<td><img src="image72" alt="Symbol" /></td>
<td><img src="image73" alt="Symbol" /></td>
<td><img src="image74" alt="Symbol" /></td>
</tr>
<tr>
<td>thrown can be caught multiple</td>
<td><img src="image75" alt="Symbol" /></td>
<td><img src="image76" alt="Symbol" /></td>
<td><img src="image77" alt="Symbol" /></td>
</tr>
<tr>
<td>times.</td>
<td><img src="image78" alt="Symbol" /></td>
<td><img src="image79" alt="Symbol" /></td>
<td><img src="image80" alt="Symbol" /></td>
</tr>
<tr>
<td>Multiple: Catching one out of</td>
<td><img src="image81" alt="Symbol" /></td>
<td><img src="image82" alt="Symbol" /></td>
<td><img src="image83" alt="Symbol" /></td>
</tr>
<tr>
<td>a set of events. Throwing all</td>
<td><img src="image84" alt="Symbol" /></td>
<td><img src="image85" alt="Symbol" /></td>
<td><img src="image86" alt="Symbol" /></td>
</tr>
<tr>
<td>events defined.</td>
<td><img src="image87" alt="Symbol" /></td>
<td><img src="image88" alt="Symbol" /></td>
<td><img src="image89" alt="Symbol" /></td>
</tr>
<tr>
<td>Parallel Multiple: Catching</td>
<td><img src="image90" alt="Symbol" /></td>
<td><img src="image91" alt="Symbol" /></td>
<td><img src="image92" alt="Symbol" /></td>
</tr>
<tr>
<td>all out of a set of parallel</td>
<td><img src="image93" alt="Symbol" /></td>
<td><img src="image94" alt="Symbol" /></td>
<td><img src="image95" alt="Symbol" /></td>
</tr>
<tr>
<td>events.</td>
<td><img src="image96" alt="Symbol" /></td>
<td><img src="image97" alt="Symbol" /></td>
<td><img src="image98" alt="Symbol" /></td>
</tr>
<tr>
<td>Terminate: Triggering the</td>
<td><img src="image99" alt="Symbol" /></td>
<td><img src="image100" alt="Symbol" /></td>
<td><img src="image101" alt="Symbol" /></td>
</tr>
<tr>
<td>immediate termination of a</td>
<td><img src="image102" alt="Symbol" /></td>
<td><img src="image103" alt="Symbol" /></td>
<td><img src="image104" alt="Symbol" /></td>
</tr>
<tr>
<td>process.</td>
<td><img src="image105" alt="Symbol" /></td>
<td><img src="image106" alt="Symbol" /></td>
<td><img src="image107" alt="Symbol" /></td>
</tr>
</tbody>
</table>
Data in Processes

- Activities can read and write data objects, represented by directed associations
- Associating a data object to sequence flow is interpreted as data transfer
- Data objects have states that may change during processing

Diagram:

- Task A
- Task B
- Document A
- Document B
- Task A
- Passed data
- Document
  - [state 1]
- Document
  - [state 2]
Input / Output of Processes

Symbols to represent input and output for process as a whole

- Input data must be available to execute activity
- Exception: attribute `optional = true`

Data store describes a place where the process can read or write data

- For instance, information system, shelf, ...
- Is independent of process lifecycle

Collection data object represents multiple instances of a data object (type)
Roles: Pools and Lanes

- Responsibility is defined by roles, those are depicted graphically.
- Modelling of the internal structure of an organisation and interactions with other organisations.
Interaction between Organisations

Interaction between organisations is realised solely by sending and receiving messages

- Media (e-mail, mail, fax, telephone) is often abstracted

Sequence flow only for internal dependencies of an organisation

- Only for internal dependencies, the order of activity execution can be enforced
- Besides message flow, there are no means to influence processing of a partner

Rule in BPMN

- Sequence flow only inside of a pool (may be implicit)
- Message flow only between different pools
White Box Pool vs. Black Box Pool

• If internal structure is not relevant, collapse pool (black box pool)
• Then, message flow is attached to the pool
• Background:
  • If interactions are discussed, the internal process of a partner is often unknown or not of interest
  • Still, one can discuss the general message exchange
  • BPMN 2.0 provides means to precisely define interaction protocols using choreography modelling
White Box Pool vs. Black Box Pool
Event-based Gateway, again

Common situation

• After request, one waits for response
• Solved by using an event-based gateway for the response messages
• Time-out by timer intermediate event allows for reacting, e.g., send request again
Event Stream Processing Primer
Scenario: Logistics

Real-time planning in logistics aims at

- Reduced slack time
- Reduced risk of missed connections
- Efficient vehicle utilisation

Based on

- Positions of vehicles
- Recent processing times
- Current workloads
Scenario: Cluster Monitoring

Real-time cluster monitoring aims at

- Efficient job execution
- Reduced number of evicted jobs
- Identification of stragglers

Based on

- Resource availability
- Machine utilisation
- Job scheduling
Detection of Complex Events

Observation:
- Most events are not interesting
- New events supersede old events
- Ability to react to changing situations provides value

Derive complex events from simple events
Database Management System (DBMS):
Data relatively static but queries dynamic

- Persistent relations
  - Random access
  - Low update rate
  - Unbounded disk storage

- One-time queries
  - Finite query result
  - Queries exploit (static) indices
Event Recognition System:
Queries static but data dynamic - input is time-dependant stream

Transient streams
- Sequential access
- Potentially high rate
- Bounded main memory

Continuous queries
- Produce time-dependant result stream
- Indexing?
Event Recognition: Performance Matters!

Value of analytics decreases over time

Decision making benefits from timeliness of analytics
- Limited windows of opportunities (now or never)
- Competitive advantage (quicker than the rest)

Compliance and performance assessment
- Early detection of deviations
- Early start of remedy actions
Events

What is an event?

An event is a happening of interest. An event type is a specification of a set of events of the same structure and semantics. [Etzion and Niblett (2011)]

Cluster monitoring use case:

- Events denote transitions in job/task lifecycle
- Events indicate availability of machines
Event Types

How to model events?

Event schema defined as set of attributes

- Payload of event is a set of key-value pairs
- Events often have associated time stamp
- E.g. arrival time, time of reading, ...

Cluster monitoring:

<table>
<thead>
<tr>
<th>Task events table</th>
</tr>
</thead>
<tbody>
<tr>
<td>The task events table contains the following fields:</td>
</tr>
<tr>
<td>1. timestamp</td>
</tr>
<tr>
<td>2. missing info</td>
</tr>
<tr>
<td>3. job ID</td>
</tr>
<tr>
<td>4. task index - within the job</td>
</tr>
<tr>
<td>5. machine ID</td>
</tr>
<tr>
<td>6. event type</td>
</tr>
<tr>
<td>7. user name</td>
</tr>
<tr>
<td>8. scheduling class</td>
</tr>
<tr>
<td>9. priority</td>
</tr>
<tr>
<td>10. resource request for CPU cores</td>
</tr>
<tr>
<td>11. resource request for RAM</td>
</tr>
<tr>
<td>12. resource request for local disk space</td>
</tr>
<tr>
<td>13. different-machine constraint</td>
</tr>
</tbody>
</table>

Schedule$_1$

(1444026993, -1, 239, 3, B-2, Schedule, rmalik, ... )
Streams

What is a stream?

A stream is a **real-time, continuous, ordered** (implicitly by arrival time or explicitly by timestamp) **sequence of items**. It is impossible to control the order in which items arrive, nor is it feasible to locally store a stream in its entirety. [Golab & Oszu (SIGMOD 2003)]

Data stream processing view: items are data tuples

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>miss</td>
<td>job ID</td>
<td>time</td>
<td>miss</td>
</tr>
</tbody>
</table>

Complex event processing view: items are typed events

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$x_1$</th>
<th>$S_2$</th>
<th>$E_1$</th>
</tr>
</thead>
</table>

Time
Data Stream Processing Languages

Idea:
Lift the relational model for queries to streams

SELECT timestamp, job, avg(cpu) AS avgCpu
FROM clusterEvents [range 60 slide 1]
WHERE eventType == 1
GROUP BY job

Implicit streaming operator

Predicates

Time window definition
Event Pattern Languages

Zoo of pattern specification languages
- Common core concepts
- Different syntax
- Subtle differences in semantics

Pattern SEQ(Schedule a, Schedule+ b[], Evict c)
Where skip-till-any-match
And b[].machine = a.machine
And a.job = c.job And a.task = c.task
Within 2 days
Return a.(job, task), b[].job
Next Steps
<table>
<thead>
<tr>
<th>Tentative Dates</th>
<th>Phases</th>
<th>Meeting</th>
<th>Deliverables</th>
</tr>
</thead>
<tbody>
<tr>
<td>19/04/2016</td>
<td>Organisation and planning</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td>until 10.05.2016</td>
<td>Domain and Requirement Analysis, Projectplanning</td>
<td>all</td>
<td>Spec. &amp; projectplan</td>
</tr>
<tr>
<td></td>
<td>Design: Interfaces, inter-team data structures, file formats and possible test cases</td>
<td>self-organised</td>
<td>File format &amp; test cases</td>
</tr>
<tr>
<td></td>
<td>Design: Intra-team data structures, architecture, algorithms</td>
<td>self-organised</td>
<td></td>
</tr>
<tr>
<td>31/05/2016</td>
<td>Inter-team presentation of design</td>
<td>all</td>
<td>System Design</td>
</tr>
<tr>
<td>14/06/2016</td>
<td>Inter-team presentation and mutual testing of prototype I</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Intermediate debugging</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td>05/07/2016</td>
<td>Inter-team presentation and mutual testing of prototype II</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Final debugging</td>
<td>all</td>
<td></td>
</tr>
<tr>
<td>19/07/2016</td>
<td>Final presentation</td>
<td>all</td>
<td>Final implementation</td>
</tr>
<tr>
<td></td>
<td>TBD Project closing</td>
<td>all</td>
<td>Final documentation</td>
</tr>
</tbody>
</table>
Specification and Project Plan

What:
- Clearly define scope of the problem to be solved (in vs. out)
- Relate to functionality and APIs of the used engine
- Functional and non-functional requirements

How:
- Assess and illustrate dependencies between requirements
- Estimate effort and required resources to fulfil each requirement
- Instantiate general timeline for specific engine
  - Milestones in terms of fulfilled requirements
  - Risks and mitigation strategies

Where: in github, using your wiki
When: by 10.05.2016, please notify us by mail