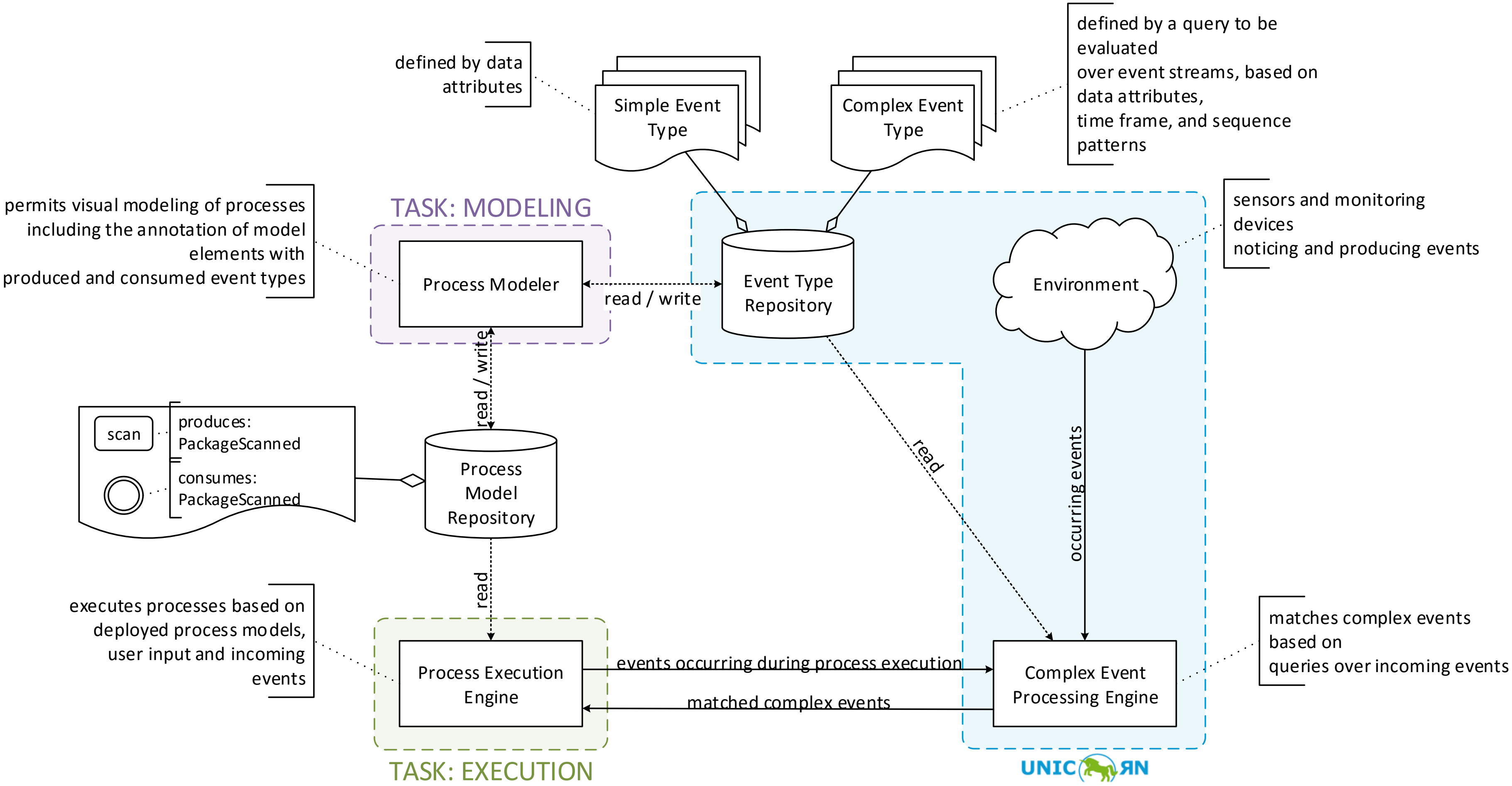
Event-driven Process Engines

Background

Matthias Weidlich

Setting



defined by data attributes
Simple Event Type
Complex Event Type
defined by a query to be evaluated
over event streams, based on data attributes,
time frame, and sequence patterns
permits visual modeling of processes including the annotation of model elements with
produced and consumed event types
TASK: MODELING
Process Modeler
read / write
Event Type Repository
Environment
sensors and monitoring devices
noticing and producing events
read / write
scan
produces: PackageScanned
consumes: PackageScanned
Process Model Repository
read
occurring events
read
executes processes based on deployed process models,
user input and incoming events
Process Execution Engine
events occurring during process execution
matched complex events
Complex Event Processing Engine
matches complex events based on
queries over incoming events
TASK: EXECUTION
UNICORN

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

BPM Lifecycle

Starting point

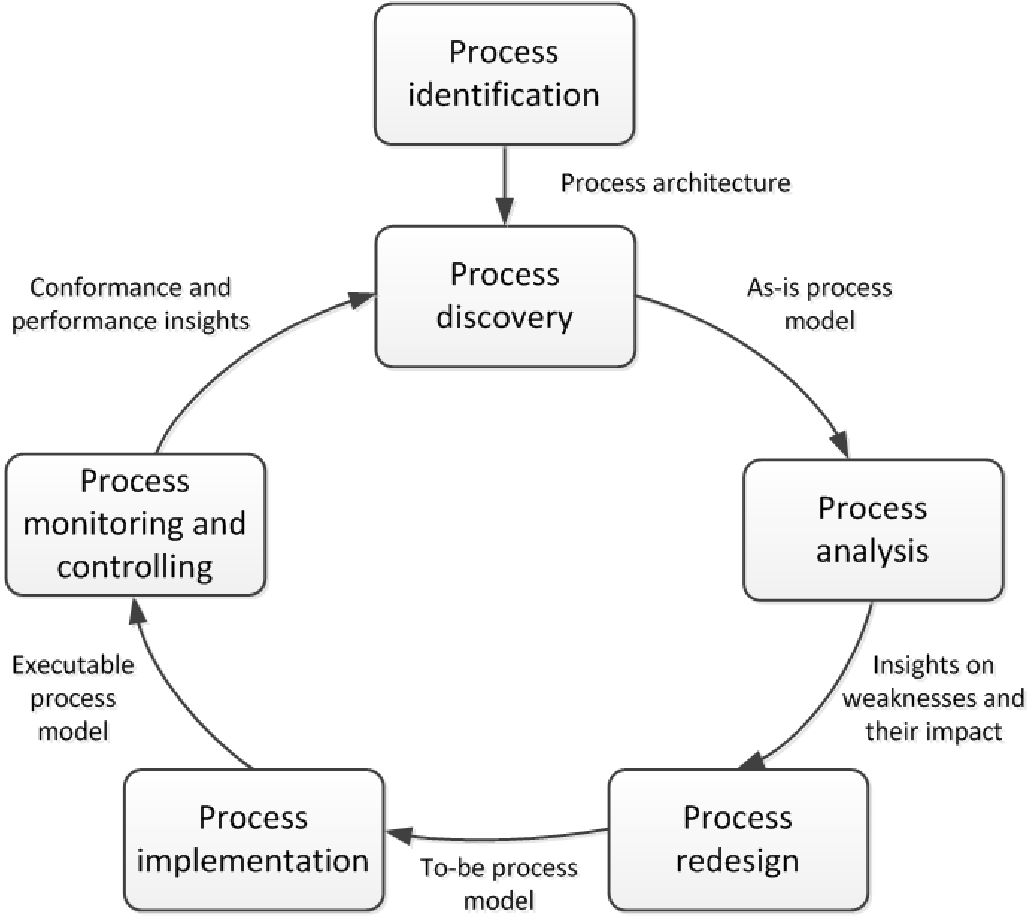
- Radical changes work out only under specific conditions
- Re-engineering neglects continuous changes of environment

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



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

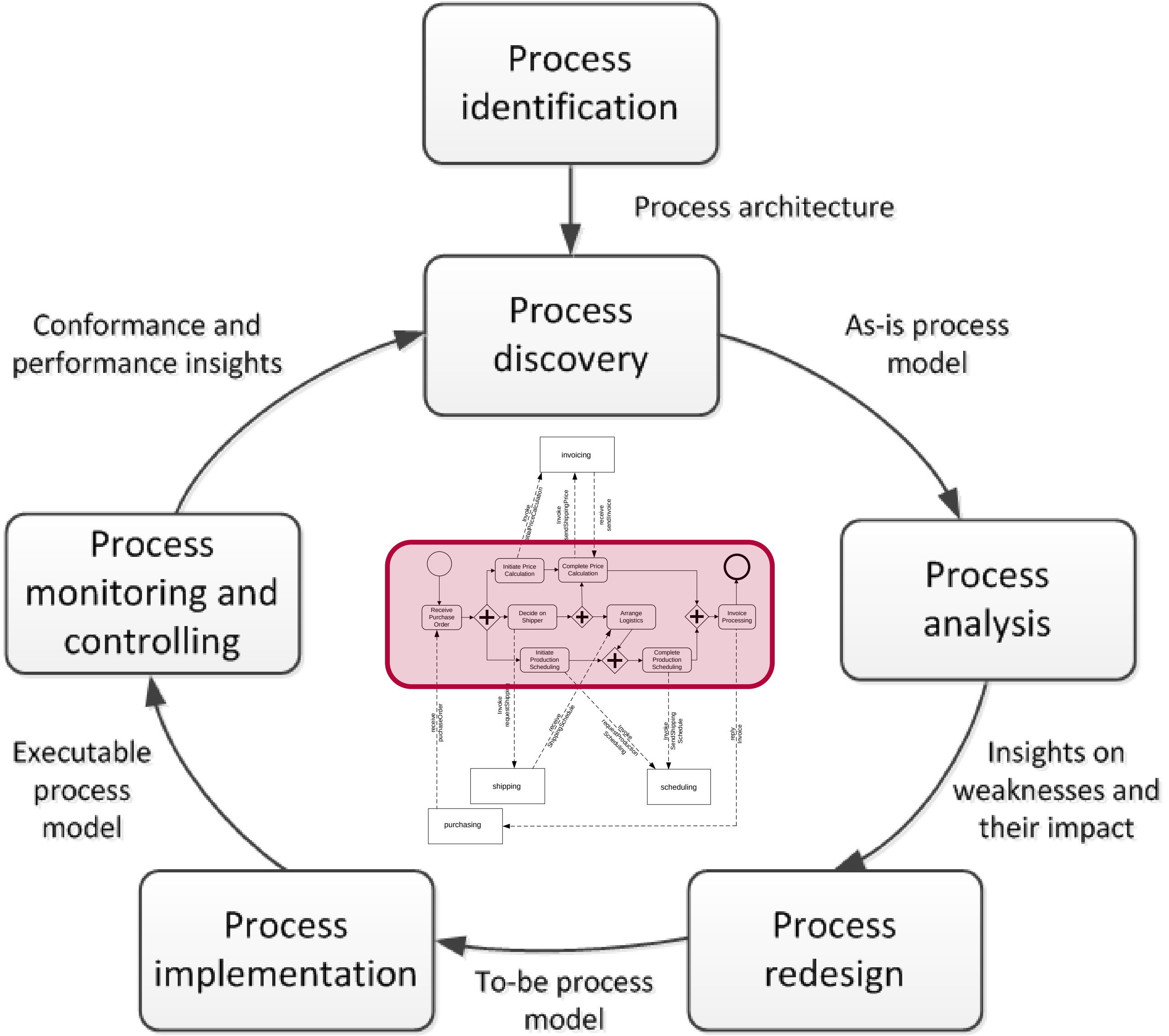
Process-oriented Information System (POIS)

- *”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



Process identification

Process architecture

Process discovery

As-is process model

Process analysis

Insights on weaknesses and their impact

Process redesign

To-be process model

Process implementation

Executable process model

Process monitoring and controlling

Conformance and performance insights

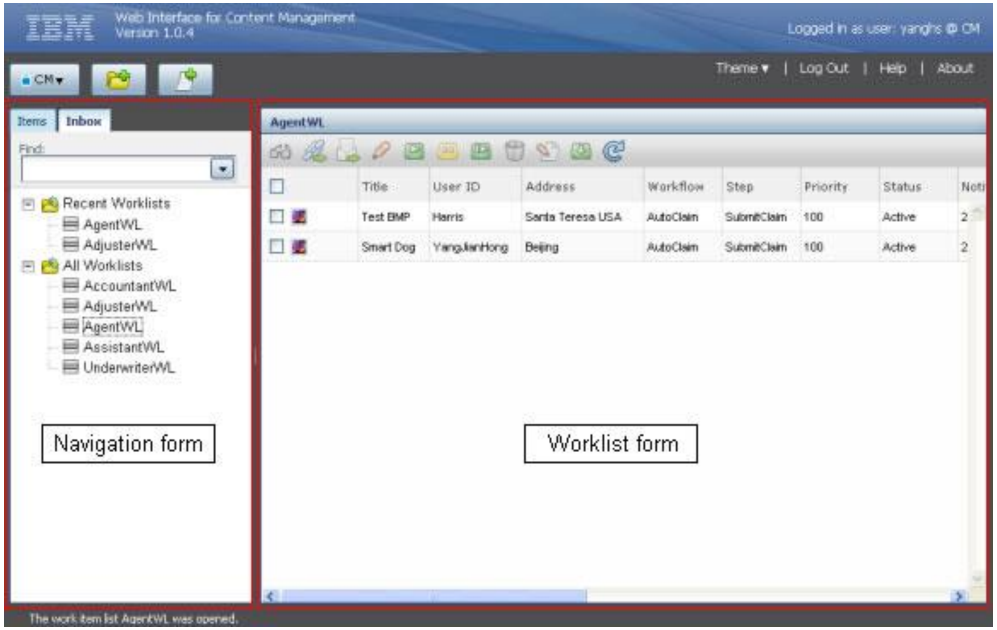
Beyond System Workflows

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)



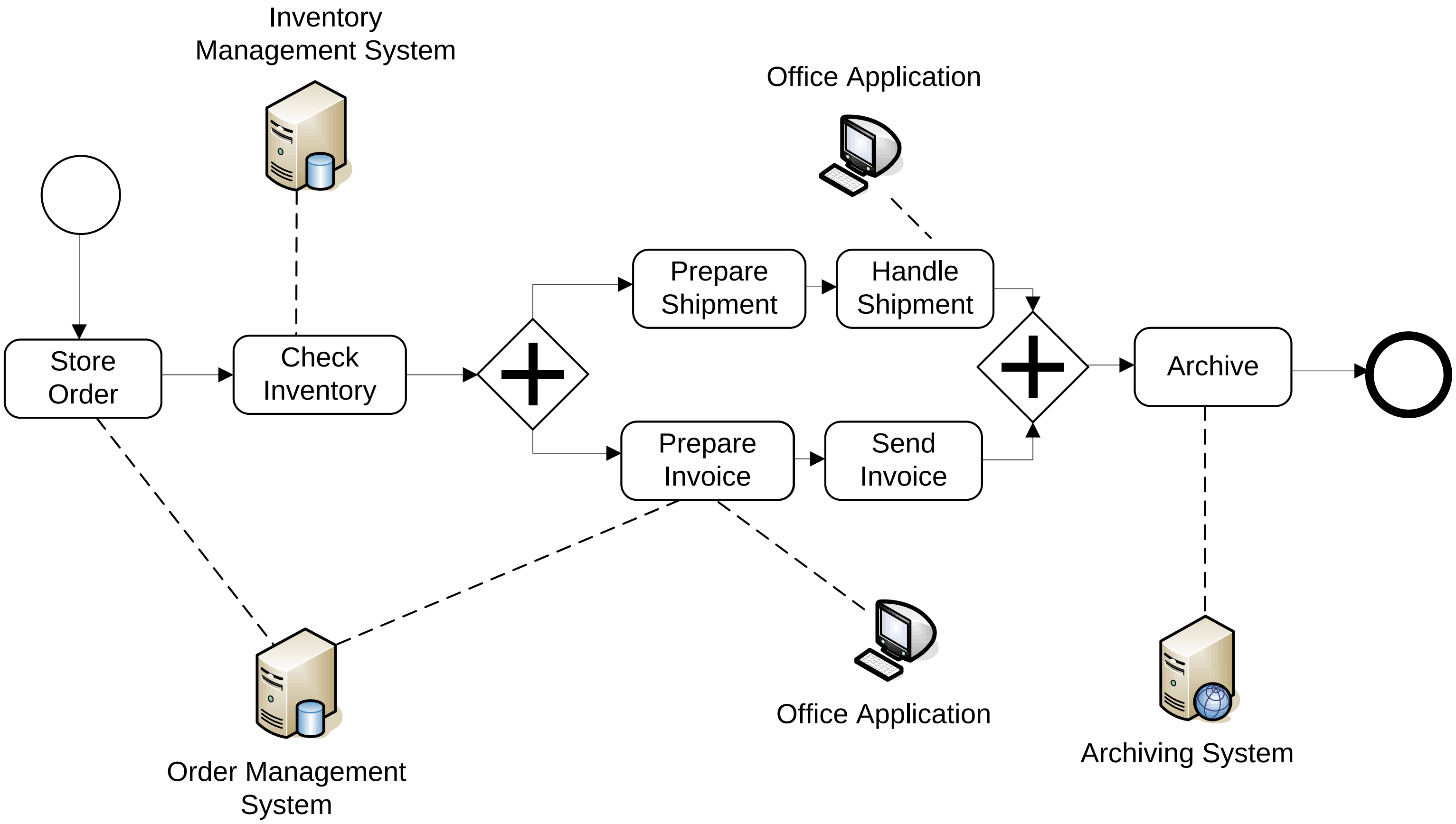
□		Title	User ID	Address	Workflow	Step	Priority	Status	Noti
□		Test BMP	Harris	Santa Teresa USA	AutoClaim	SubmitClaim	100	Active	2
□		Smart Dog	YangJianHong	Beijing	AutoClaim	SubmitClaim	100	Active	2

Navigation form

Worklist form

The work item list AgentWL was opened.

Example of a Human Interaction Workflow



from M. Weske: Business Process Management, © Springer-Verlag Berlin Heidelberg 2007

Process Modelling - The How

Process Models

Original

© Stiftung Deutsches Technikmuseum Berlin

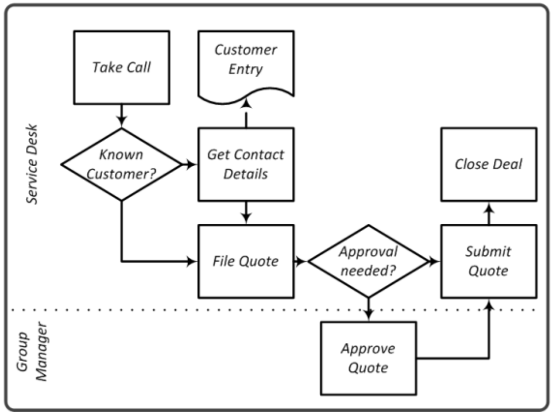
Abstraction

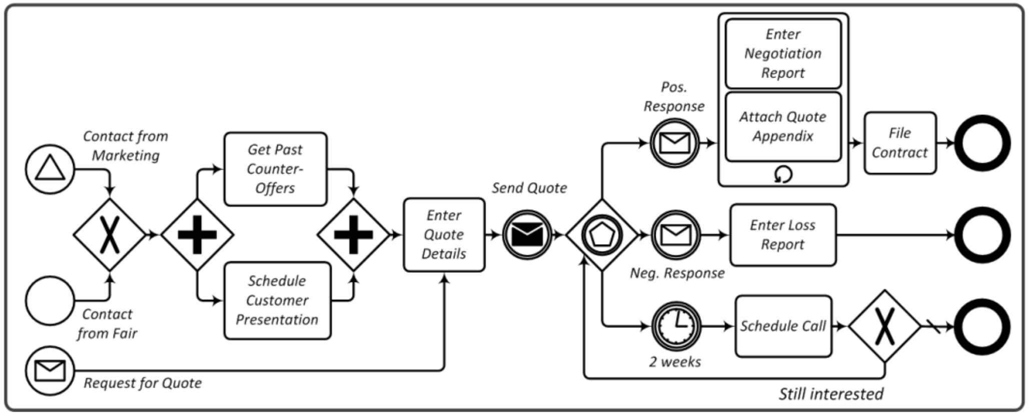
Model of

Abstraction

Model of

Model Level



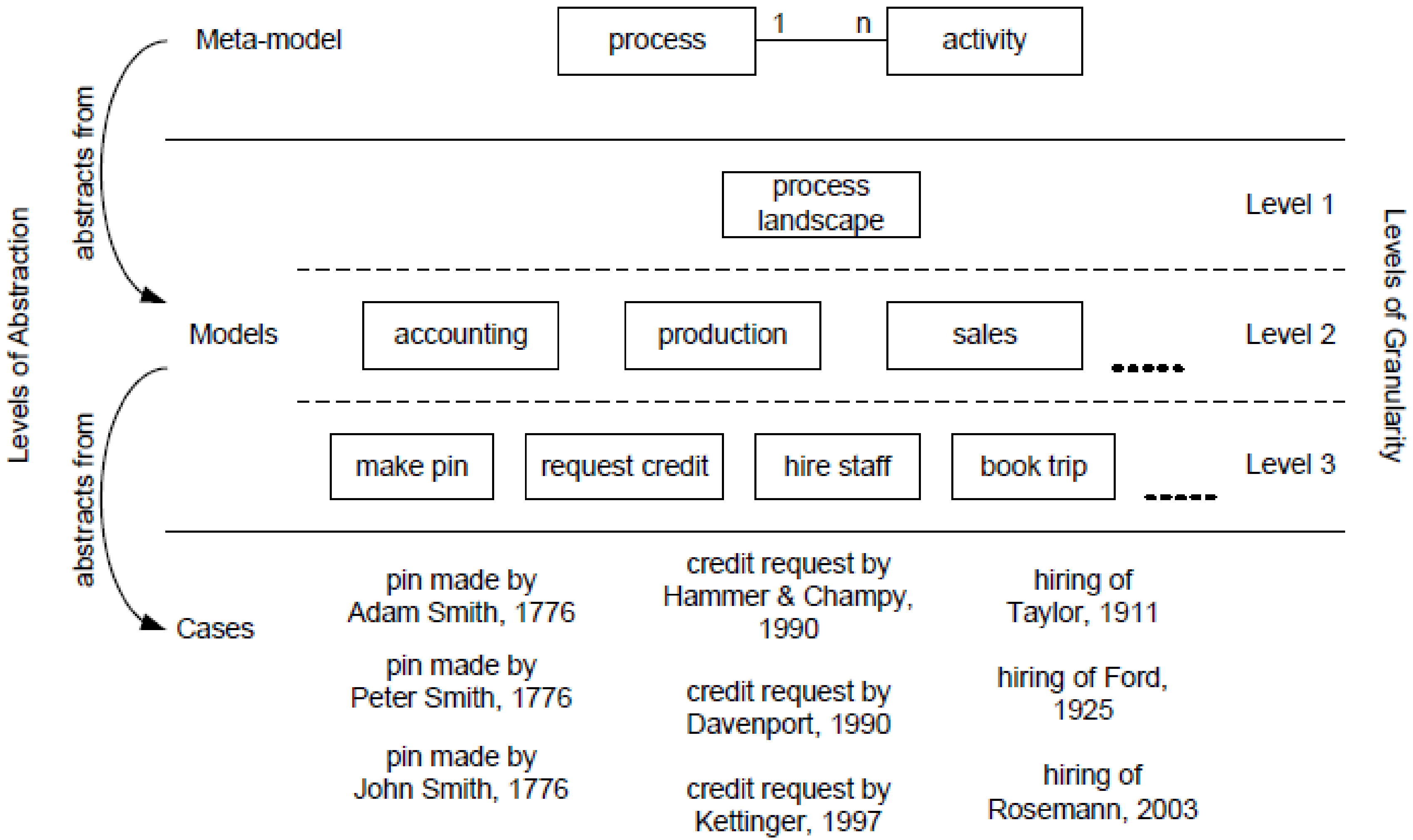


Mapping Business Processes

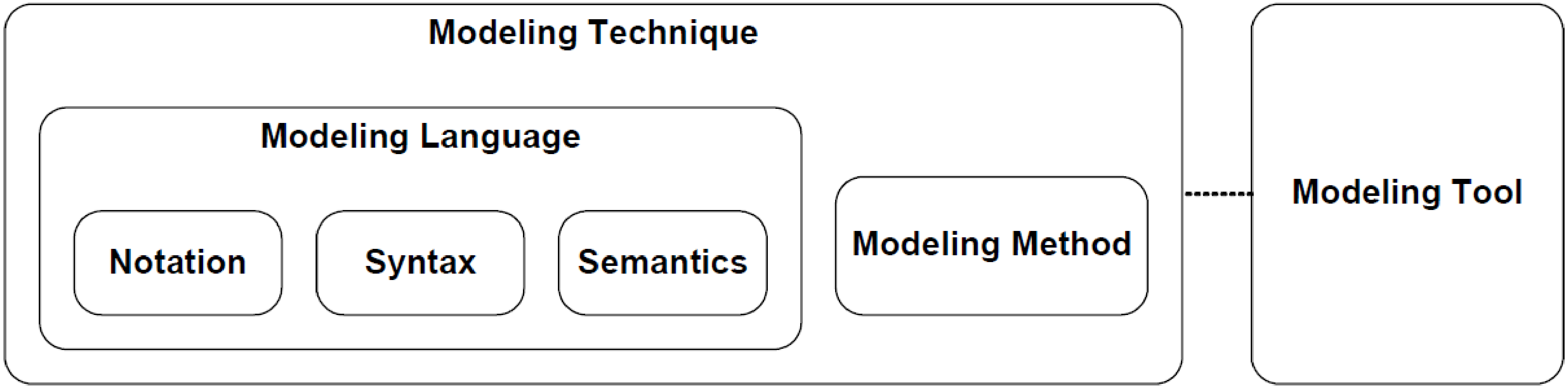
What is mapped to a process model?

- 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



Process Modelling – How?



Business Process Model and Notation (BPMN)

Business Process Model and Notation

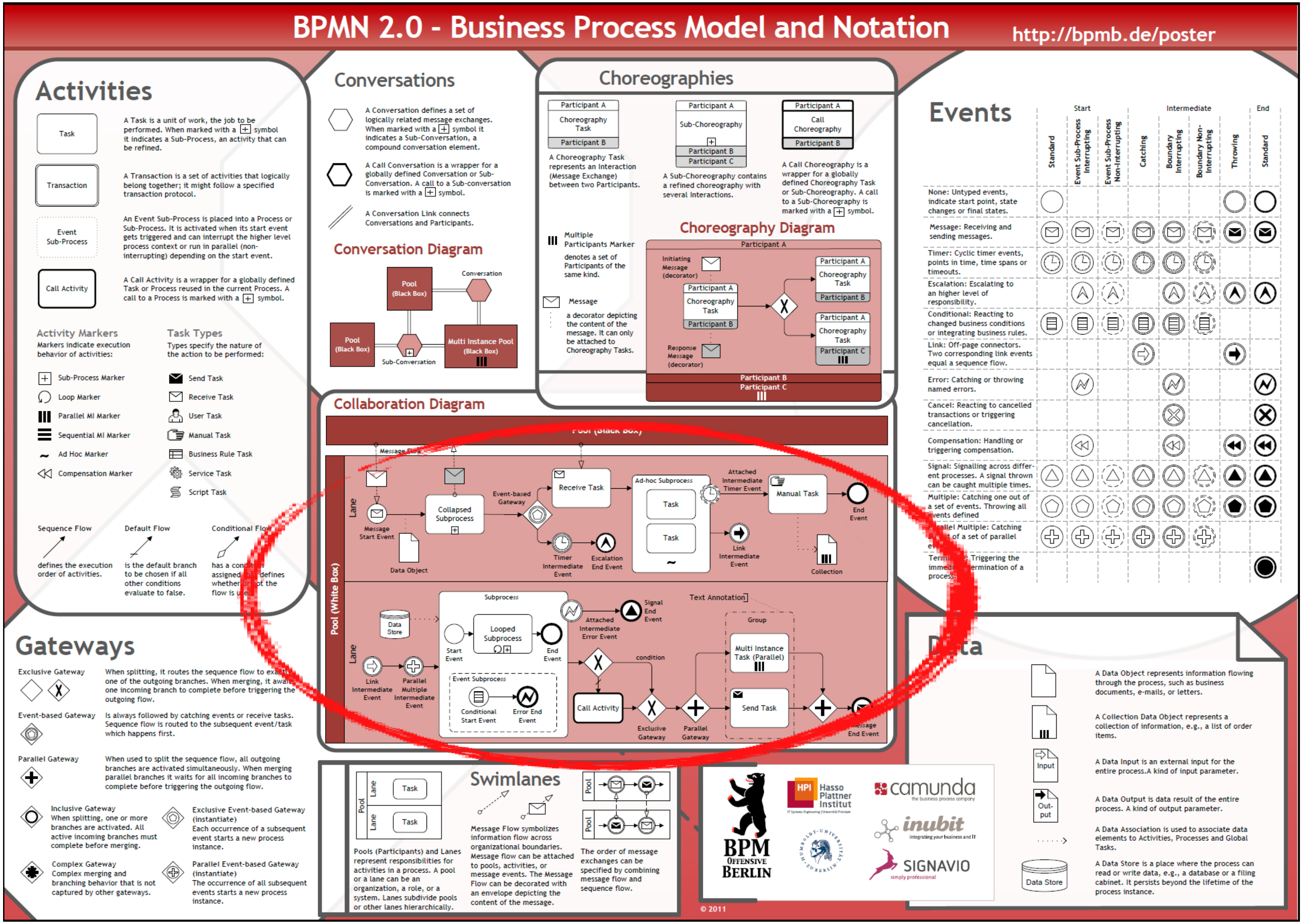
BPMN, version 2.0

- 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

BPMN Poster (http://bpmb.de/index.php/BPMNPoster)



BPMN 2.0 - Business Process Model and Notation

http://bpmb.de/poster

Activities

Task: A Task is a unit of work, the job to be performed. When marked with a ⊞ symbol it indicates a Sub-Process, an activity that can be refined.

Transaction: A Transaction is a set of activities that logically belong together; it might follow a specified transaction protocol.

Event Sub-Process: An Event Sub-Process is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.

Call Activity: A Call Activity is a wrapper for a globally defined Task or Process reused in the current Process. A call to a Process is marked with a ⊞ symbol.

Activity Markers

Markers indicate execution behavior of activities:

- ⊞ Sub-Process Marker
- Loop Marker
- ||| Parallel MI Marker
- ☰ Sequential MI Marker
- ~ Ad Hoc Marker
- ⏪ Compensation Marker

Task Types

Types specify the nature of the action to be performed:

- ✉ Send Task
- ✉ Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

Sequence Flow: defines the execution order of activities.

Default Flow: is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow: has a condition assigned that defines whether or not the flow is used.

Gateways

Exclusive Gateway: When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway: Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway: When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Inclusive Gateway: When splitting, one or more branches are activated. All active incoming branches must complete before merging.

Exclusive Event-based Gateway (instantiate): Each occurrence of a subsequent event starts a new process instance.

Complex Gateway: Complex merging and branching behavior that is not captured by other gateways.

Parallel Event-based Gateway (instantiate): The occurrence of all subsequent events starts a new process instance.

Conversations

A Conversation defines a set of logically related message exchanges. When marked with a ⊞ symbol it indicates a Sub-Conversation, a compound conversation element.

A Call Conversation is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-conversation is marked with a ⊞ symbol.

A Conversation Link connects Conversations and Participants.

Conversation Diagram

Choreographies

A Choreography Task represents an Interaction (Message Exchange) between two Participants.

A Sub-Choreography contains a refined choreography with several Interactions.

A Call Choreography is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a ⊞ symbol.

||| Multiple Participants Marker denotes a set of Participants of the same kind.

✉ Message: a decorator depicting the content of the message. It can only be attached to Choreography Tasks.

Choreography Diagram

Collaboration Diagram

Events

	Start			Intermediate				End
	Standard	Event Sub-Process Interrupting	Event Sub-Process Non-Interrupting	Catching	Boundary Interrupting	Boundary Non-Interrupting	Throwing	Standard
None: Untyped events, indicate start point, state changes or final states.	○						○	○
Message: Receiving and sending messages.	✉	✉	✉	✉	✉	✉	✉	✉
Timer: Cyclic timer events, points in time, time spans or timeouts.	◷	◷	◷	◷	◷	◷		
Escalation: Escalating to an higher level of responsibility.		⌃	⌃		⌃	⌃	▲	▲
Conditional: Reacting to changed business conditions or integrating business rules.	▤	▤	▤	▤	▤	▤		
Link: Off-page connectors. Two corresponding link events equal a sequence flow.				⇨			➡	
Error: Catching or throwing named errors.		ϟ			ϟ			ϟ
Cancel: Reacting to cancelled transactions or triggering cancellation.					⊗			⊗
Compensation: Handling or triggering compensation.		⏪			⏪		⏪	⏪
Signal: Signalling across different processes. A signal thrown can be caught multiple times.	△	△	△	△	△	△	▲	▲
Multiple: Catching one out of a set of events. Throwing all events defined	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallel Multiple: Catching all out of a set of parallel events.	✚	✚	✚	✚	✚	✚		
Terminate: Triggering the immediate termination of a process.								●

Data

A Data Object represents information flowing through the process, such as business documents, e-mails, or letters.

A Collection Data Object represents a collection of information, e.g., a list of order items.

Input: A Data Input is an external input for the entire process.A kind of input parameter.

Out-put: A Data Output is data result of the entire process. A kind of output parameter.

A Data Association is used to associate data elements to Activities, Processes and Global Tasks.

Data Store: A Data Store is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

Swimlanes

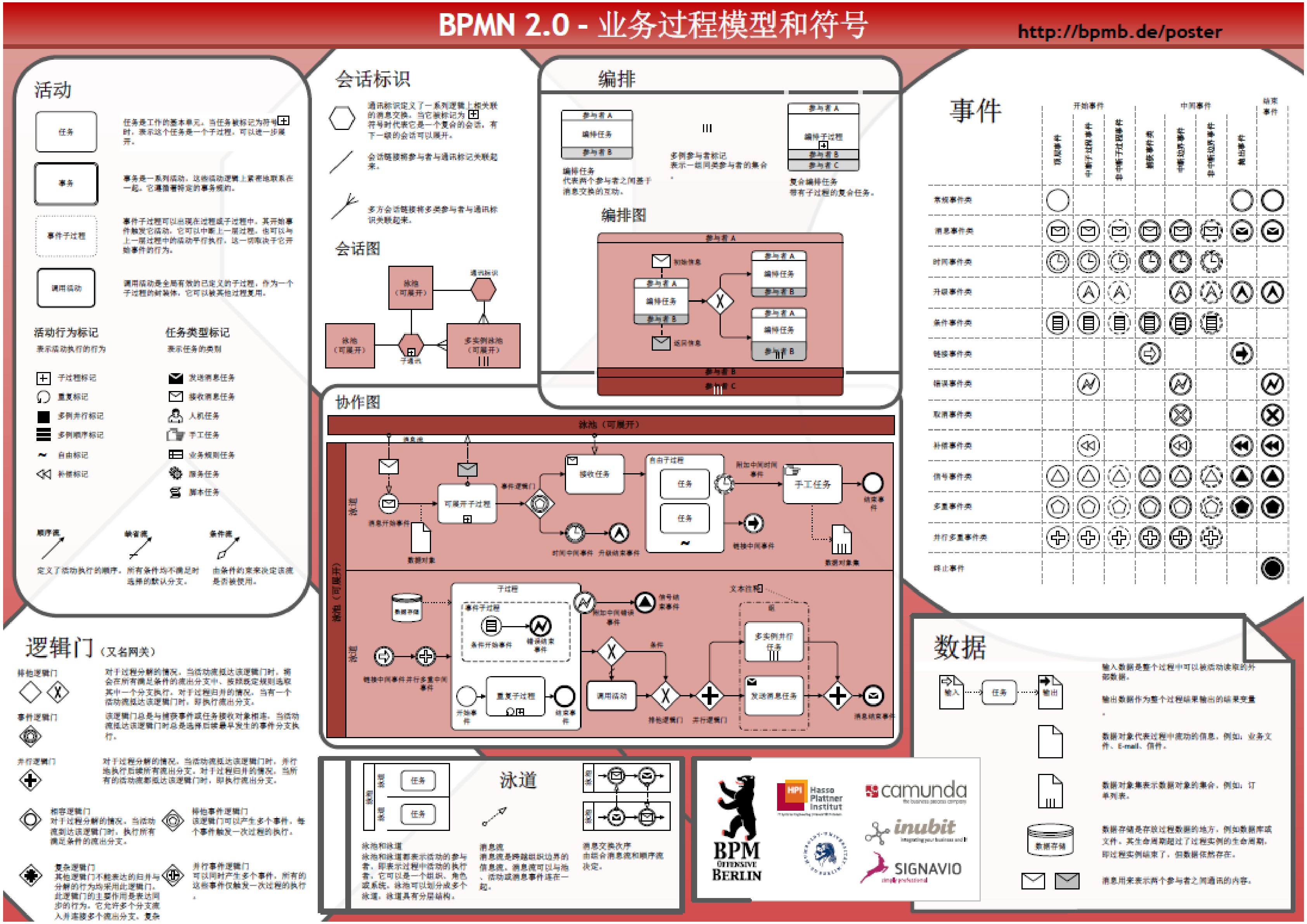
Pools (Participants) and Lanes represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events. The Message Flow can be decorated with an envelope depicting the content of the message.

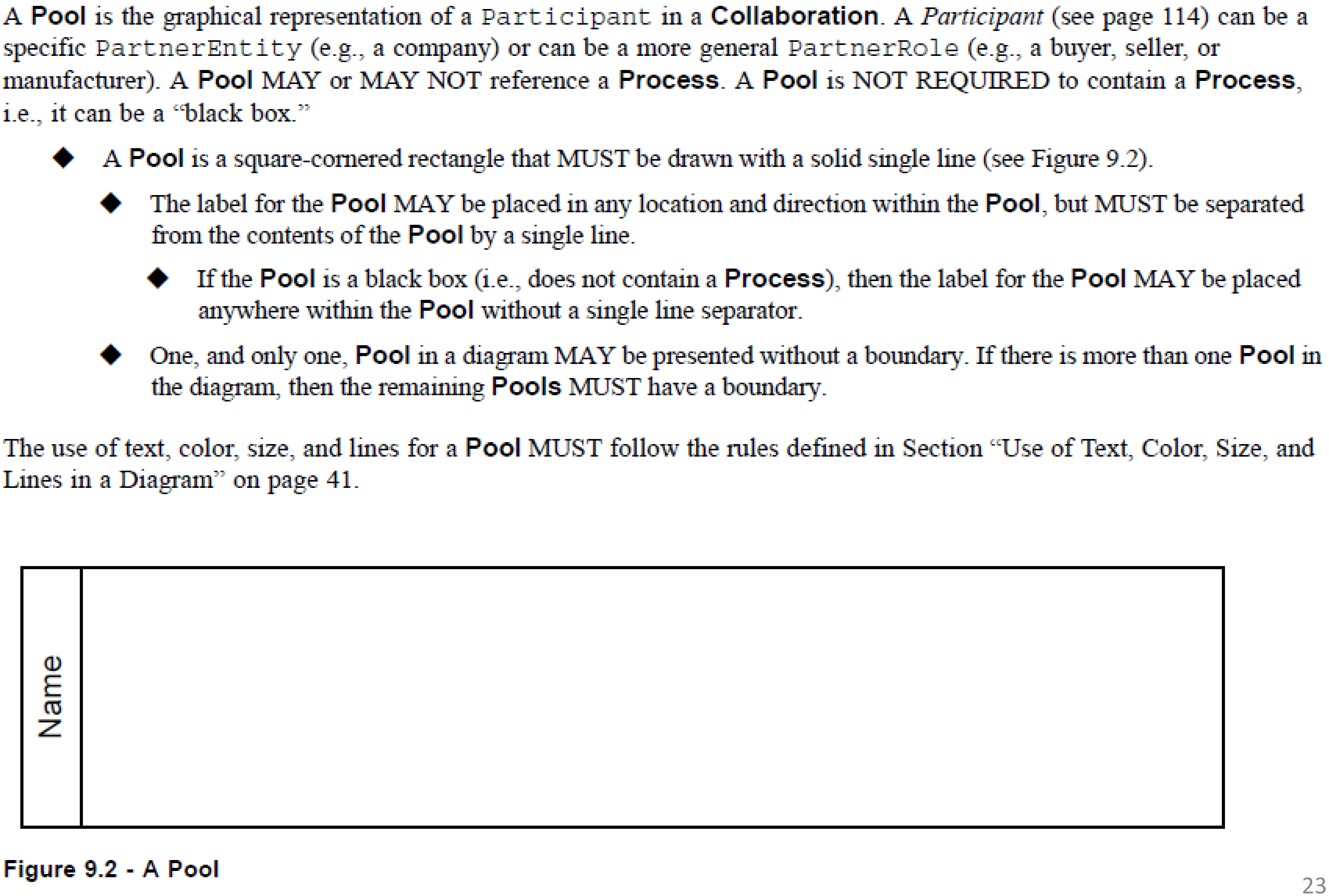
The order of message exchanges can be specified by combining message flow and sequence flow.

BPM Offensive Berlin
HPI Hasso Plattner Institut
camunda the business process company
inubit integrating your business and IT
SIGNAVIO simply professional

Check that...


BPMN 2.0 - 业务过程模型和符号
http://bpmb.de/poster
活动
任务
事务
事件子过程
调用活动
活动行为标记
任务类型标记
会话标识
会话图
编排
编排图
协作图
事件
逻辑门（又名网关）
泳道
数据
BPM OFFENSIVE BERLIN
Hasso Plattner Institut
camunda
inubit
SIGNAVIO

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.

Figure 9.2 - A Pool

Meta Model

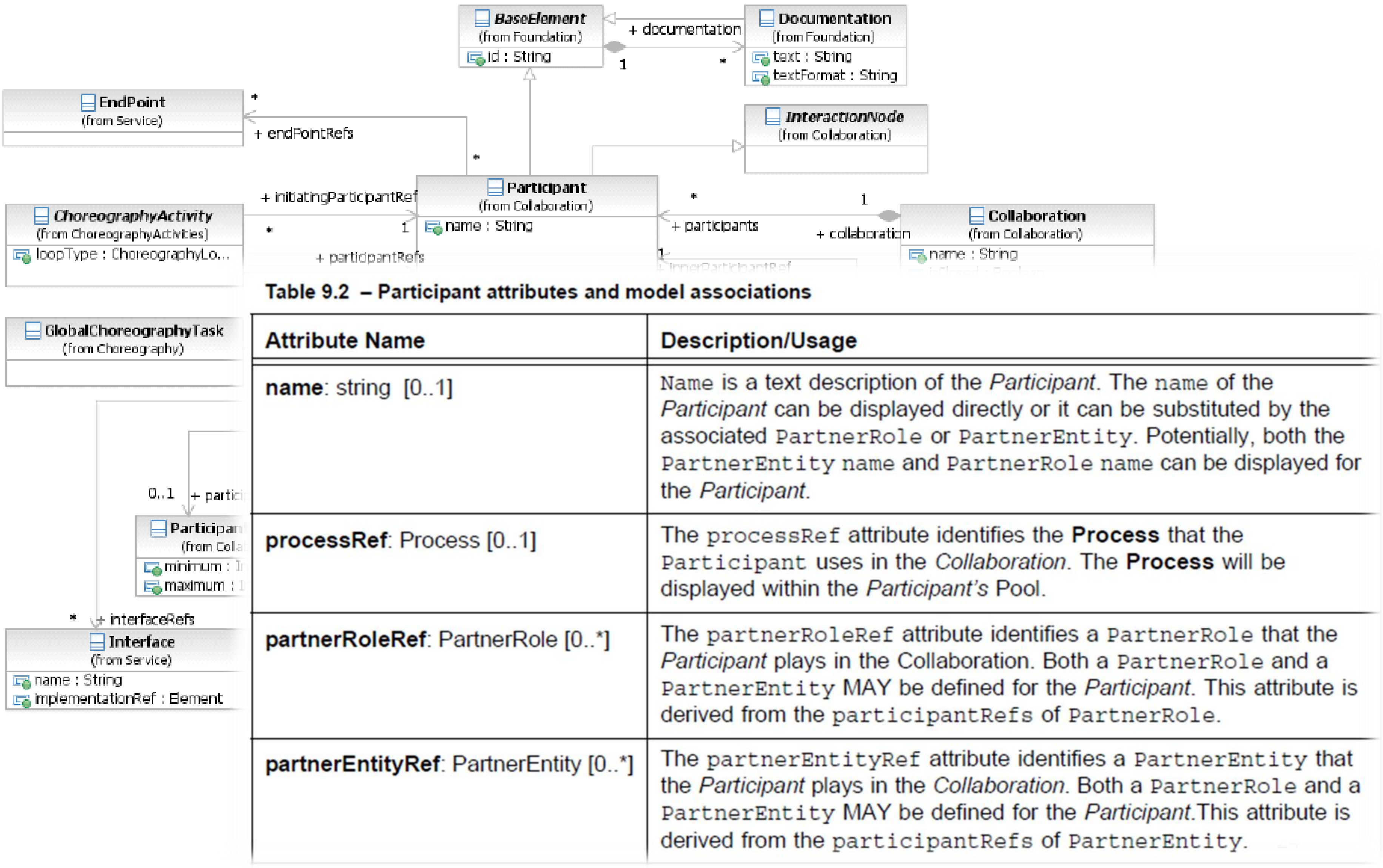


Table 9.2 – Participant attributes and model associations

Attribute Name	Description/Usage
name: string [0..1]	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*.
processRef: Process [0..1]	The processRef attribute identifies the **Process** that the Participant uses in the *Collaboration*. The **Process** will be displayed within the *Participant's* Pool.
partnerRoleRef: PartnerRole [0..*]	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.
partnerEntityRef: PartnerEntity [0..*]	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.

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?

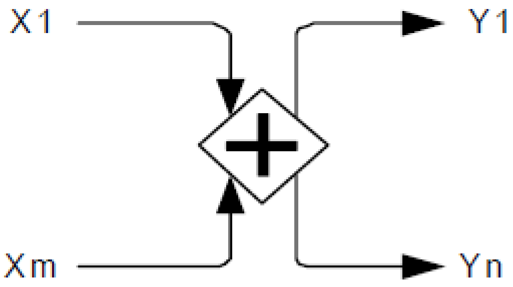
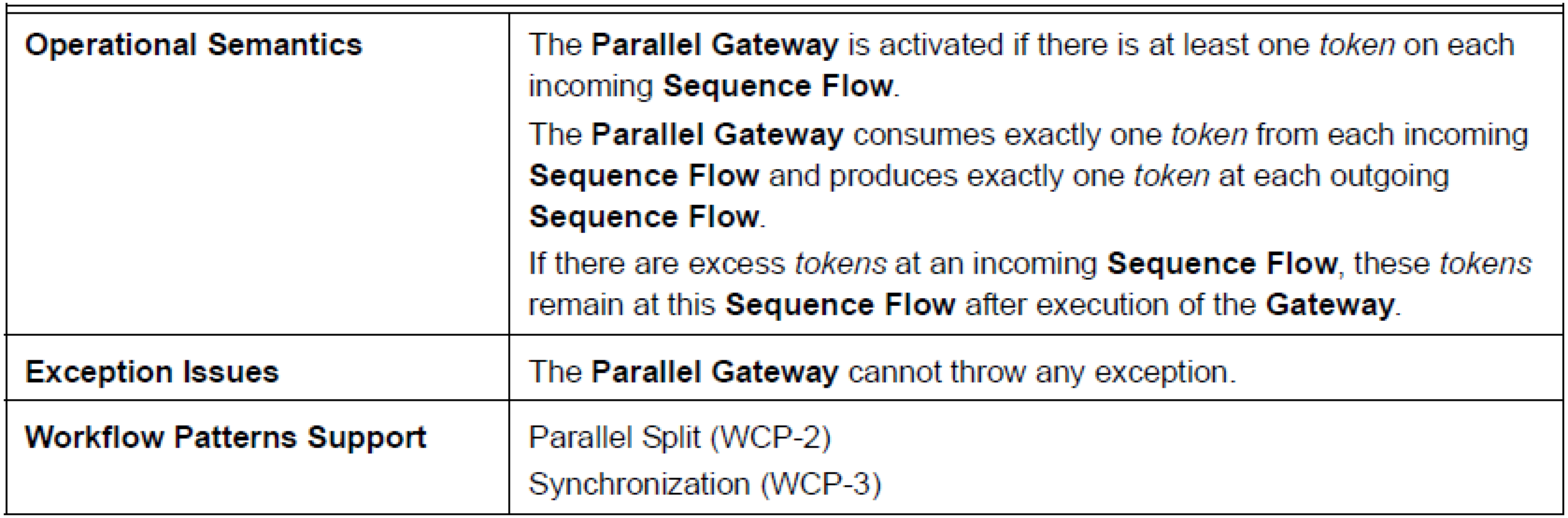


Figure 13.3 - Merging and Branching Sequence Flows for a Parallel Gateway

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

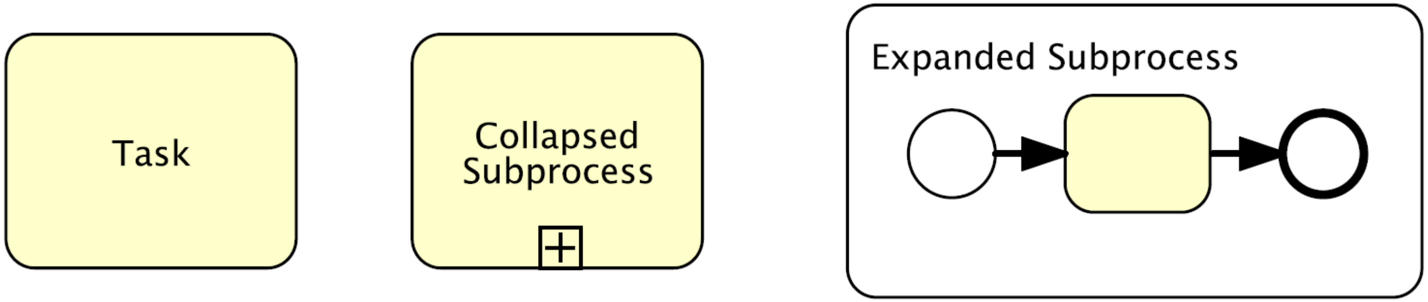


Operational Semantics	The **Parallel Gateway** is activated if there is at least one *token* on each incoming **Sequence Flow**. The **Parallel Gateway** consumes exactly one *token* from each incoming **Sequence Flow** and produces exactly one *token* at each outgoing **Sequence Flow**. If there are 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*

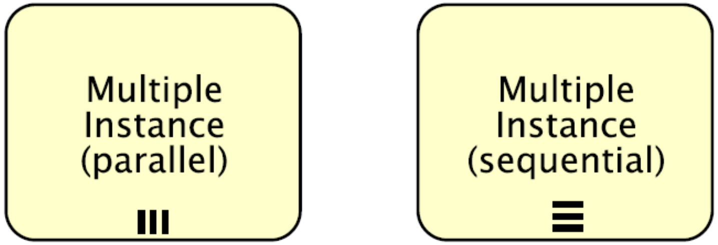
- Activities take time
- Activities are atomic (task) or subprocesses
- Subprocesses can be collapsed, if contained process is not relevant in current model



Activities cont.

Multiple instances

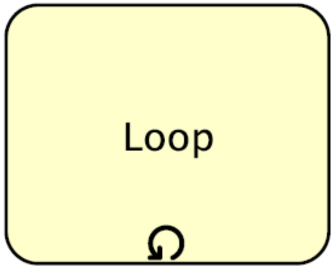
- Compact representation of activities that are executed multiple times
- Example: Activity is executed for each position of an order
- Resembles For-loop in common programming languages if executed sequentially
- Important attributes: `LoopCharacteristics` is of type `MultiInstanceLoopCharacteristics`,



Activities cont.

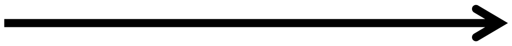
Loop activities

- 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 attributs:
  `LoopCharacteristics` is of type
  `StandardLoopCharacteristics`



Sequence Flow

Execution order is defined by sequence flow



Execution semantics of A → 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

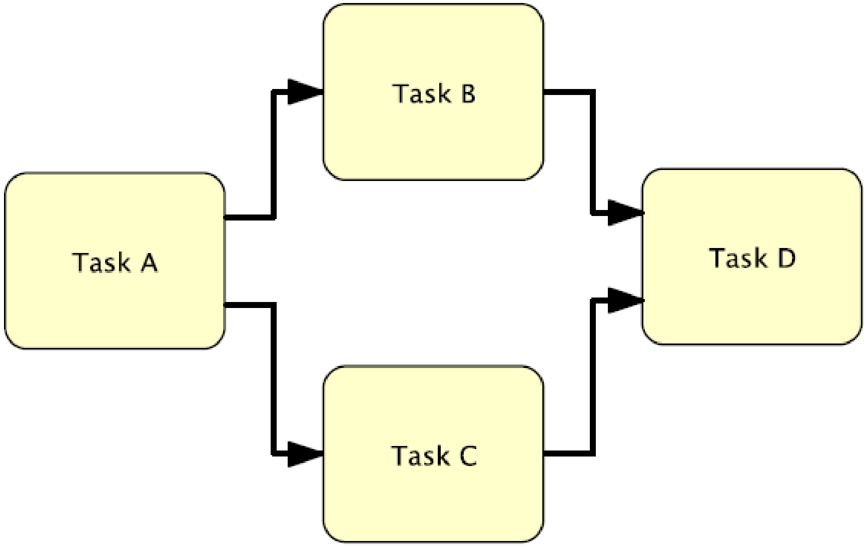
Tasks can have multiple incoming / outgoing sequence flows

“Uncontrolled“ flow semantics

- A token is sent on every outgoing flow
- Every token on an incoming flow results in execution

Best Practice:

🚫



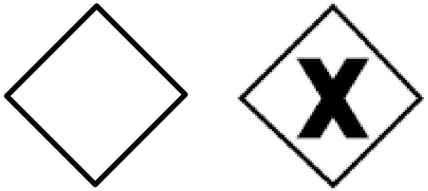
Gateways

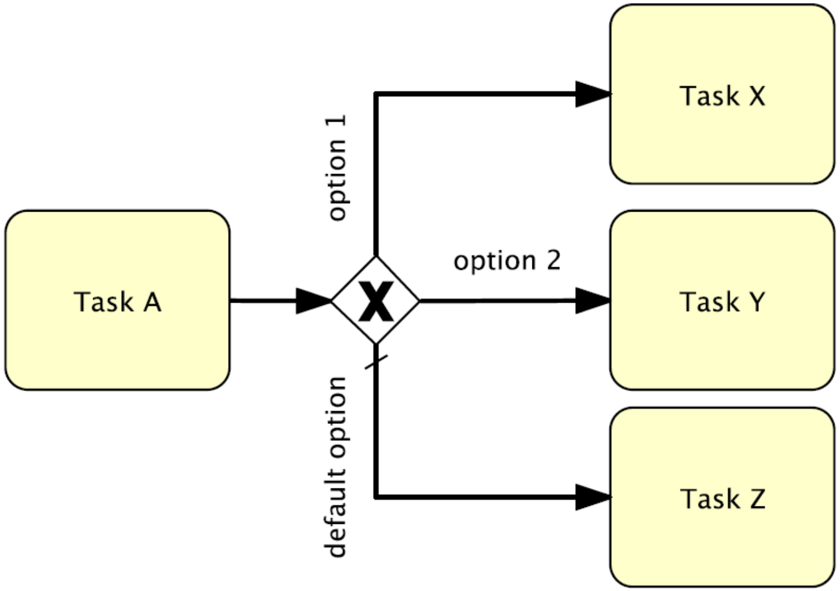
Data-based XOR gateway as split

- 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





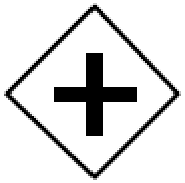
Gateways cont.

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



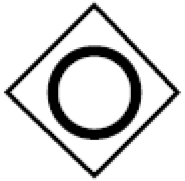
Gateways cont.

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!)



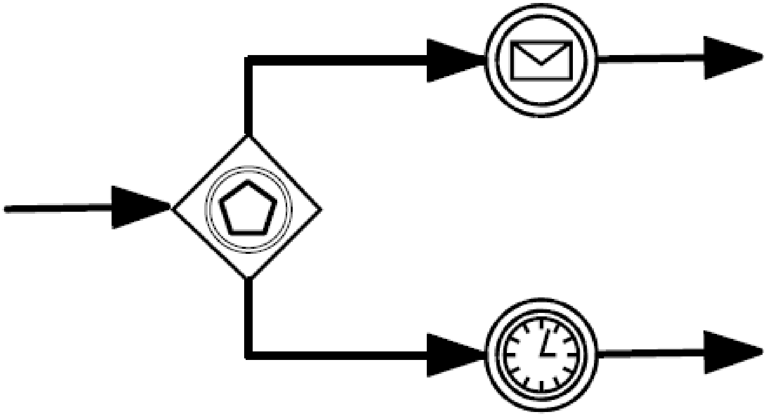
Gateways cont.

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)

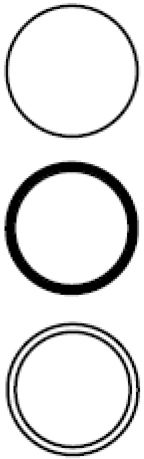
- Commonly, it leads to the creation of a new process instance

End event (throwing)

- Is triggered once a token arrives
- Commonly, it signals completion of a process instance

Intermediate event (catching / throwing)

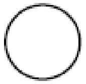
- May occur in the course of processing



Event triggers

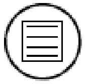
- Define business semantics (reception of a message) for increasing understandability
- Events have type and trigger, but not all combinations are valid

Start Events



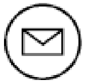
Blank

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



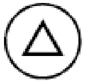
Conditional

- If condition becomes true, instantiate process



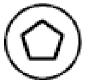
Message

- Receive a message



Signal

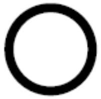
- Observe a milestone



Multiple

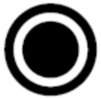
- Different alternatives to instantiate a process

End Events



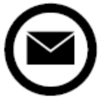
Blank

- Ends execution path, not necessarily the process instance



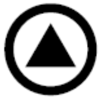
Termination

- 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

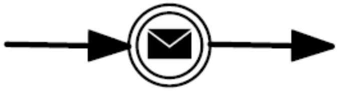
- Catching: process waits for event to occur
- Throwing: process triggers events and continues

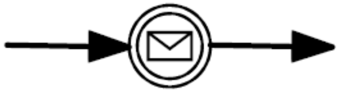
Intermediate events are connected to the process

- 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

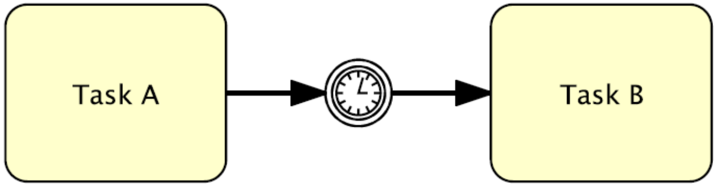

send message

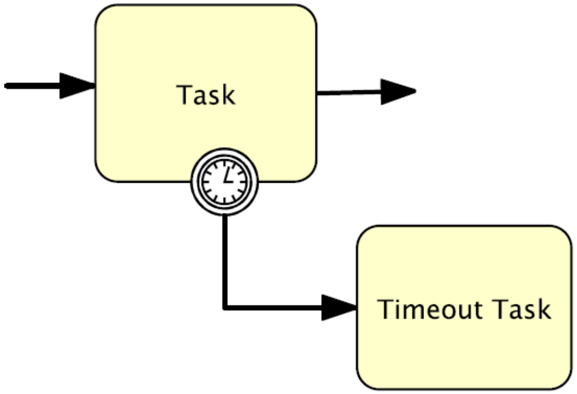

receive message

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, …)

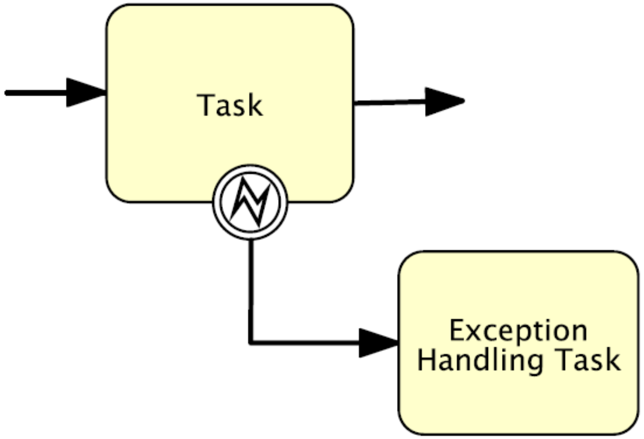


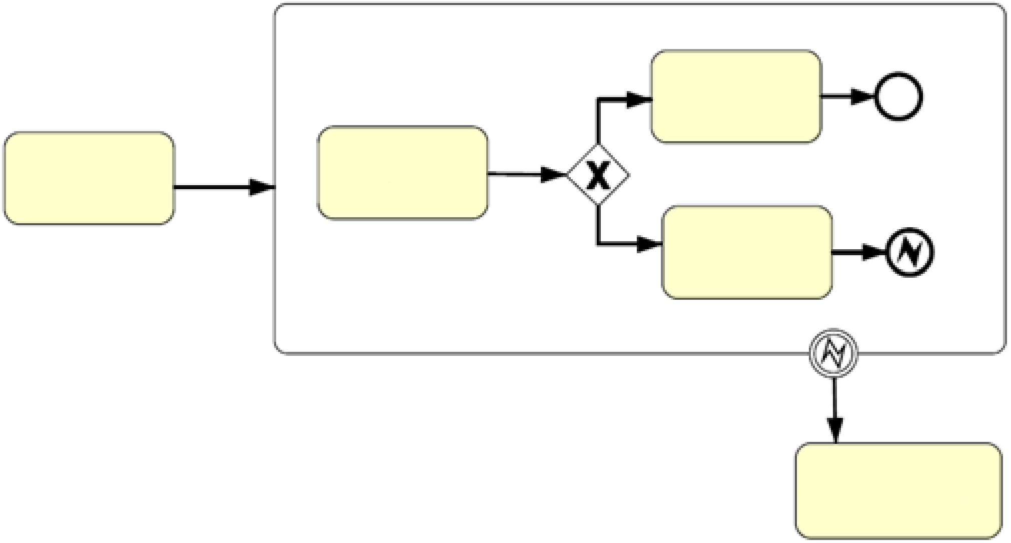


Intermediate Events cont.

Error intermediate event: react to exceptions

- Must be catching and boundary event
- Used to define exception handling

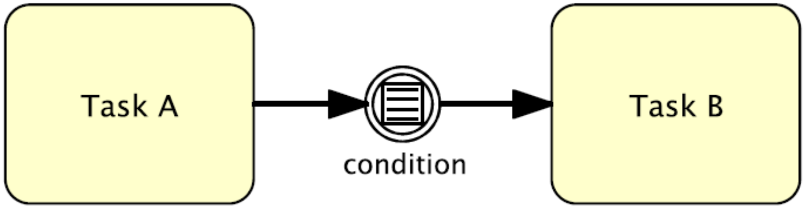


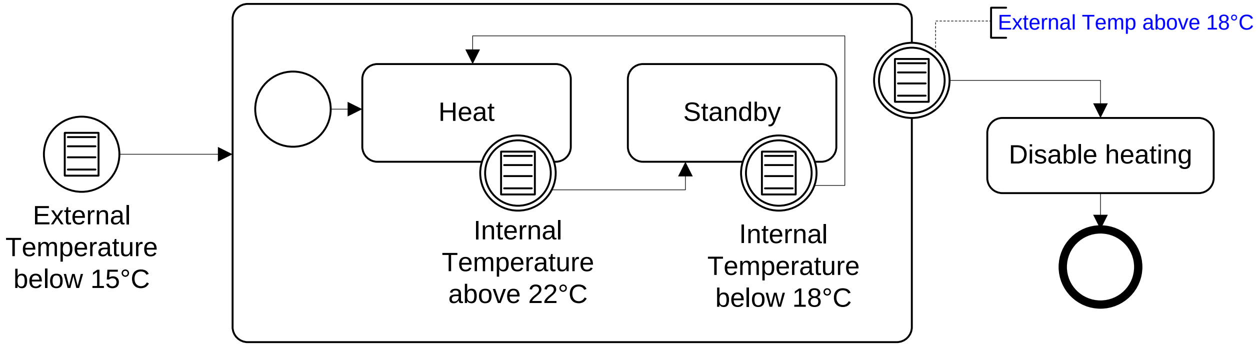


Intermediate Events cont.

Condition intermediate event: react to changing conditions

- Processing continues only if condition is true

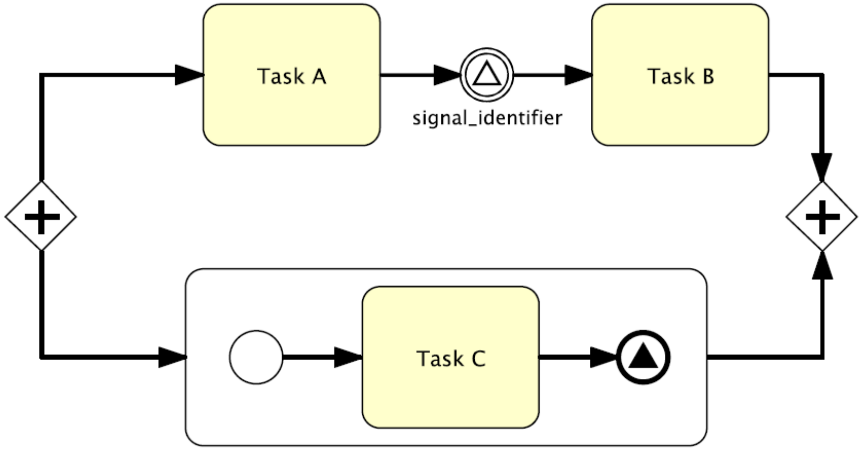


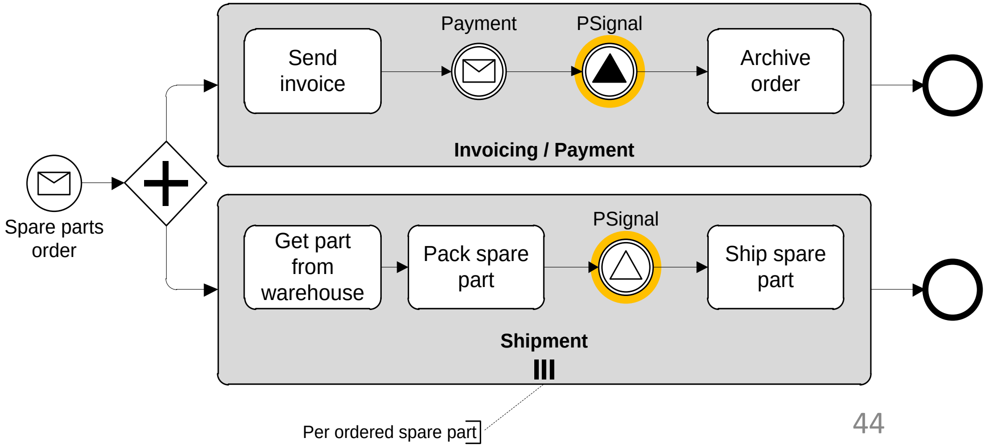

External Temperature below 15°C
Heat
Standby
Internal Temperature above 22°C
Internal Temperature below 18°C
External Temp above 18°C
Disable heating

Signal Events

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

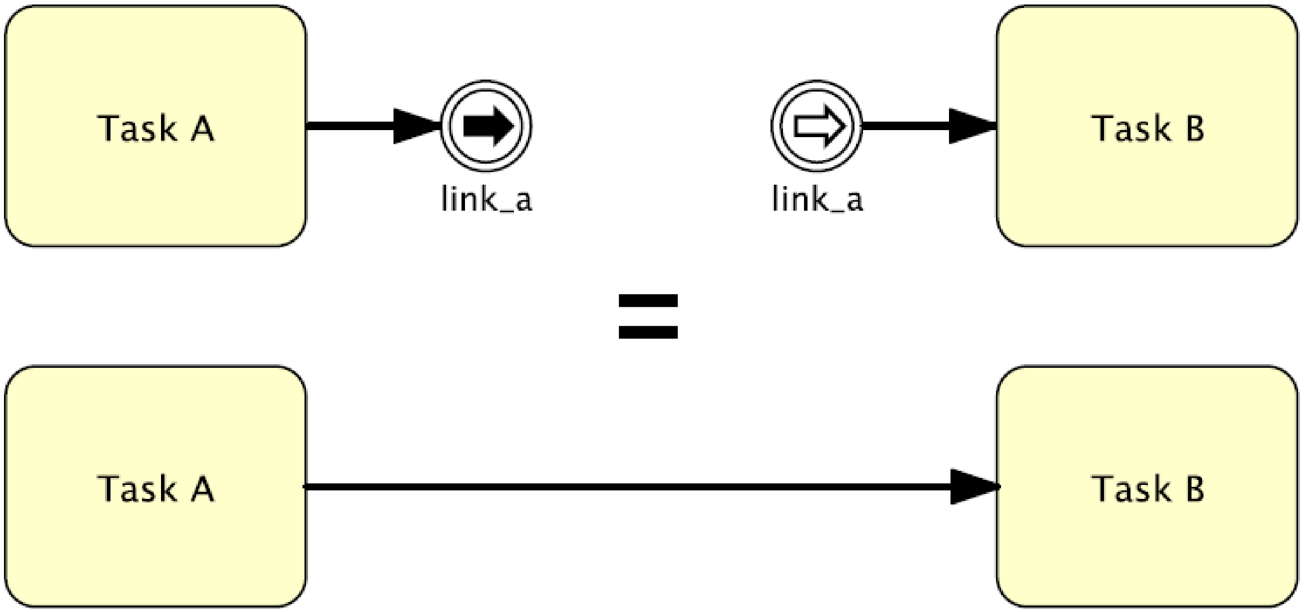
- Broadcasting: a signal may be processed at different places
- Model complex control flow, like synchronisation between subprocesses





Link Events

Link events allow for connecting different parts of a process model without sequence flow

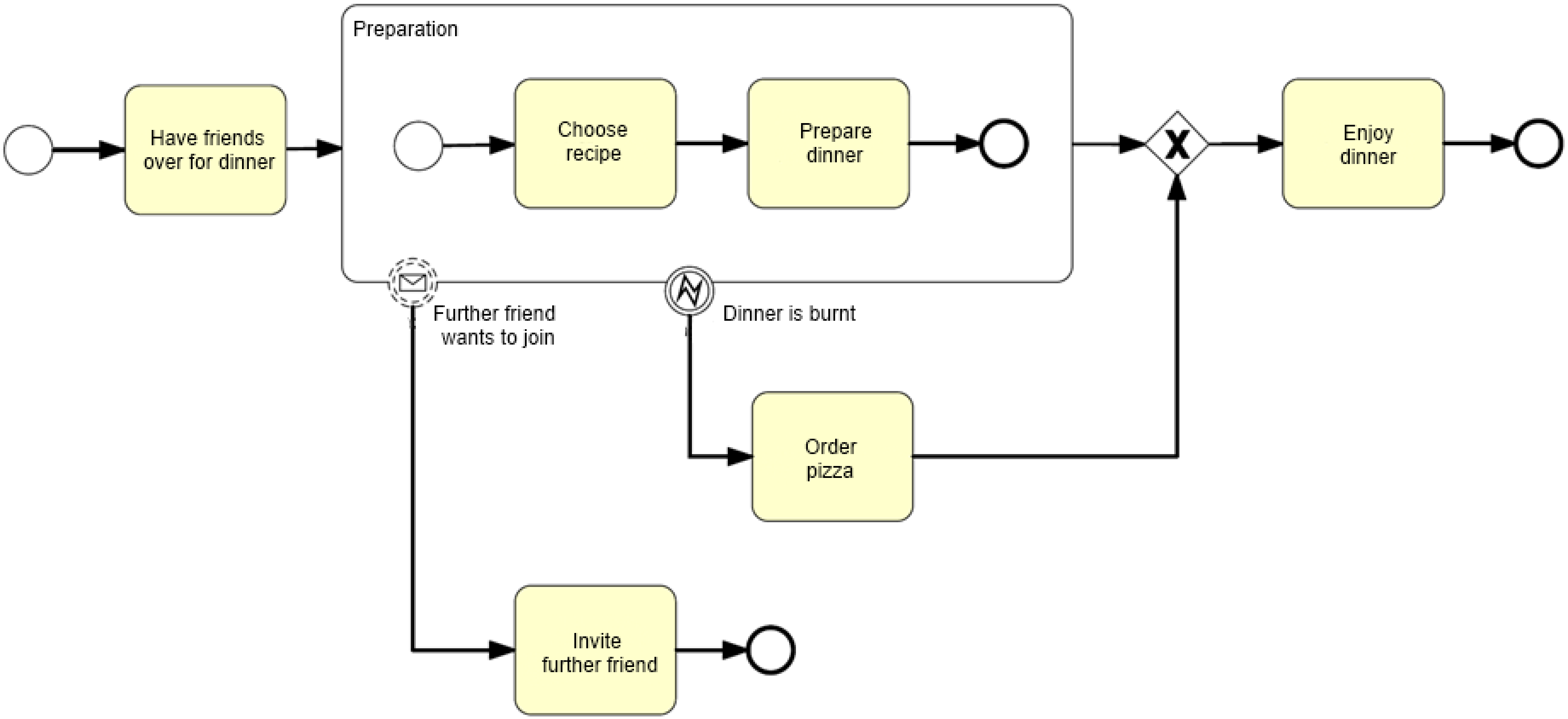


Interrupting vs. Non-Interrupting

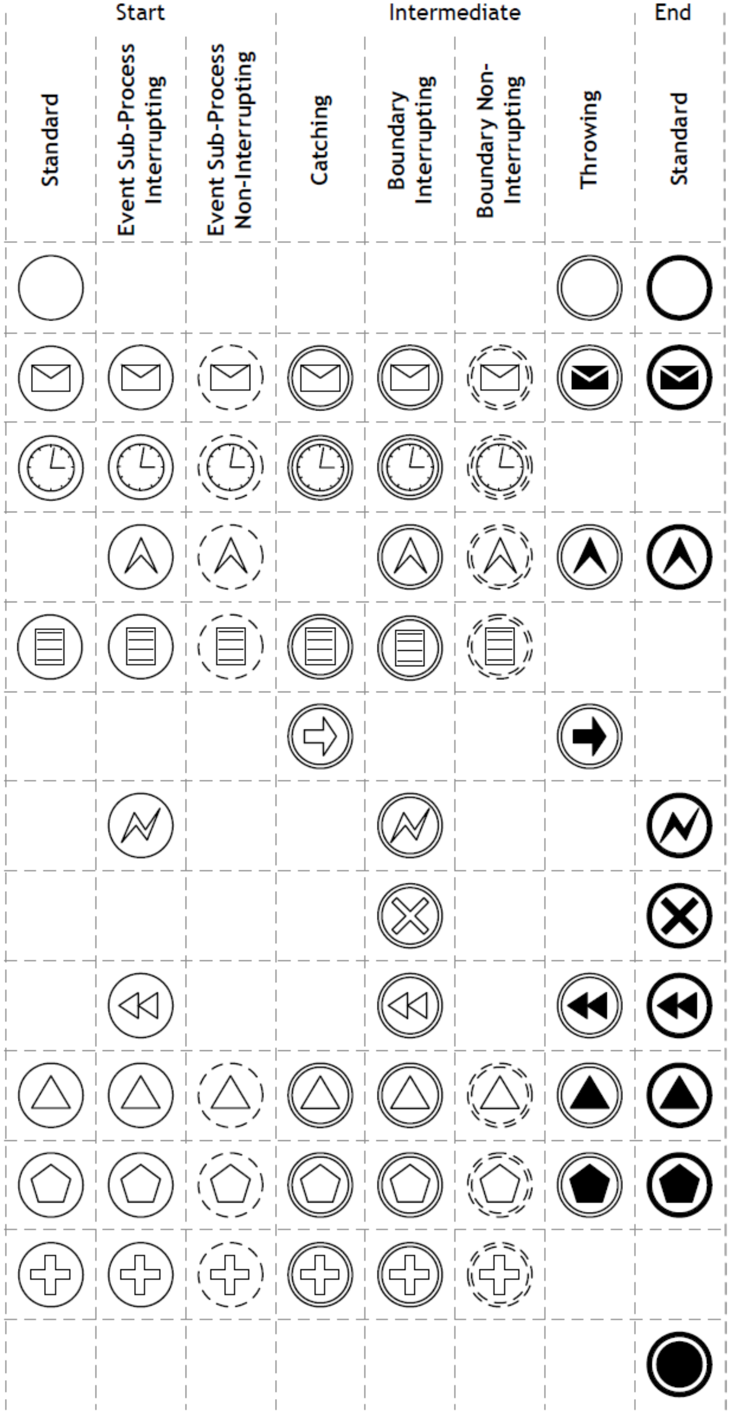
Catching boundary events catch event during execution of the parent activity

- 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
Preparation
Choose recipe
Prepare dinner
Enjoy dinner
Further friend wants to join
Dinner is burnt
Order pizza
Invite further friend

Events



Start
Intermediate
End
Standard
Event Sub-Process Interrupting
Event Sub-Process Non-Interrupting
Catching
Boundary Interrupting
Boundary Non-Interrupting
Throwing
Standard

None: Untyped events, indicate start point, state changes or final states.

Message: Receiving and sending messages.

Timer: Cyclic timer events, points in time, time spans or timeouts.

Escalation: Escalating to an higher level of responsibility.

Conditional: Reacting to changed business conditions or integrating business rules.

Link: Off-page connectors. Two corresponding link events equal a sequence flow.

Error: Catching or throwing named errors.

Cancel: Reacting to cancelled transactions or triggering cancellation.

Compensation: Handling or triggering compensation.

Signal: Signalling across different processes. A signal thrown can be caught multiple times.

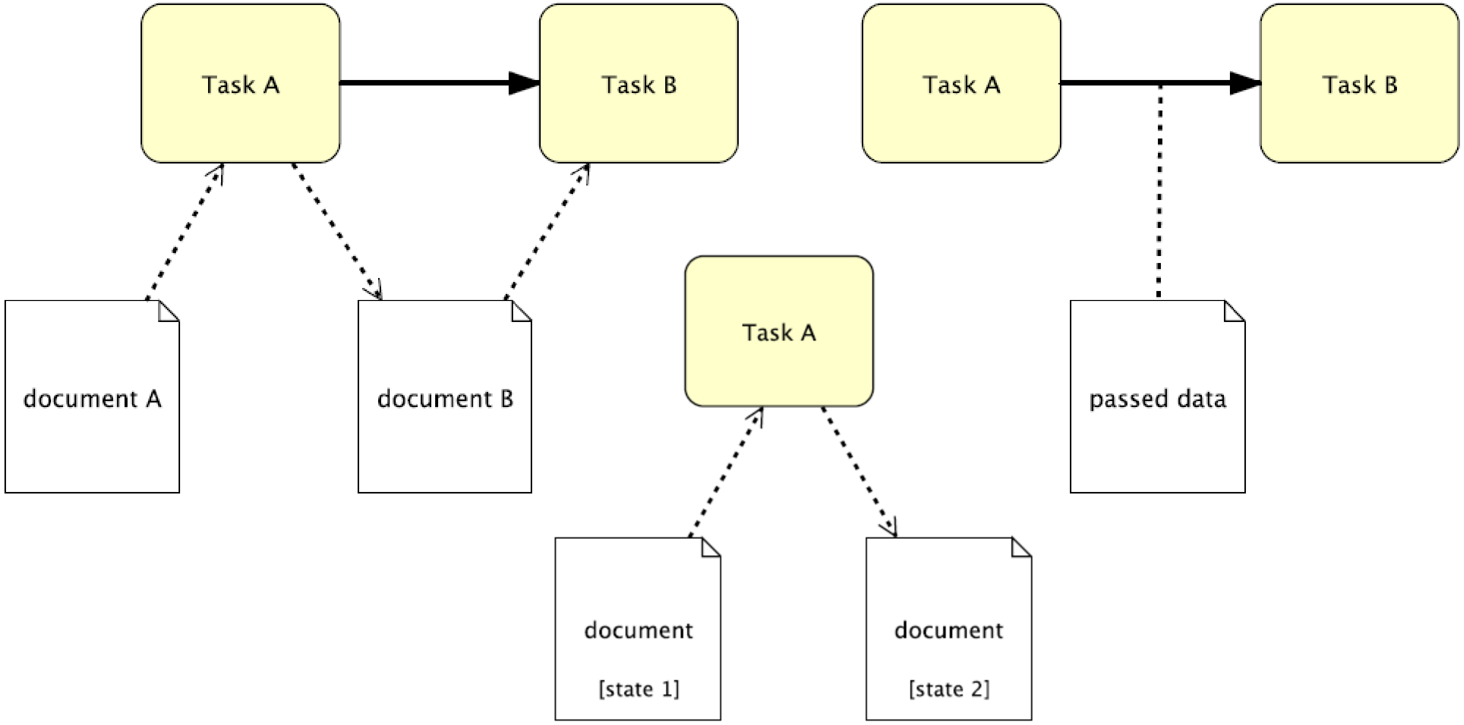
Multiple: Catching one out of a set of events. Throwing all events defined

Parallel Multiple: Catching all out of a set of parallel events.

Terminate: Triggering the immediate termination of a process.

Data in Processes

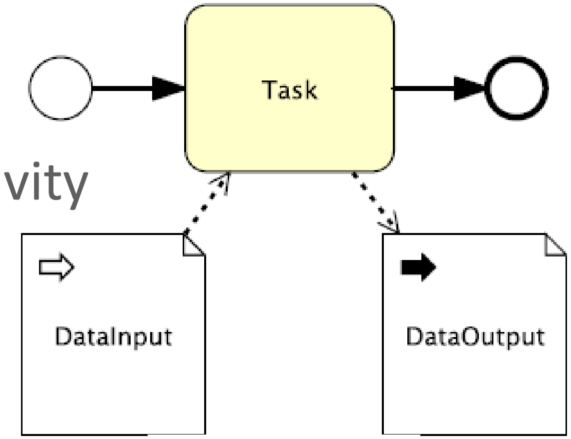
- Activities can read and write data objects, represented by directed associations
- Associating a data object to sequence flow is interpreted as data transfer
- Data objects have states that may change during processing



Input / Output of Processes

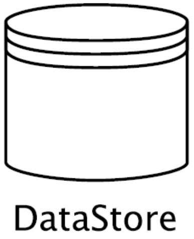
Symbols to represent input and output for process as a whole

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

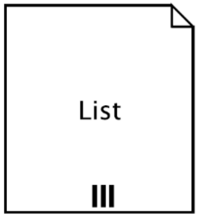


Data store describes a place where the process can read or write data

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

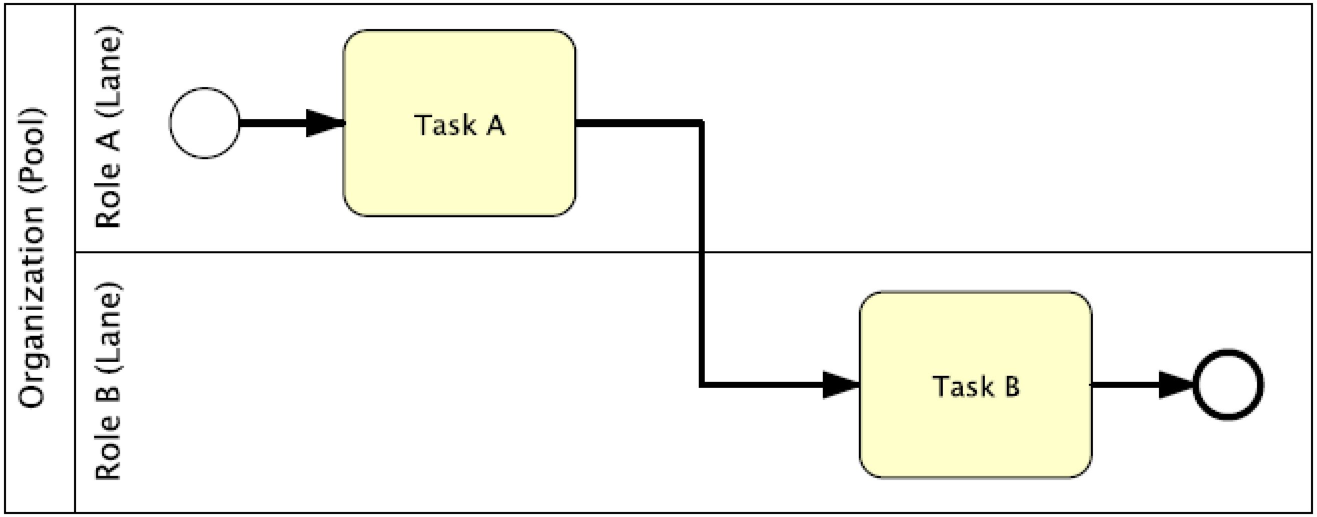


Collection data object represents multiple instances of a data object (type)



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

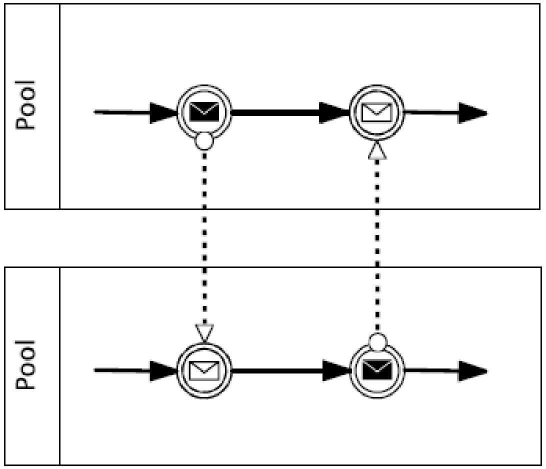
- Media (e-mail, mail, fax, telephone) is often abstracted

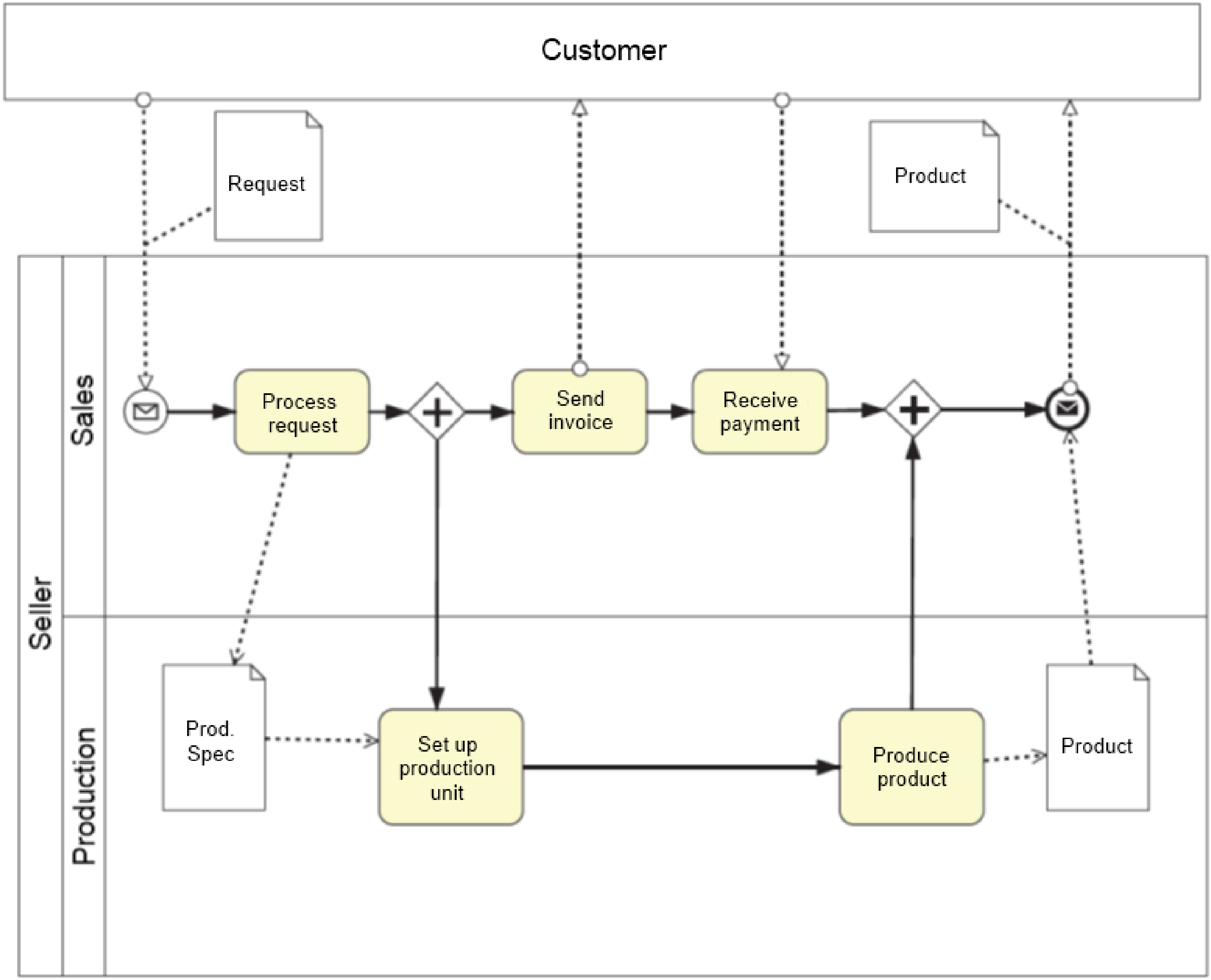
Sequence flow only for internal dependencies of an organisation

- 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

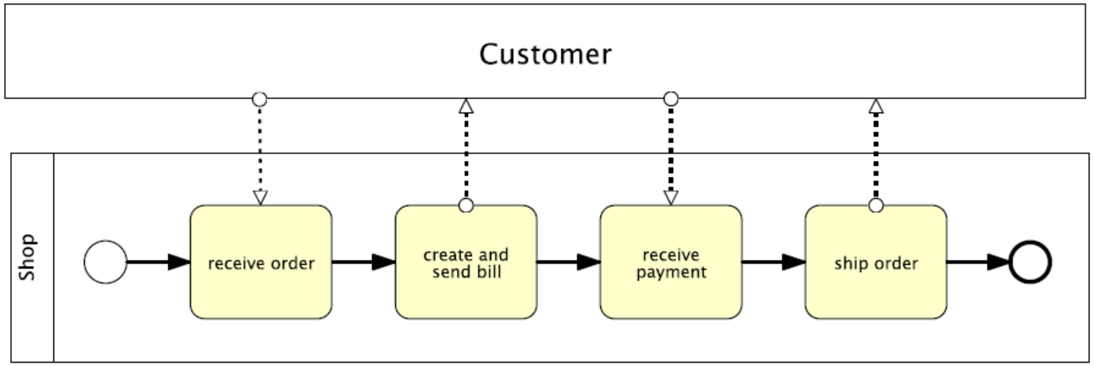


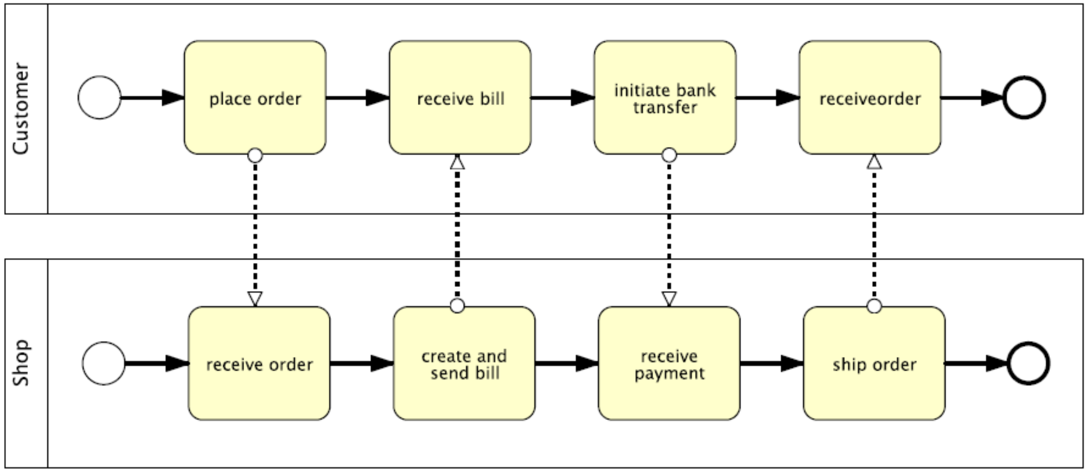

Customer
Request
Product
Seller
Sales
Production
Process request
Send invoice
Receive payment
Prod. Spec
Set up production unit
Produce product
Product

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

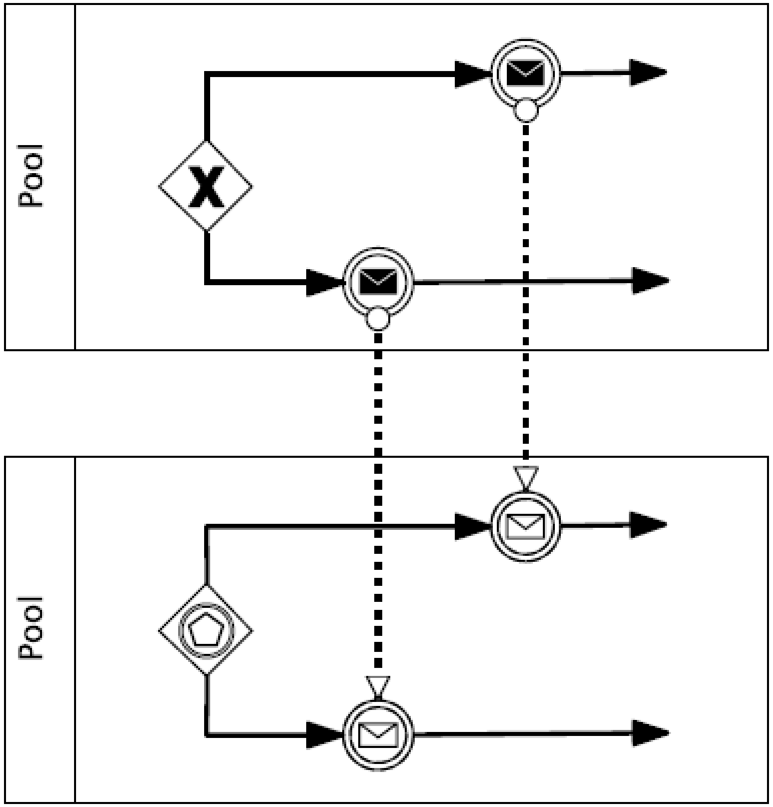




Event-based Gateway, again

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

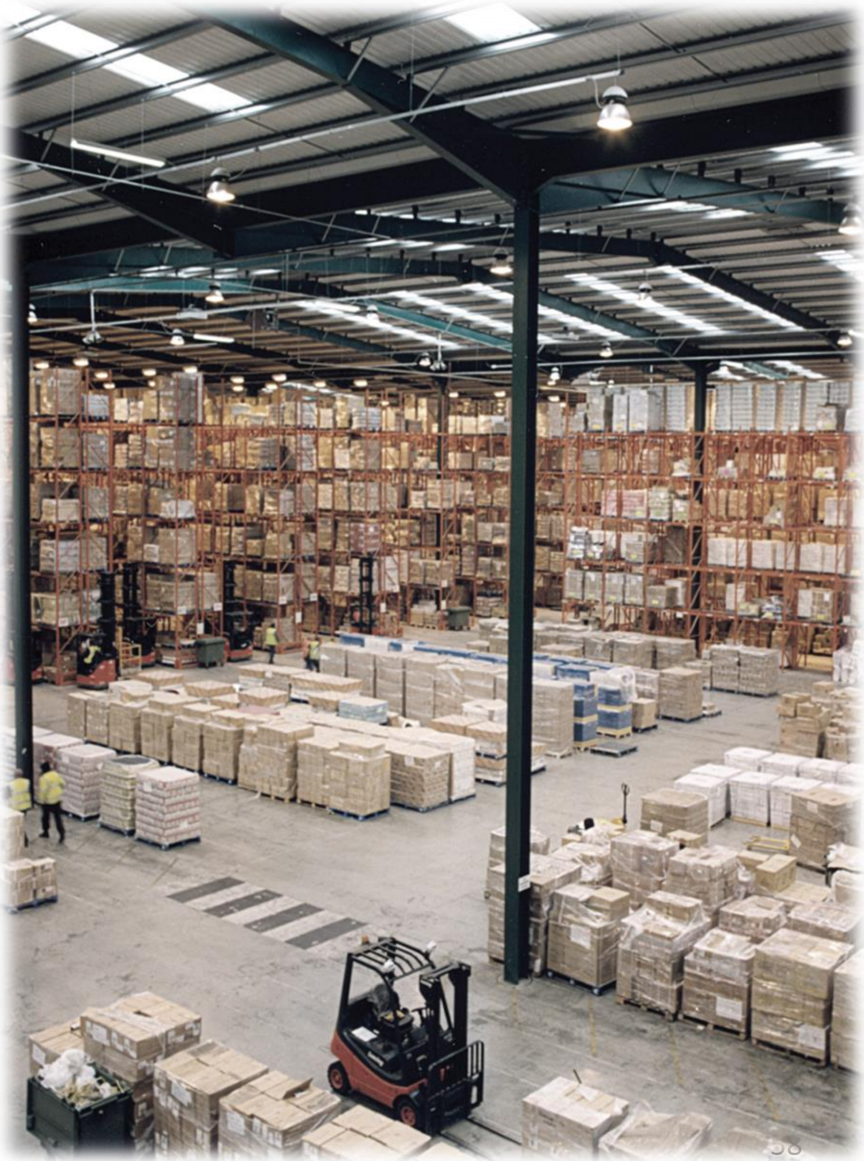
Scenario: Logistics

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

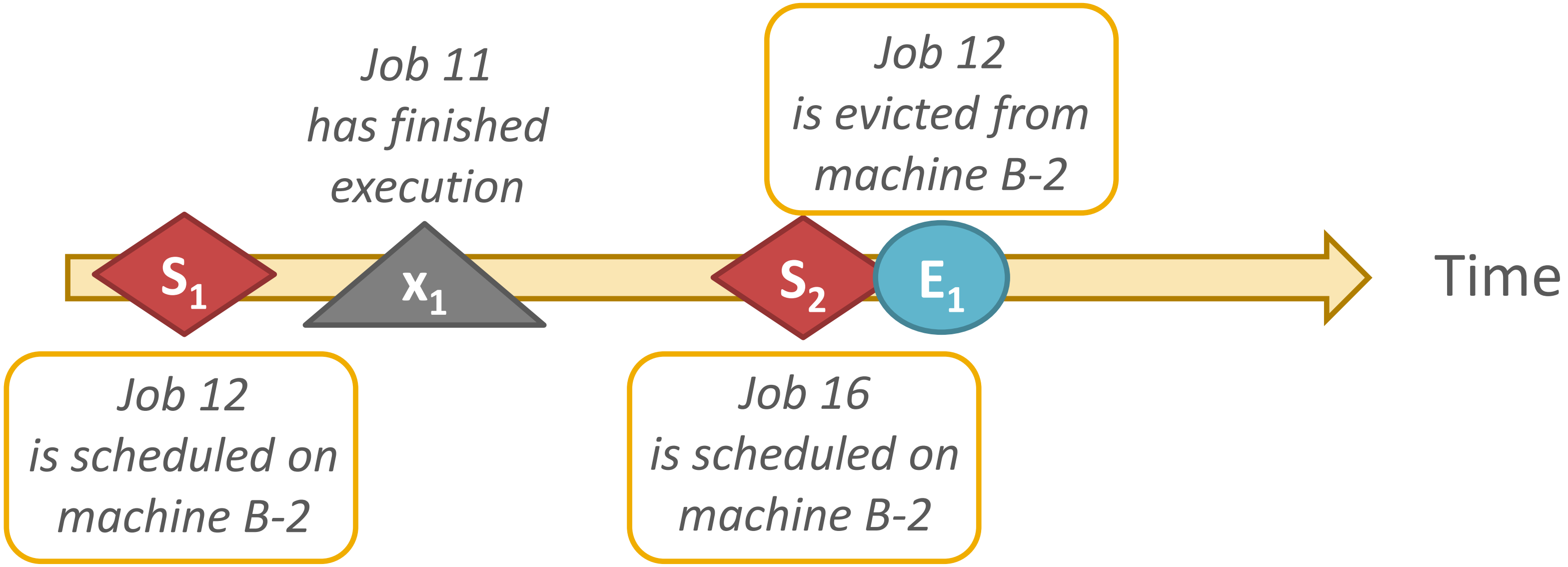
- Resource availability
- Machine utilisation
- Job scheduling

Detection of Complex Events

Observation:

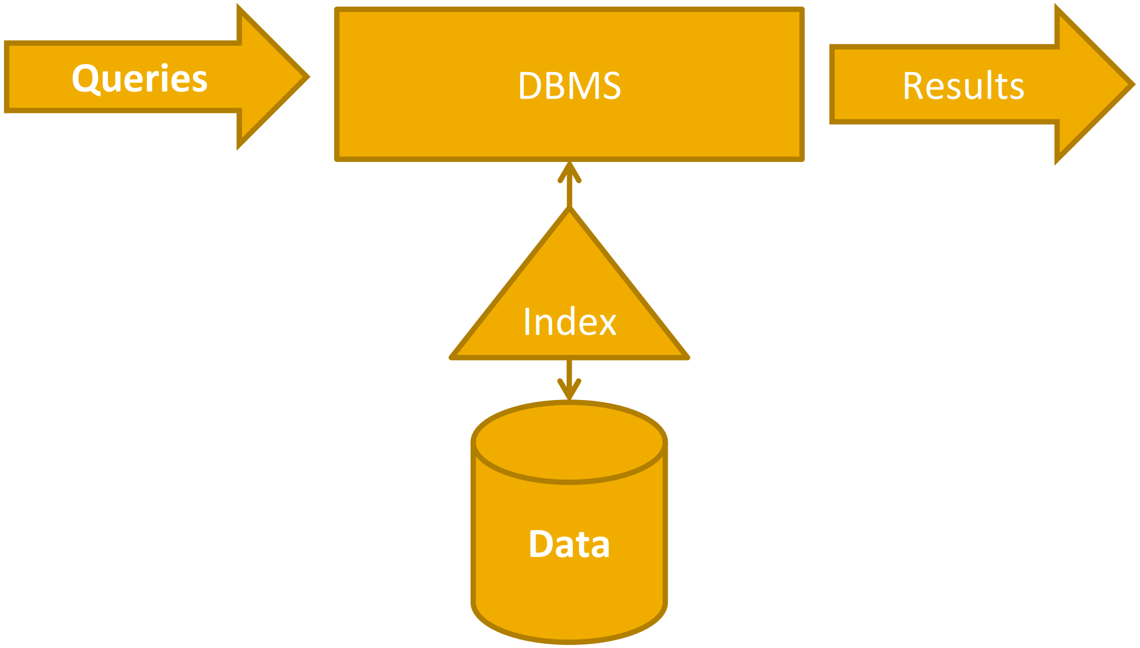
- Most events are not interesting
- New events supersede old events
- Ability to react to changing situations provides value

Derive complex events from simple events


Job 11 has finished execution
Job 12 is evicted from machine B-2
S_1
X_1
S_2
E_1
Time
Job 12 is scheduled on machine B-2
Job 16 is scheduled on machine B-2

Traditional Databases

Database Management System (DBMS):
Data relatively static but queries dynamic



Persistent relations

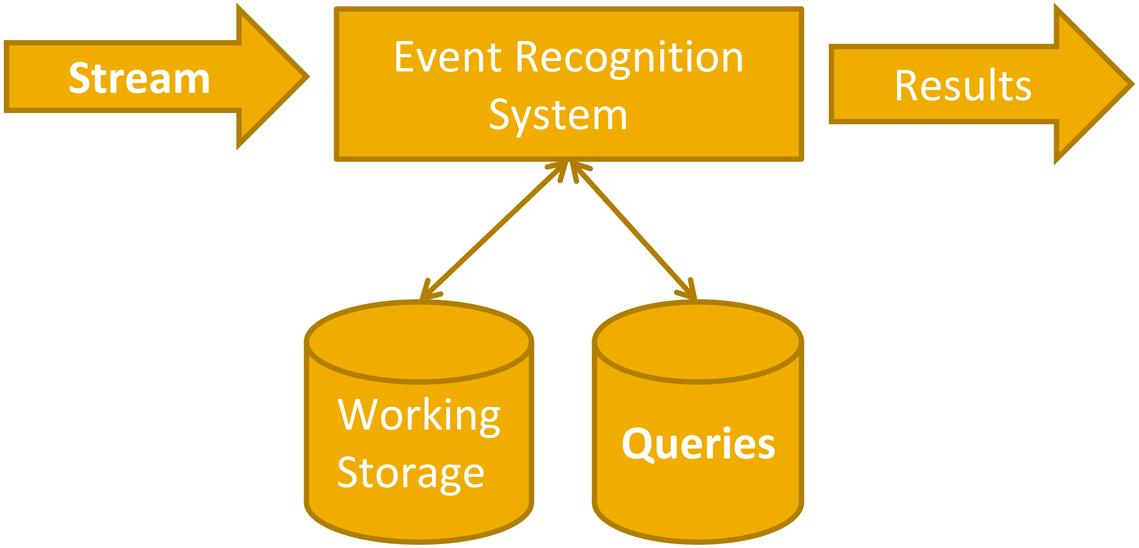
- Random access
- Low update rate
- Unbounded disk storage

One-time queries

- Finite query result
- Queries exploit (static) indices

Event Recognition System

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

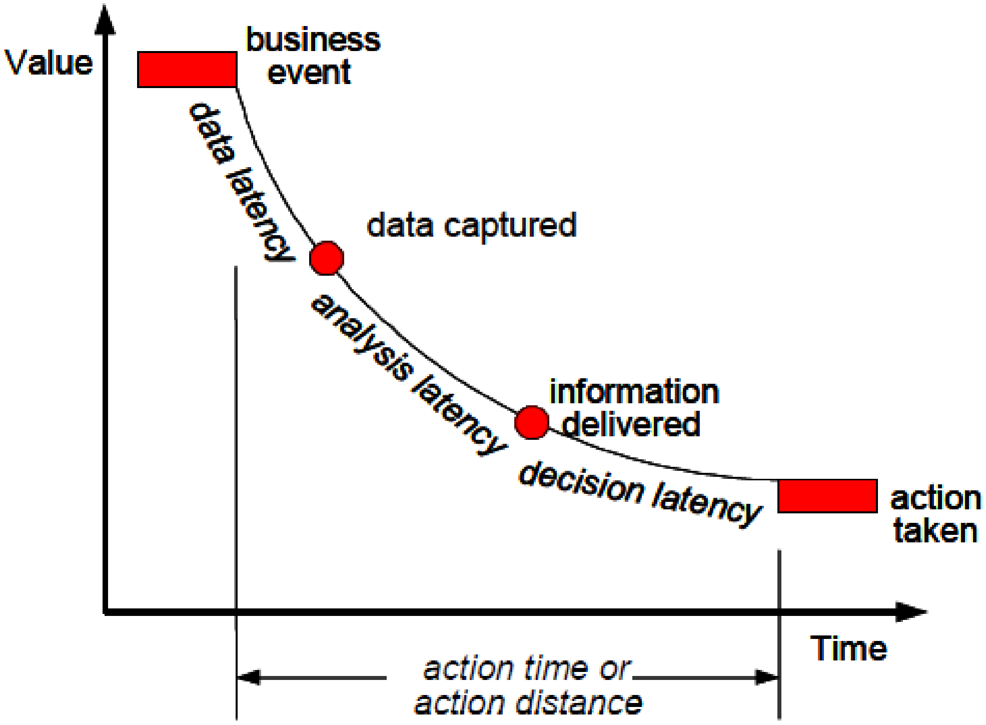
- Produce time-dependant result stream
- Indexing?

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

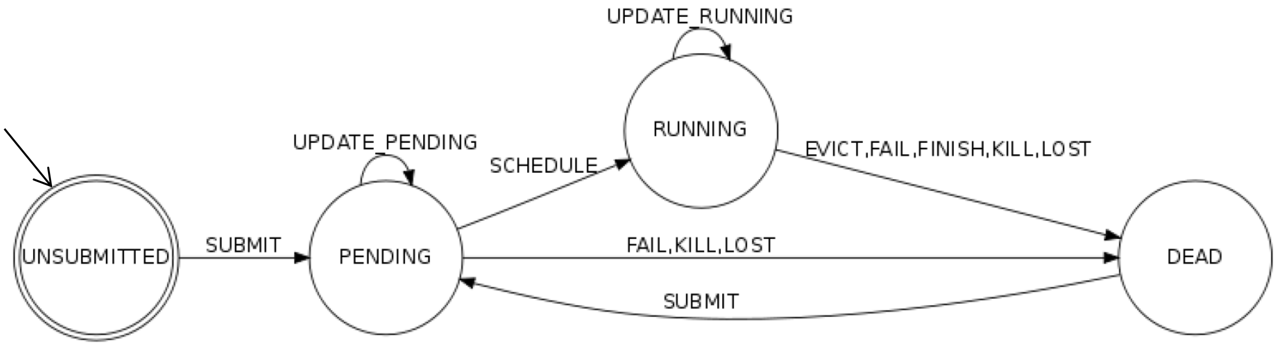
Events

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



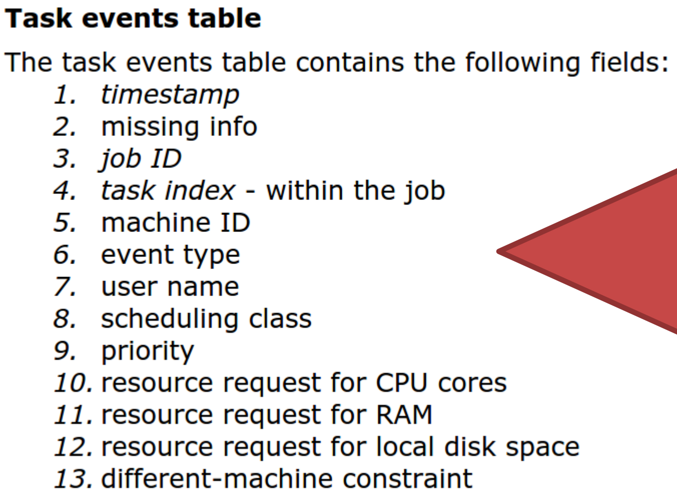
Event Types

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:



Task events table

The task events table contains the following fields:

1. *timestamp*
2. missing info
3. *job ID*
4. *task index* - within the job
5. machine ID
6. event type
7. user name
8. scheduling class
9. priority
10. resource request for CPU cores
11. resource request for RAM
12. resource request for local disk space
13. different-machine constraint

Schedule_1 (1444026993, -1, 239, 3, B-2, Schedule, rmalik,…)

Streams

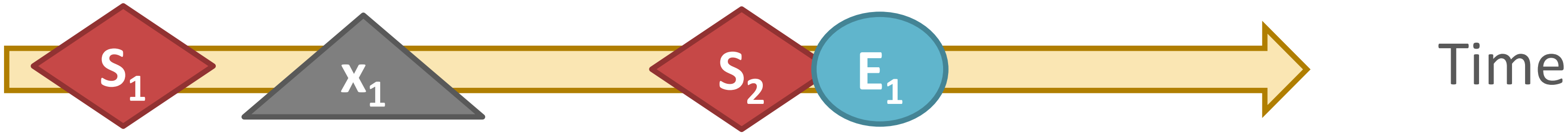
What is a stream?

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 & Ozsu (SIGMOD 2003)]

Data stream processing view: items are data tuples

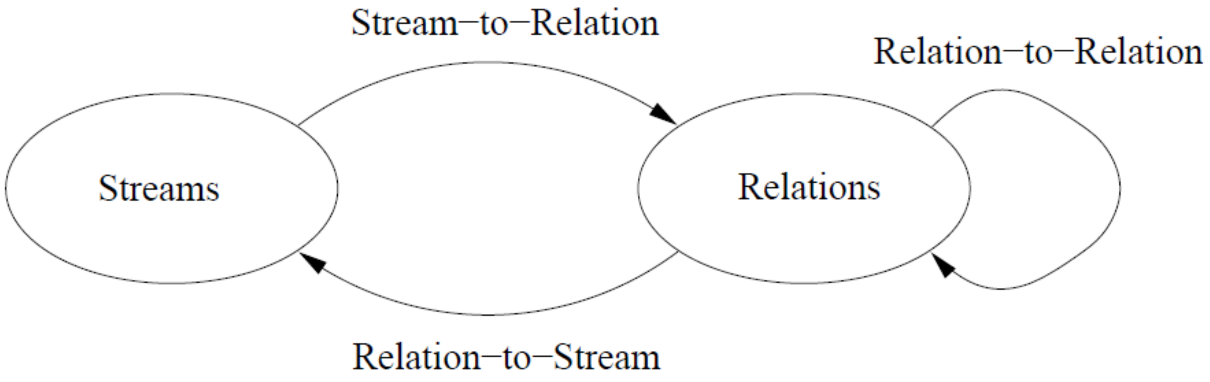
t_1	t_2	t_3	t_4	…					
time miss job ID	**time miss job ID**	**time miss job ID**	**time miss job ID**	**time miss job ID**	**time miss job ID**	**time miss job ID**	**time miss job ID**	**time miss job ID**	**time miss job ID**

Complex event processing view: items are typed events



Data Stream Processing Languages

Idea:
Lift the relational model for queries to streams



Implicit streaming operator

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

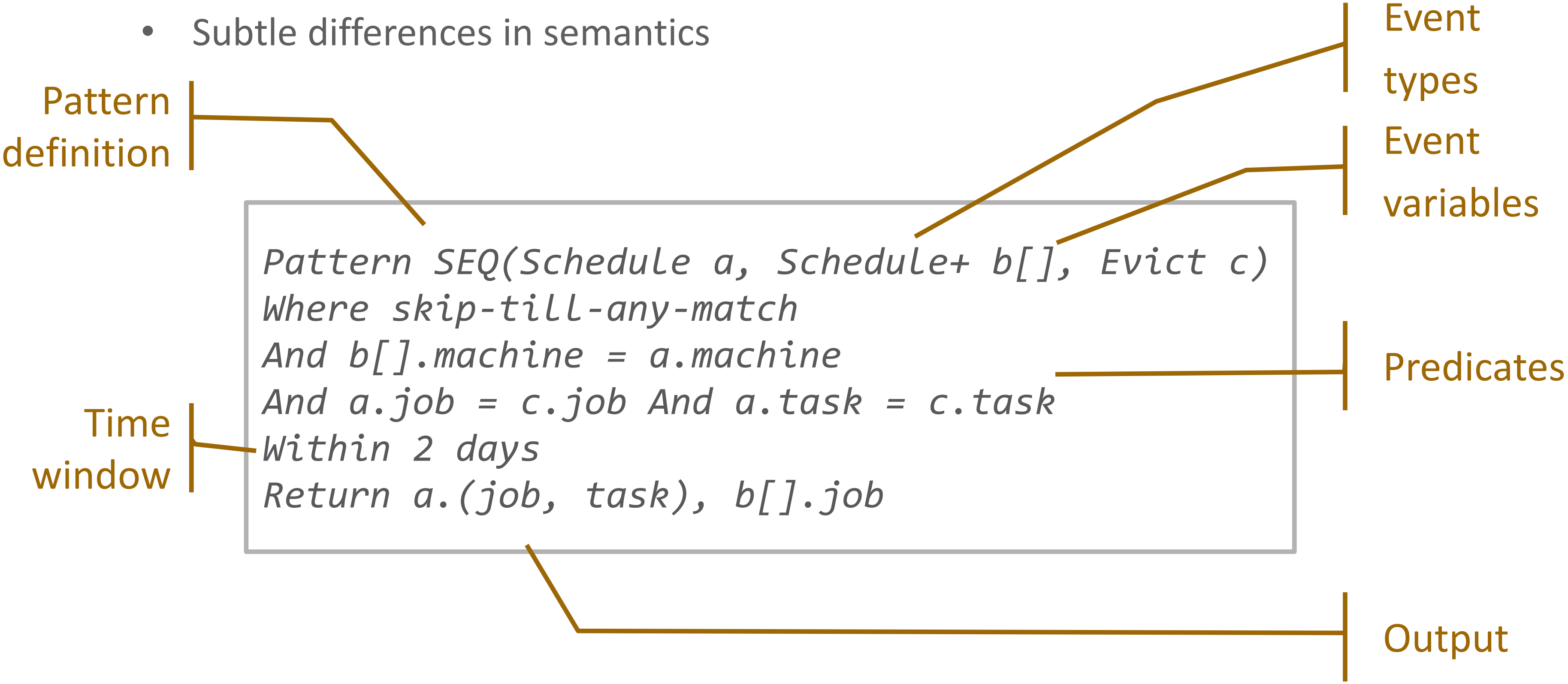
Predicates

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

Timeline

Tentative Dates	Phases	Meeting	Deliverables
19/04/2016	Organisation and planning	all	
until 10.05.2016	Domain and Requirement Analysis, Projectplanning		Spec. & projectplan
	Design: Interfaces, inter-team data structures, file formats and possible test cases	self-organised	File format & test cases
	Design: Intra-team data structures, architecture, algorithms	self-organised	
31/05/2016	Inter-team presentation of design	all	System Design
	First implementation / prototype		
14/06/2016	Inter-team presentation and mutual testing of prototype I	all	
	Intermediate debugging		
05/07/2016	Inter-team presentation and mutual testing of prototype II	all	
	Final debugging		
19/07/2016	Final presentation	all	Final implementation
TBD	Project closing		Final documentation

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