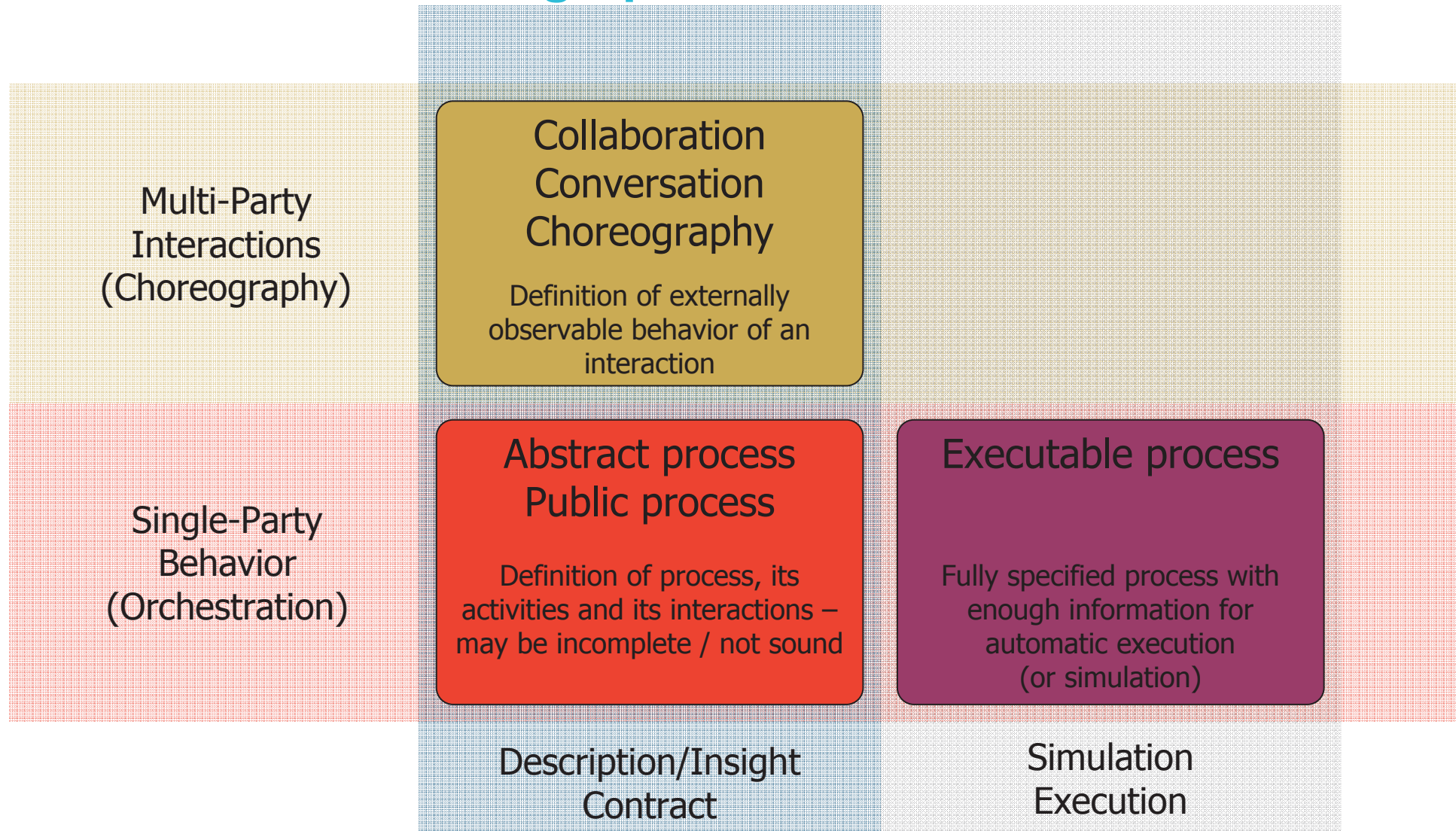


## BPMN

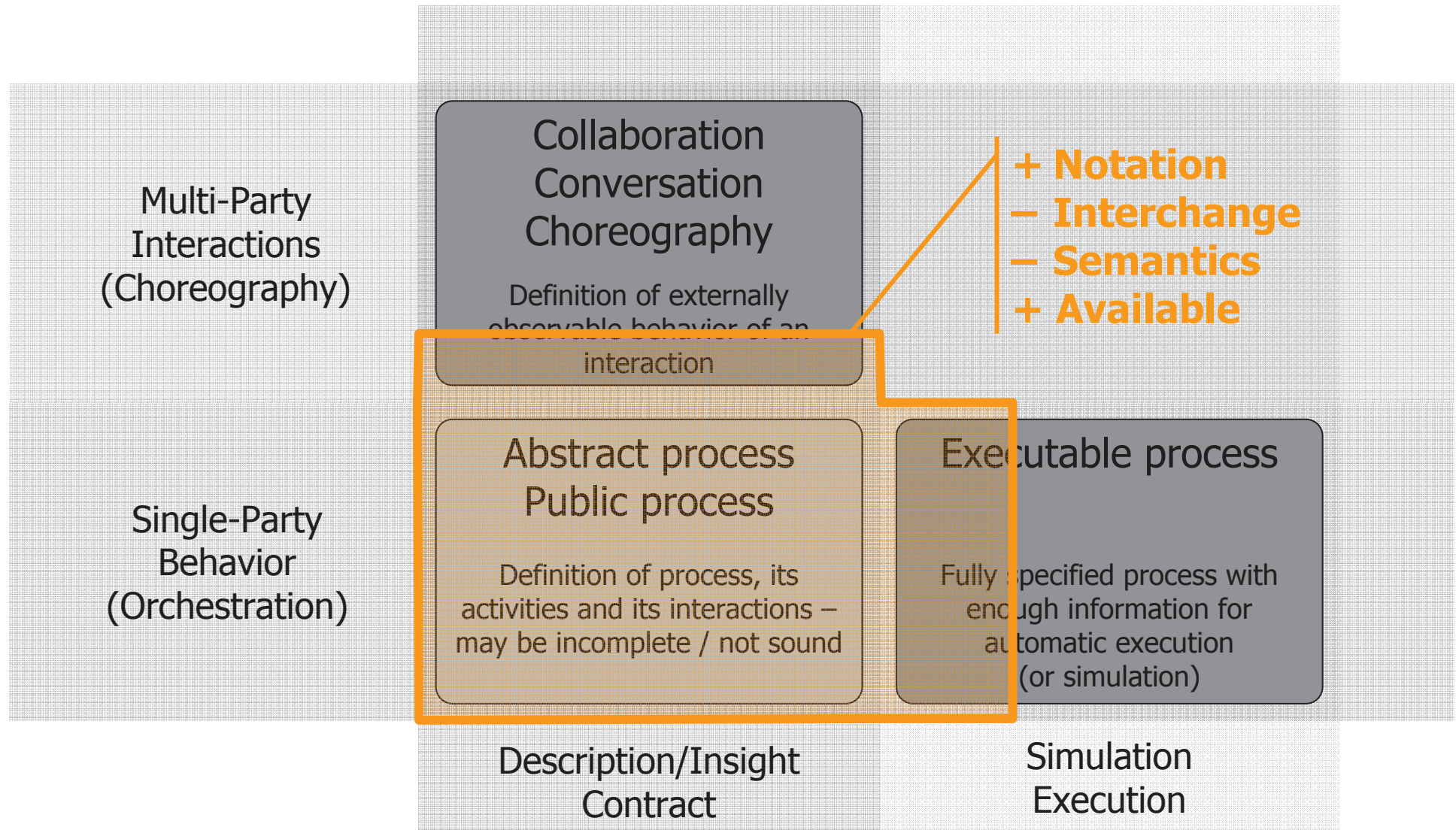


- BPMN 1.x
  - Business Process Modeling Notation
  - Current version 1.2 (January 2009)
    - <http://www.omg.org/spec/BPMN/1.2/>
- BPMN 2.0
  - Business Process Model and Notation
  - In RFP process

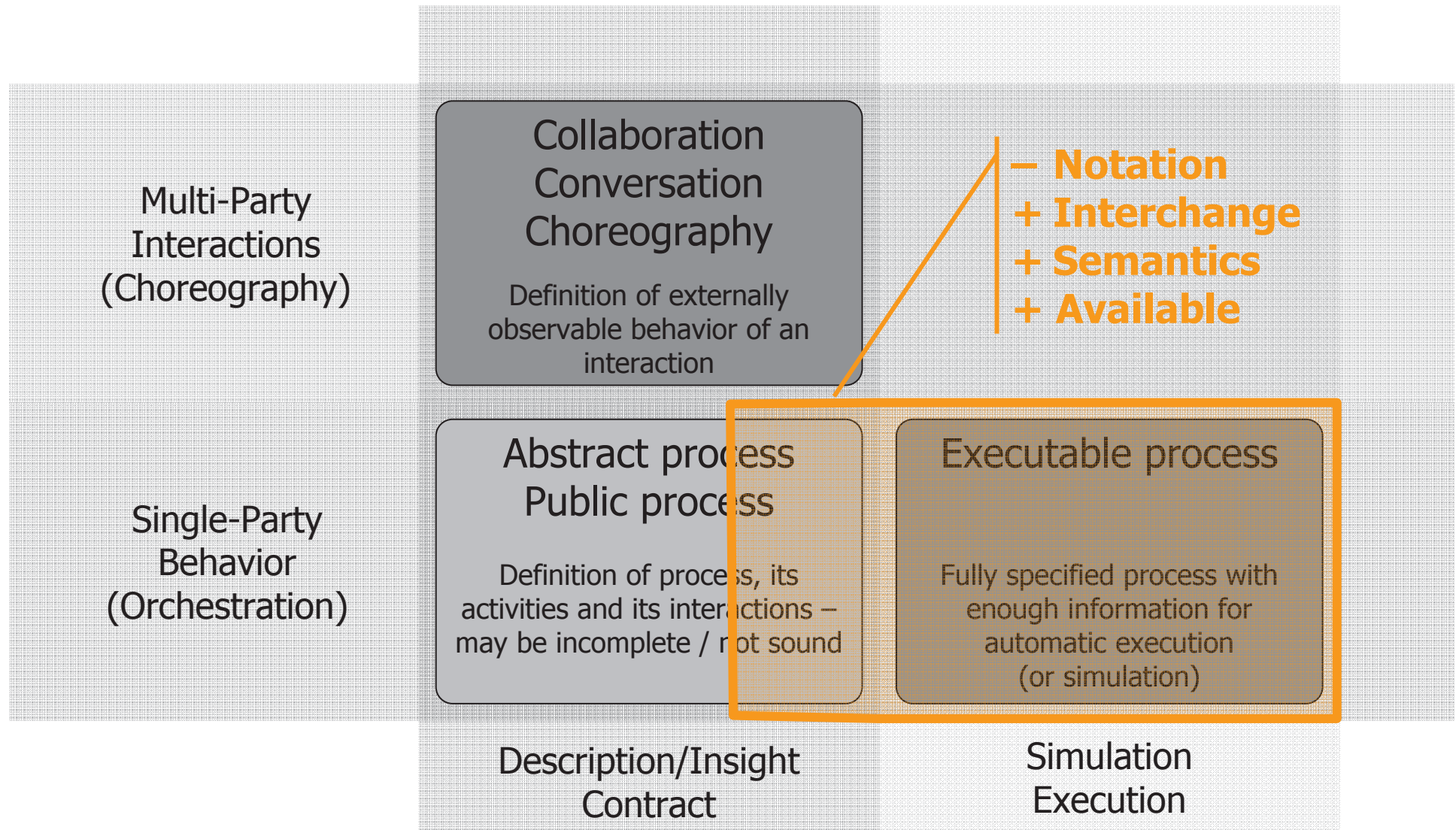
## The BPM Modeling Space



## BPMN 1.1

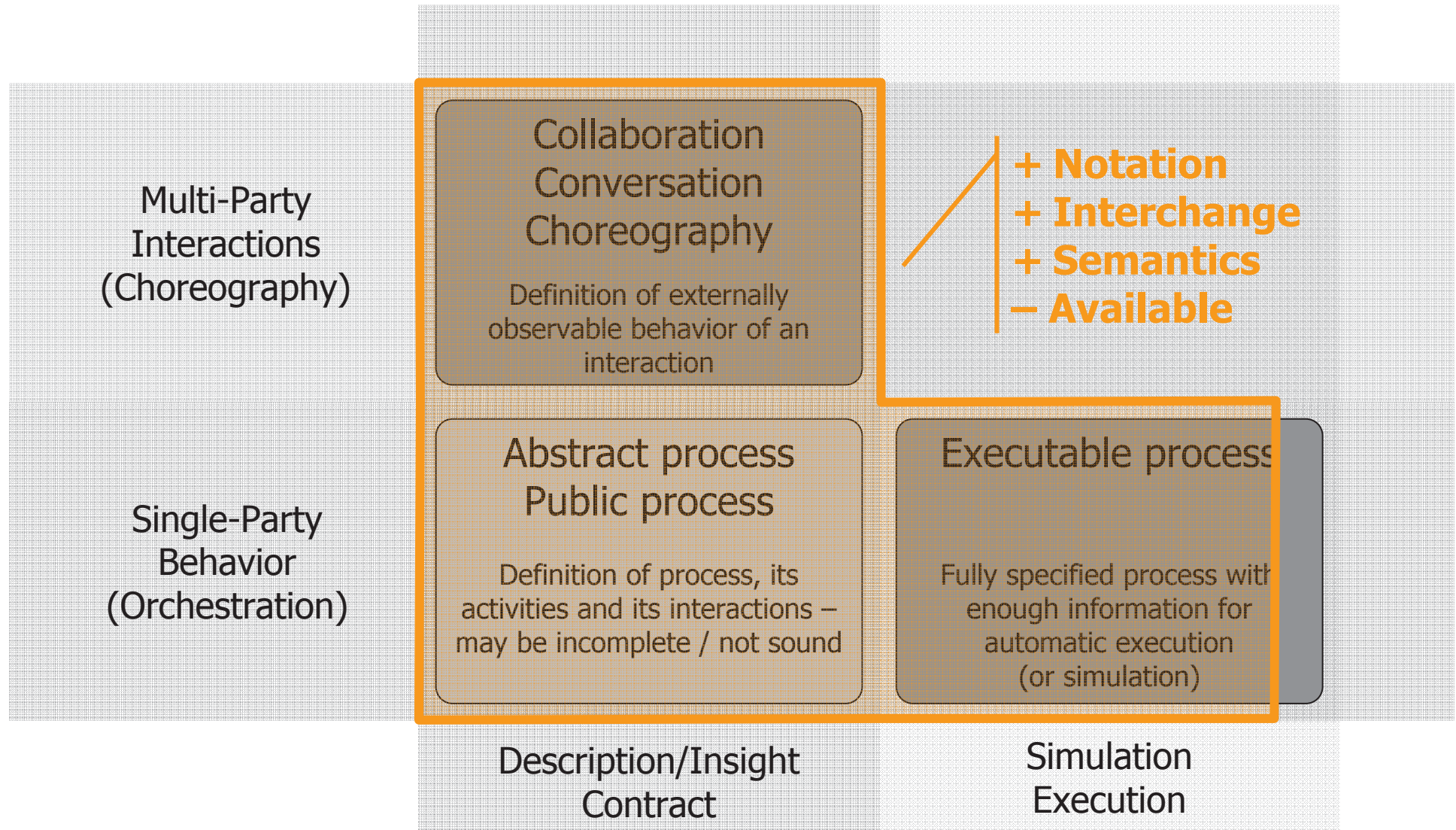


## WS-BPEL 2.0

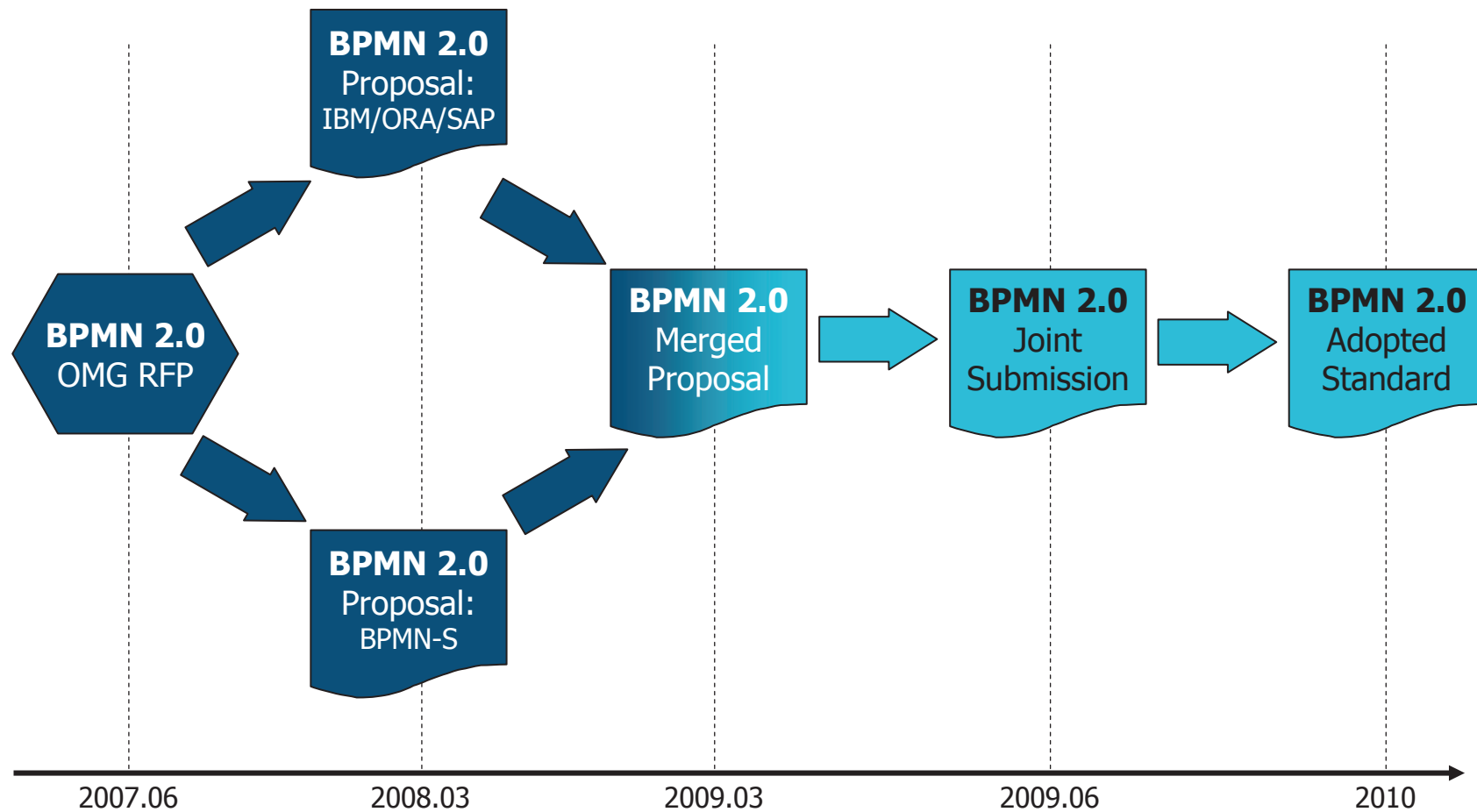




## BPMN 2.0



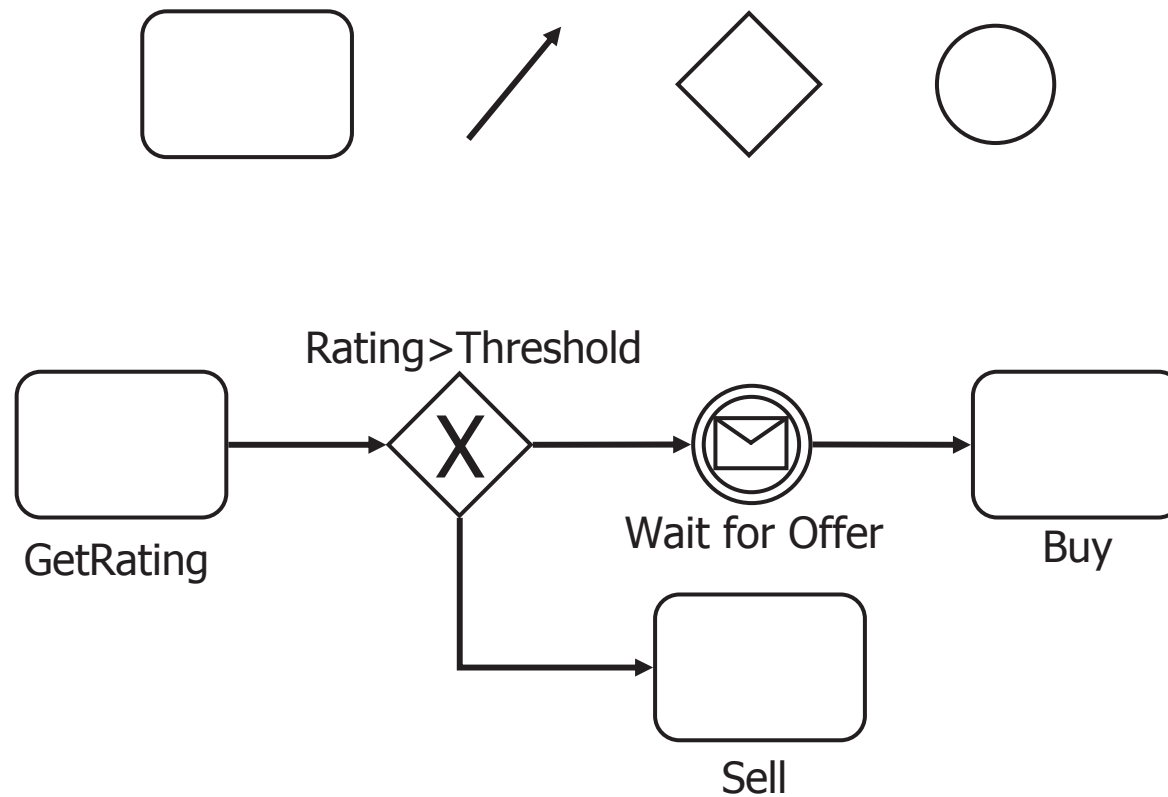
## BPMN 2.0 Timeline



## BPMN 2.0 Conformance Classes

- Process Modeling Conformance
  - Support modeling of processes and collaborations
  - Corresponds roughly to the BPMN 1.1 set of capabilities
- Choreography Modeling Conformance
  - Support modeling of choreographies
- Process Execution Conformance
  - Support execution of BPMN processes according to defined execution semantics
- BPEL Process Execution Conformance
  - Support execution of BPMN processes by mapping to BPEL as per the defined BPMN-BPEL mapping.

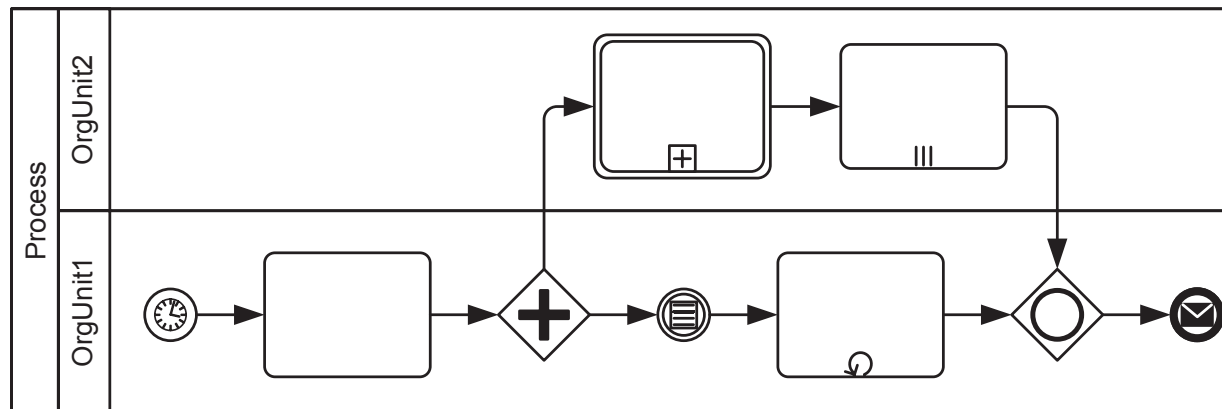
# Business Process Modeling Notation






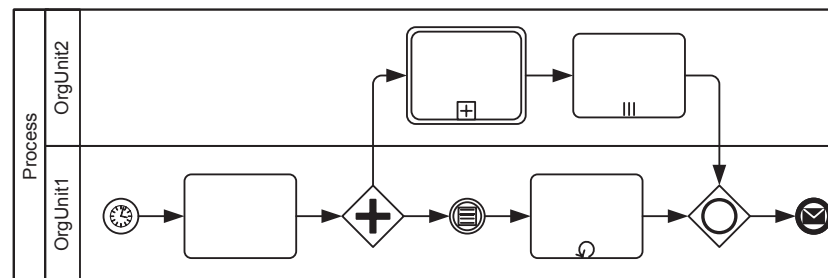
## Overall Structure of a BPMN Process

- Lanes
- Activities
  - Structured
  - Atomic → Tasks
- Events
  - Inbound
  - Outbound
- Ordering
  - Sequence Flow
  - Gateways
  - “Adornments”



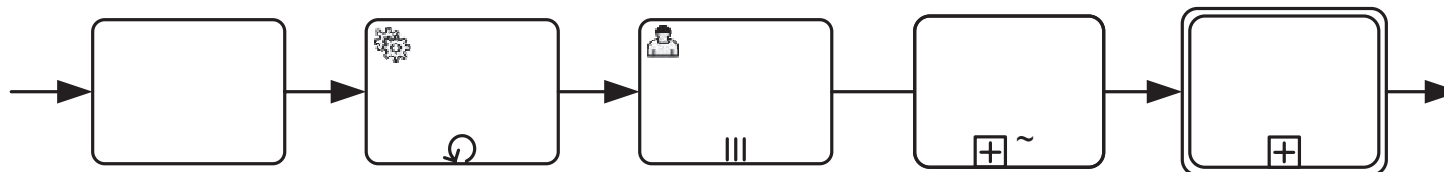
## Lanes

- Sub-partition of a Process
- Used to organize and categorize activities
- Typically represent an organizational unit of an enterprise
- Lanes are a visual construct only
-  Can be used to structure visualization of activities by *any* suitable attribute

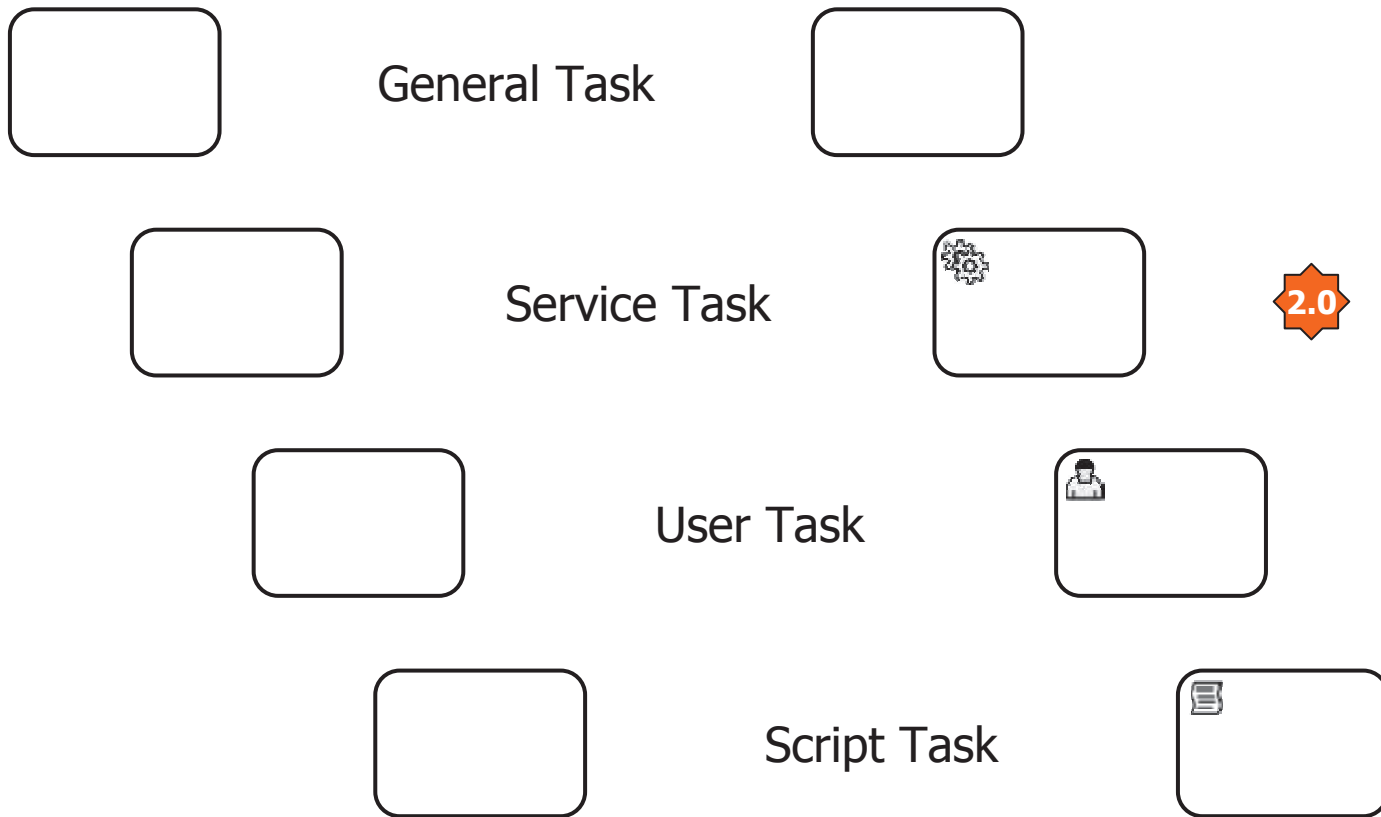


## Activities

- Represents work performed in a process
- Atomic activities – Tasks
  - General, service, human, script, ...
- Compound activities
  - Inline (sub-process) or referenced (2.0 call activity)
- Decorations for looping, multiple instantiation, ad-hoc execution



# Tasks



## Tasks (continued)



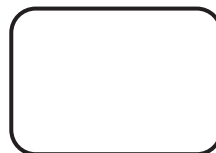
Receive Task



Send Task



Manual Task



Business Rule  
Task







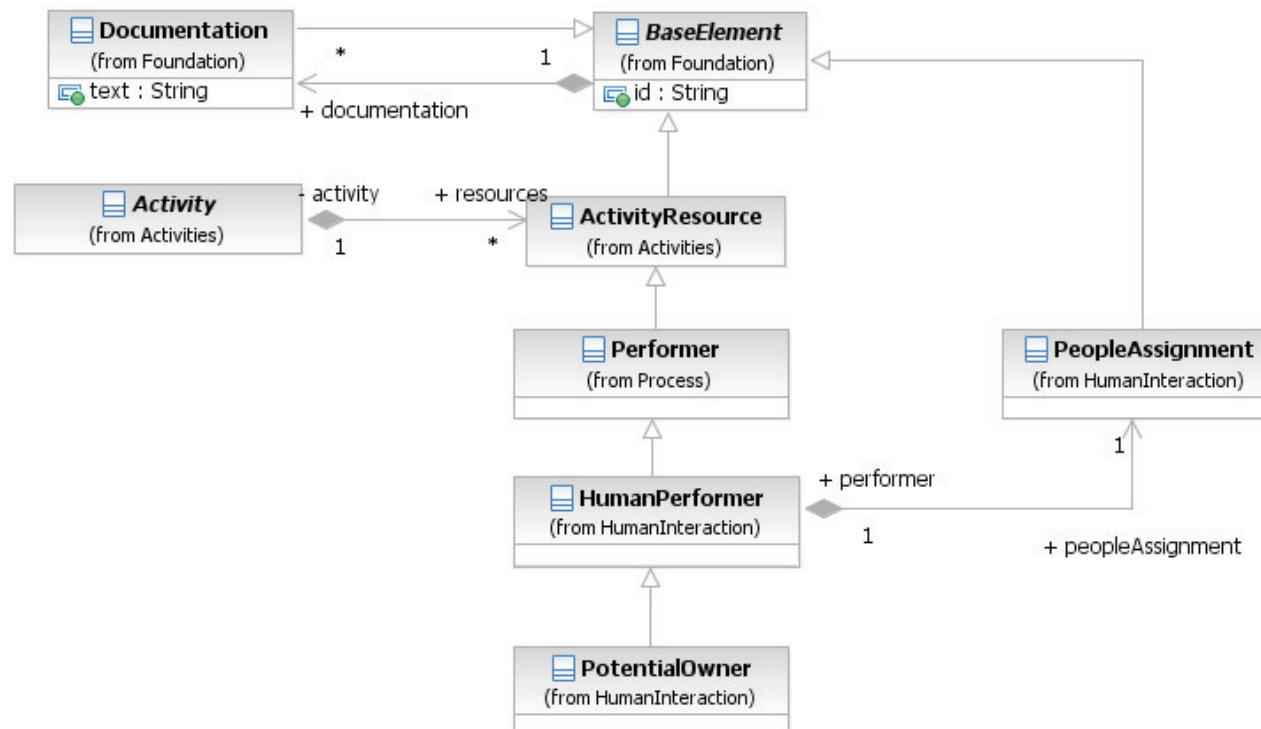
## Service Model

- BPMN 2.0 adds a service model to capture service-related information
  - Interface
  - Operation
  - Endpoint
- Captures WSDL 1.1 style information
  - Allows referencing service information from activities and processes
- Alignment with SoaML in progress
  - Similar relationship as BPEL and SCA

2.0

## People Assignment Model (1)

- Human Performer identifies human resources
- Potential Owner identifies responsible people
- People Assignment to link to actual people





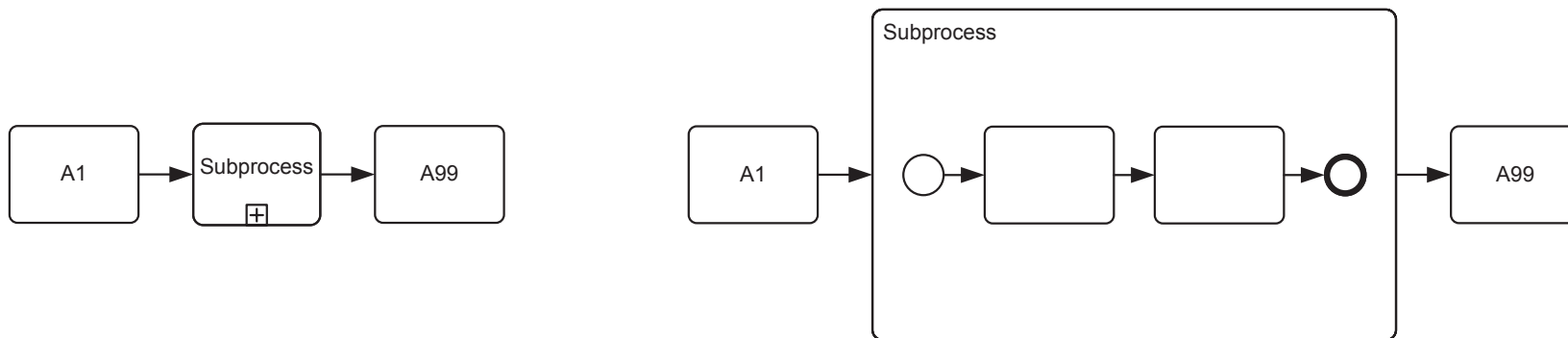
## People Assignment Model (2)

### Multiple forms of People Assignment

- Literal – list of user IDs
- Expression – eval at runtime, returns list of user IDs
  - Example: `getActivityInstanceAttribute("approve", "owner")`
- Process Role – abstract group of people
  - Named, possibly parameterized
    - Example: `salesManager(region)`
  - Evaluated at runtime, binding arguments from process context
    - Example: `salesManager("CHE") → "Konrad Schmidt"`

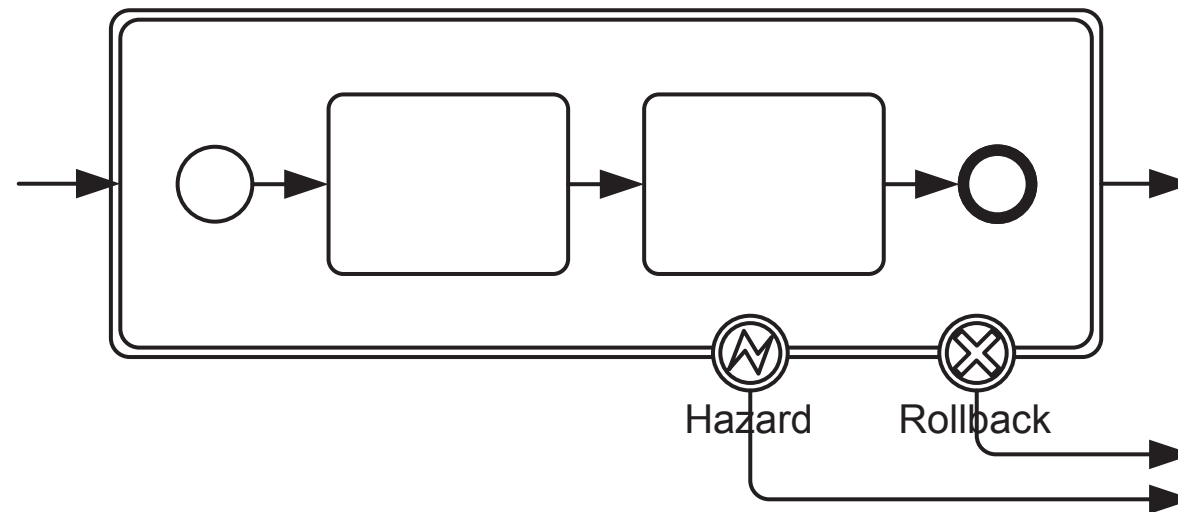
## Subprocess

- Activity whose internal structure is a flow
  - Compound activity
  - Recursive aggregation to structure processes
- Can be collapsed/expanded
  - Affects the visual model only
- Provides scoping boundary
  - For data, exception handling, transactions, ...



## Transaction Subprocess

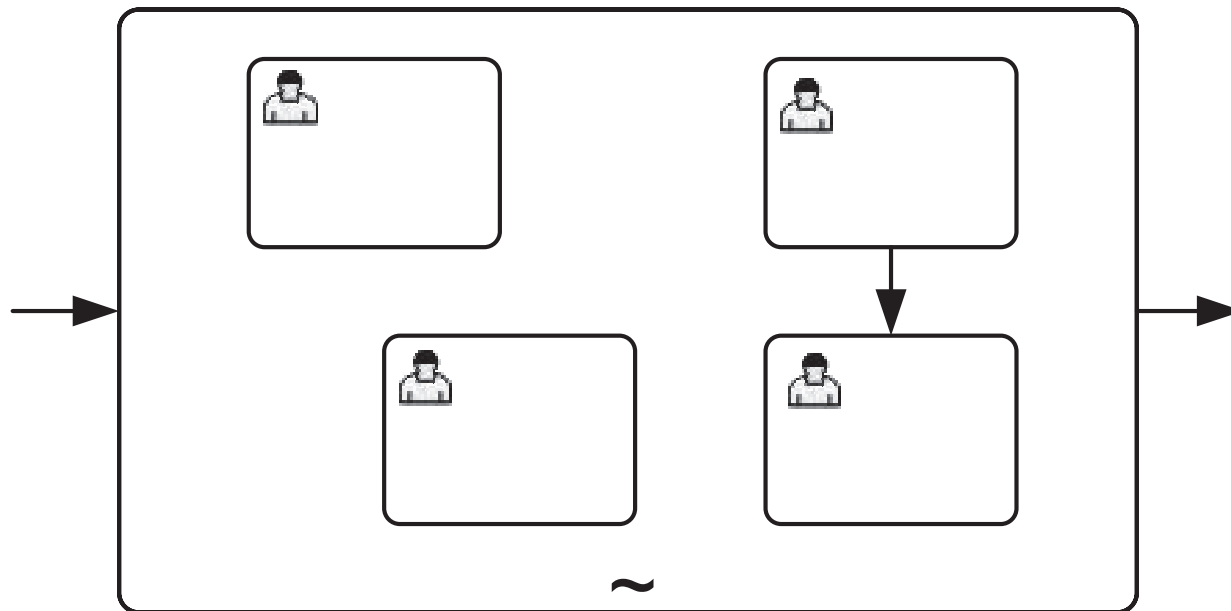
- A subprocess with transactional behavior of its activities, maintained through a coordination protocol





## Ad-hoc Subprocess

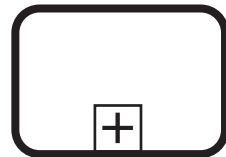
- Used for collaborative processes
- Contained activities can be performed in any order
  - As decided by assigned people



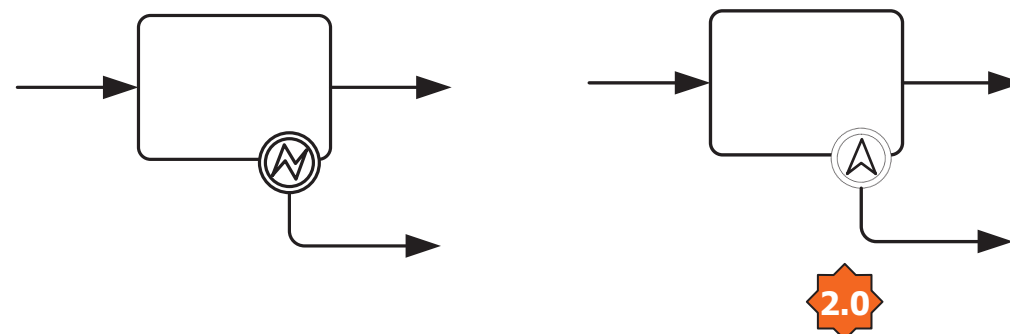
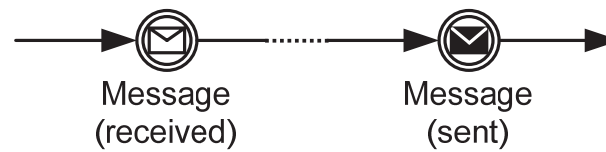


## Call Activity

- Activity to call another process, or global task
  - Process/task-aware call with life-cycle coupling
  - Notation: Thick boundary line



# Events



## Event Types (1)

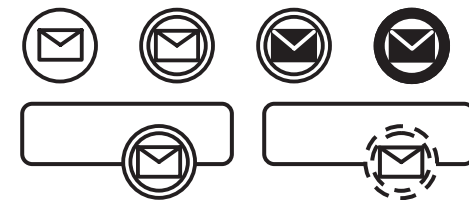
- General / unspecified

- Event details not known or immaterial



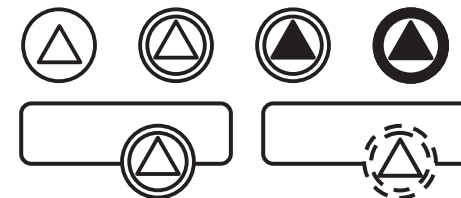
- Message

- A message is received or sent



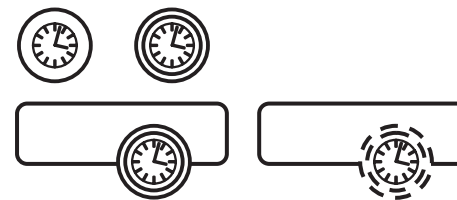
- Signal

- A signal is received or sent



- Timer

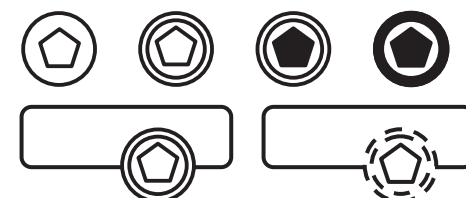
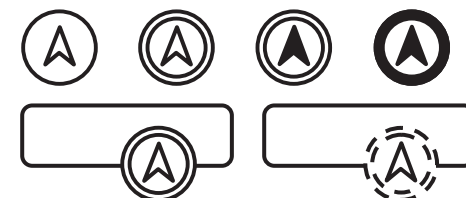
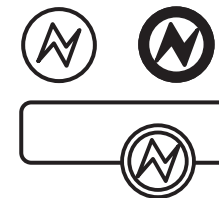
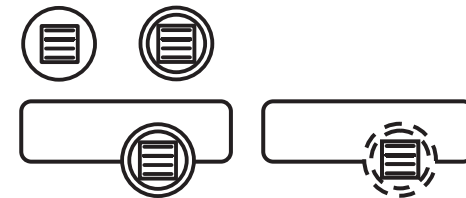
- A timer expires



## Event Types (2)

- Conditional
  - A condition becomes true
- Error
  - An error is thrown or caught
- Escalation
  - An escalation is thrown or caught
- Multiple
  - A combination of other events is thrown or caught

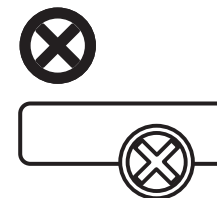
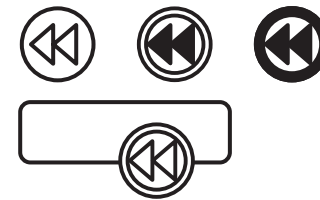
2.0





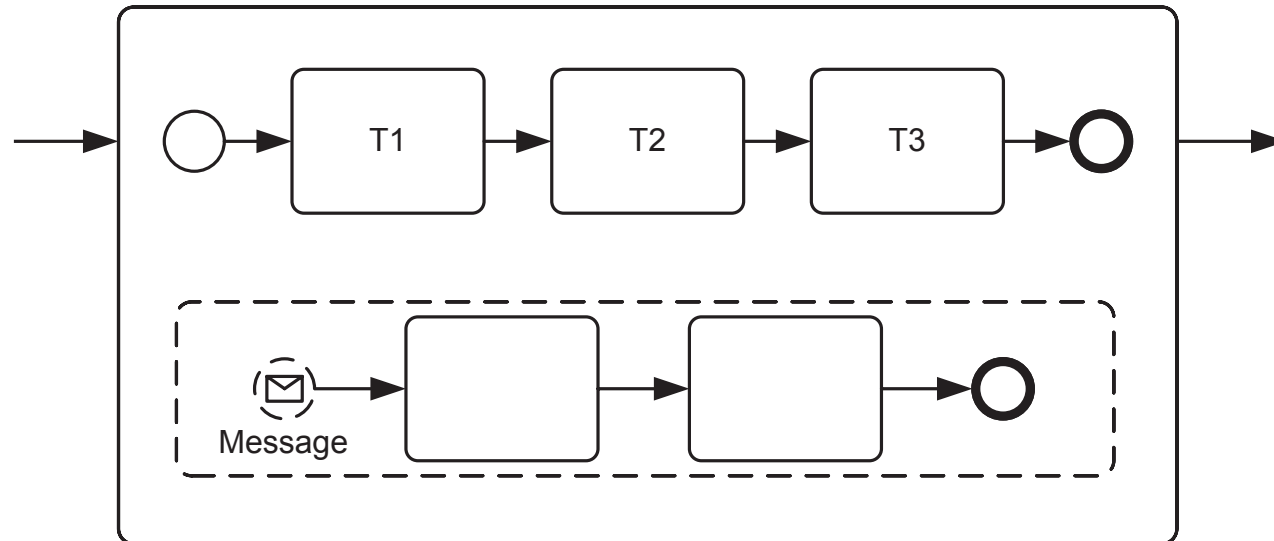
## Event Types (3)

- Compensation
  - Compensation is triggered or handled
- Terminate
  - Abort the current process
- Cancel
  - Rollback of current transaction is triggered or handled



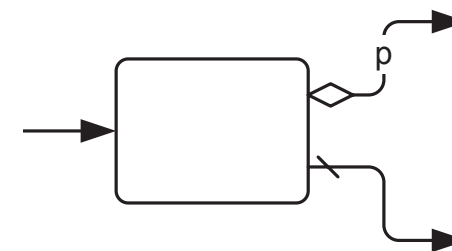
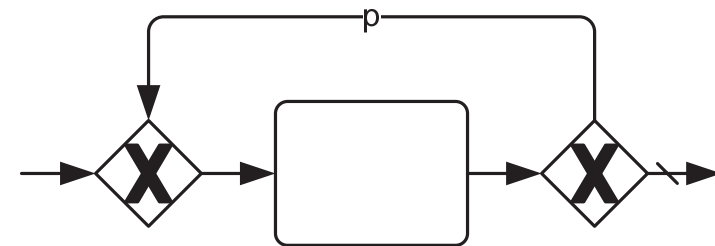
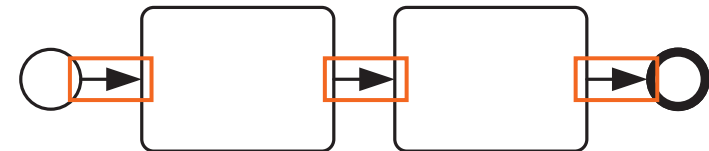


## Event Subprocess



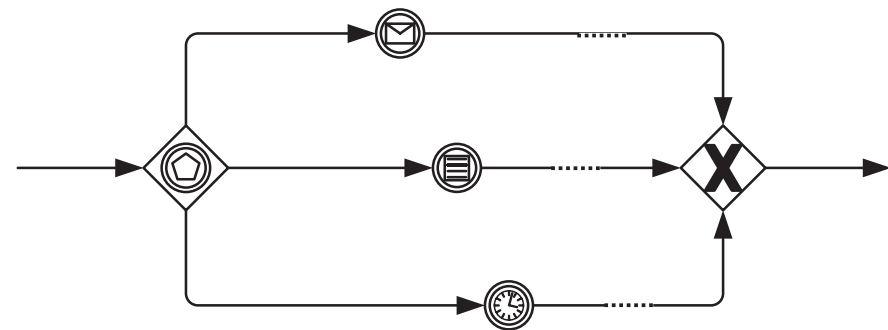
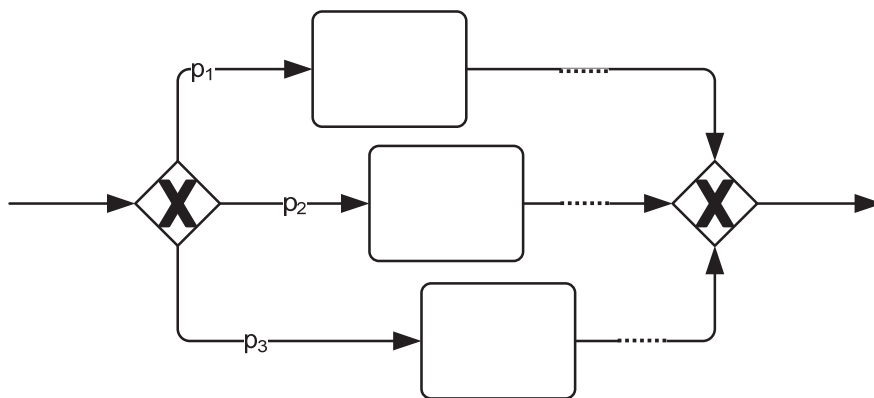
## Establishing Order

- Sequence flow
  - Perform steps in sequence
- Gateways
  - Split and merge the flow of control
    - Conditional execution
    - Parallel execution
    - Cyclic execution
    - ...
- Shortcuts
  - Implicit IOR in case of multiple outs
  - Implicit XOR in case of multiple ins



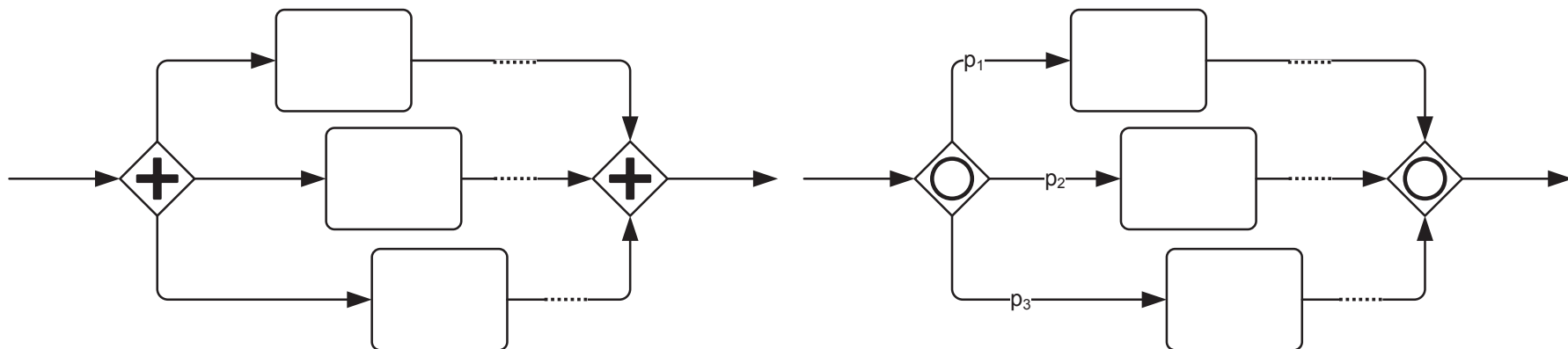
## Gateway Types (1)

- Exclusive Gateway
  - Selects exactly one branch, the first whose condition is true
  - Merges control flow from several exclusive branches
- Exclusive Gateway – event-based
  - Selects exactly one branch, based on event trigger
  - (Merge as before)



## Gateway Types (2)

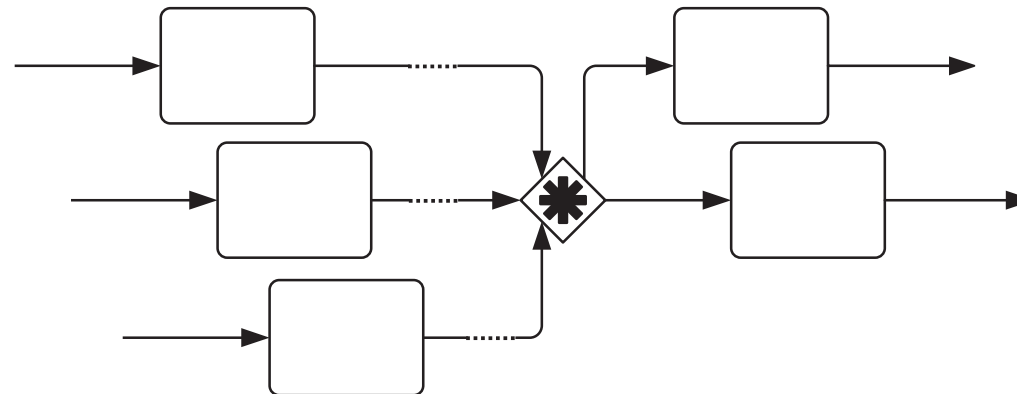
- Parallel Gateway
  - Forks control flow into parallel branches
  - Joins control flow from parallel branches
- Inclusive Gateway
  - Forks control flow into parallel branches, activating only those whose conditions are true
  - Joins control flow from several inclusive branches



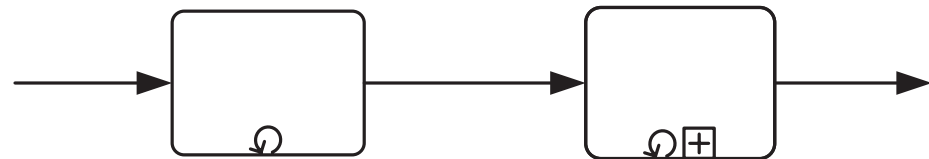
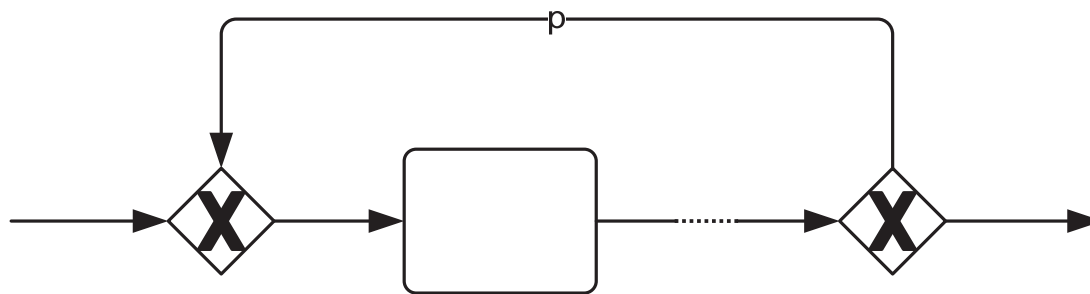


## Gateway Types (3)

- Complex Gateway
  - Only exists as converging gateway
    - Mandatory standard output, optional “reset” output
  - Used to support complex synchronization behavior
    - Arbitrary condition on incoming connectors for standard output
      - E.g., n out of m
    - IOR for remaining connectors for reset output
      - E.g. remaining (m-n) out of m



# Loops



## Multi-Instance

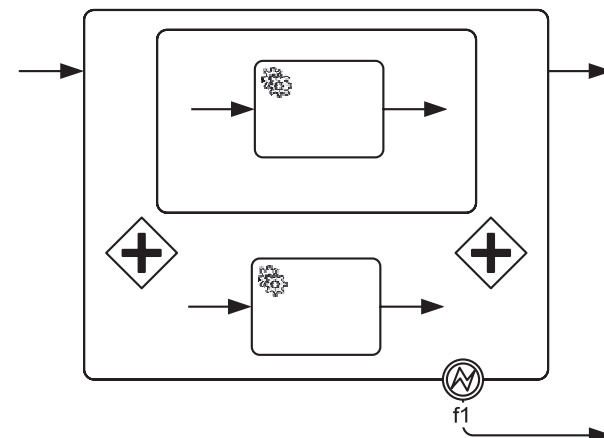
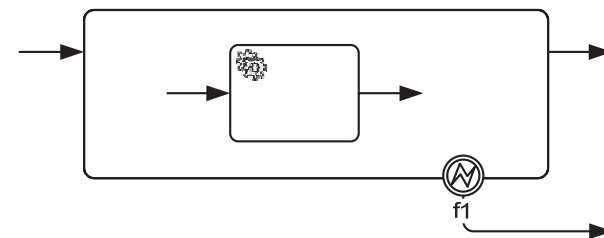
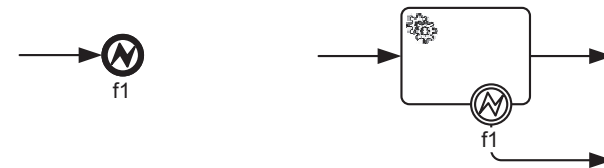
Multiple instantiation of an activity at runtime

- Data-driven – via integer expression or cardinality of data array
- Can be performed sequentially or in parallel
- Different behavior alternatives regarding completion and aborting still running instances
- Multi-instance behavior can be specified for any activity



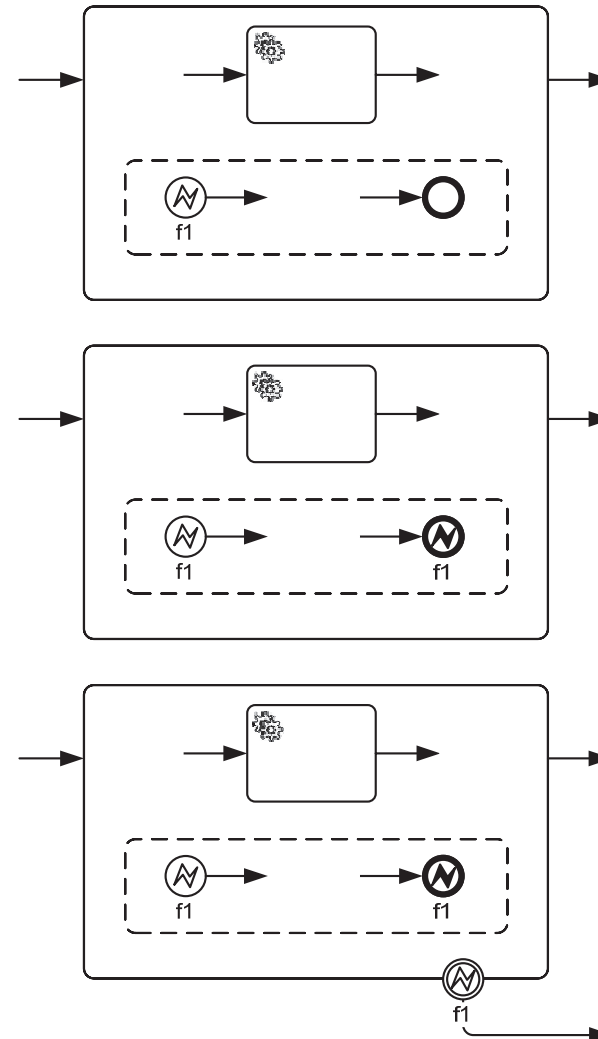
## Error Handling

- Errors are raised internal to an activity, or explicitly
- Errors are caught on a boundary
  - ... task
  - ... sub-process
    - Ongoing parallel work is terminated
- Uncaught errors are propagated up the scope tree
  - Termination is propagated down



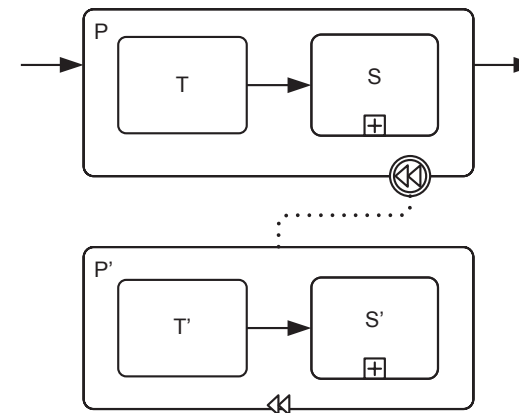
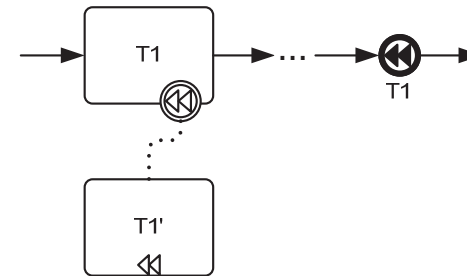
## Error Handling (2)

- Inline error handler
  - ... can “absorb” error
  - ... can rethrow error
- Upwards propagation extended
  - Inline handlers are considered first
  - Immediately enclosed boundary handlers second



## Compensation (1)

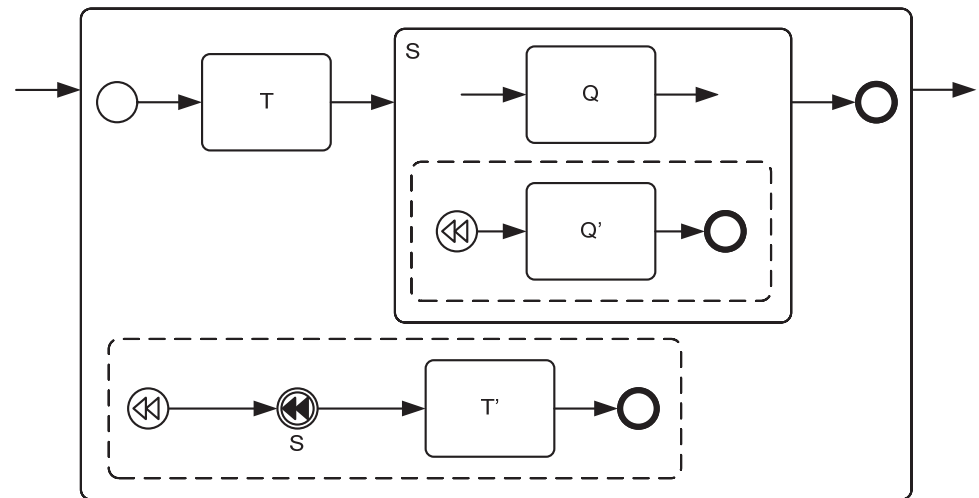
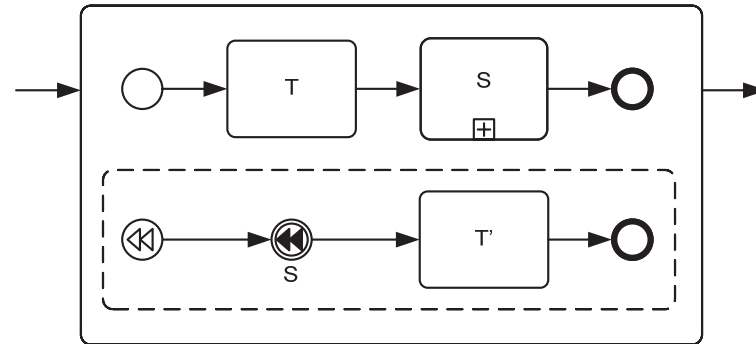
- Compensation pairs
  - Activity can be associated with “undo” activity
- Compensation triggered via event
  - Will trigger execution of associated
- No access to subprocess context
  - Only black-box compensation





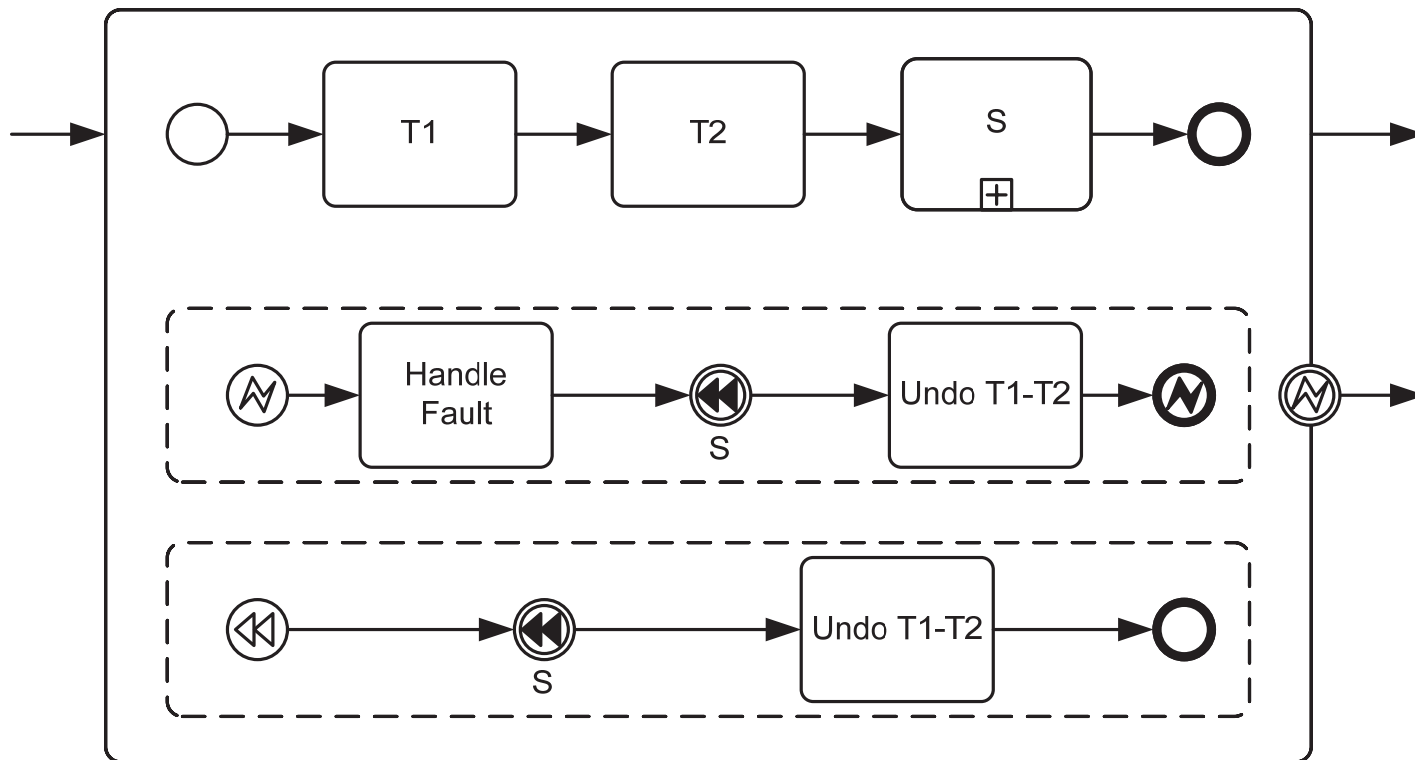
## Compensation (2)

- Compensation “Event Sub-process” (Handler)
  - Access to subprocess context – white-box compensation
  - Allows for recursive compensation definition
- Compensation Triggering
  - From a compensation handler
  - From an error handler



2.0

## Error Handling and Compensation





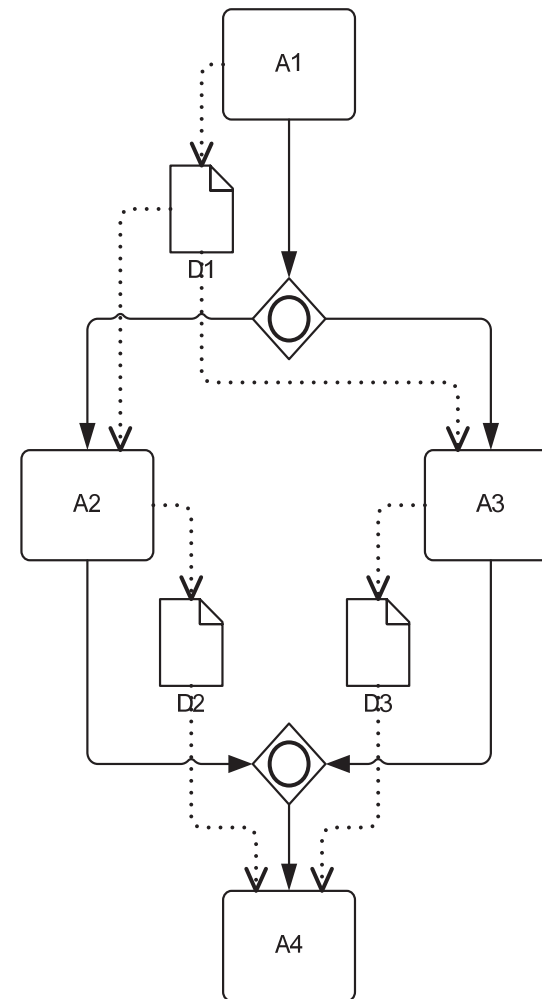


## Error Handling and Compensation

- Successful completion of subprocess
  - Preparation for possible compensation, snapshot of data is taken
- Unsuccessful completion of subprocess
  - *Presumed abort*: no side effects may persist
  - Error handler possibly needs to take care
- Compensation
  - Successful completion is undone
  - Snapshot data is made available to handler

## Data Flow

- Data Object
  - Used to store data
  - Pre-defined structure definition
  - Scoped, with standard visibility rules and life-time
- Data association 2.0
  - Describe data flow between data objects and flow nodes (activities or events)



## “Item-aware Elements” (1)

### Data Object

- Pre-defined (modeled) structure definition
- Visualized in process diagram
- Contained in enclosing (sub-) process
- Single instance or collection



CustomerRecord



Set of Customers

### Property

- Pre-defined (modeled) structure definition
- Not visualized
- Contained in (sub-) process or activity



## “Item-aware Elements” (2)

### Data Input

- Input of an activity
- Not visualized

### Input Set

- Collection of inputs forming a valid set

### Example Input Set

- { customerID  
| (customerName, address, SSN) }

### Data Output

- Output of an activity
- Not visualized

### Output set

- Collection of outputs forming a valid set

### Example Output Set

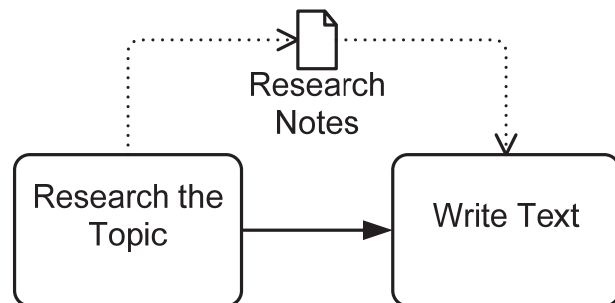
- { customerRating  
| error  
}



## Data Association

### Input data association

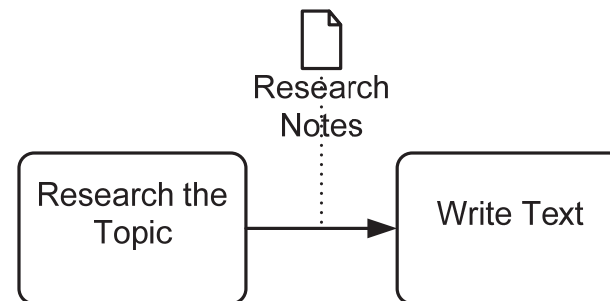
- Provide data for inputs of an activity
- Possibly multiple sources: data objects, properties, expressions
- Transformation specified as expression



Data input and data output association

### Output data association

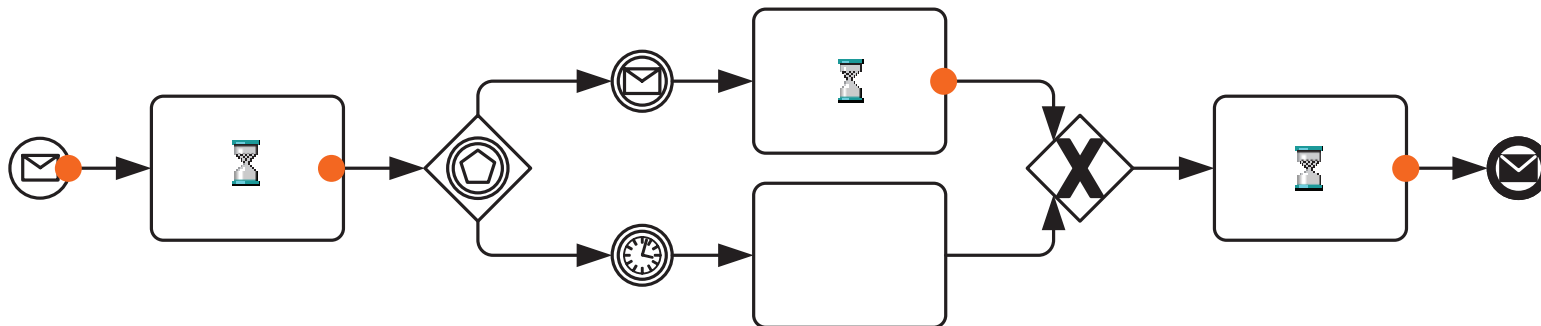
- Update data objects or properties after completion of an activity
- Sources: activity's data outputs
- Transformation specified as expression



Shortcut notation

## Execution Semantics

- Execution Semantics is described by means of token flow
  - Tokens flow along sequence flow
  - Tokens are produced and consumed by activities, events and gateways

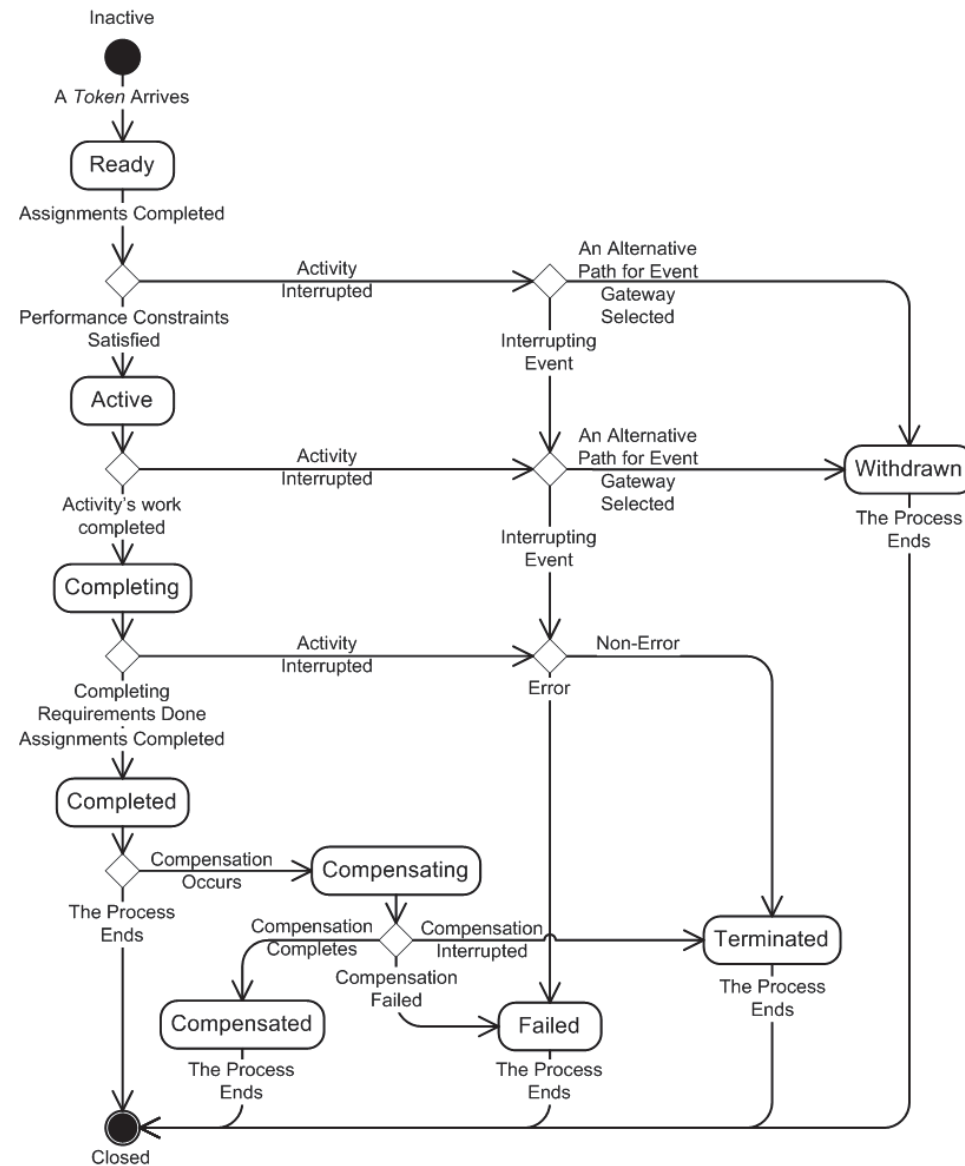


## Process Life-cycle

- Instantiation ...
  - ... when start event is triggered
    - Exception: If start event is part of an existing conversation, that conversation is joined
- Termination ...
  - ... when all of the following holds
    - All start events have been triggered (one for each group in case of starting event-based gateways)
    - No remaining token
    - No active activity

2.0

# Activity Life-cycle



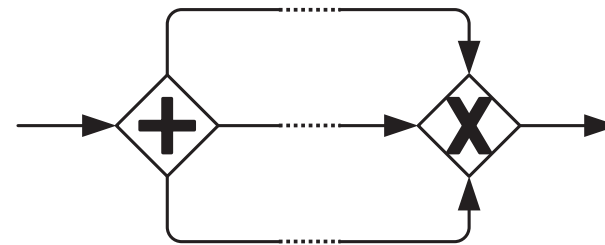


## Soundness

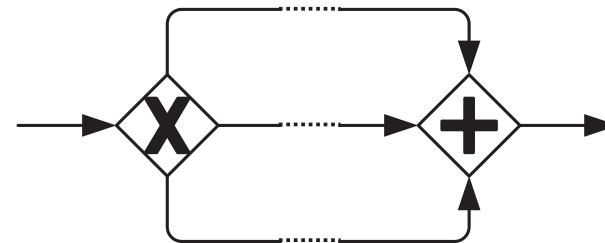
Sound process:

- No lack of synchronization
- Deadlock free

- Lack of synchronization



- Deadlock



BPMN does not require processes to be sound



## Mapping from BPMN to BPEL

- Pre-req: Process is sound
- Basic mapping
  - Syntactical mapping: Basic BPMN blocks are mapped to BPEL constructs
    - Semantics are preserved
  - Applies to subset of all sound BPMN processes
- Extended mapping
  - Builds on basic mapping
  - Any mapping of BPMN to BPEL that preserves the BPMN semantics
  - Spec describes a number of specific patterns
  - Extensible by vendors, allows usage of vendor extensions

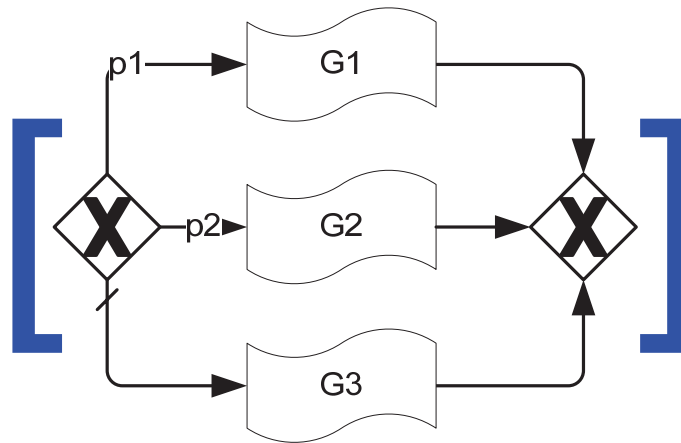


## BPEL Mapping

- Recursive description, driven by BPMN syntax structure
- Notation:
  - [BPMN fragment] = BPEL fragment



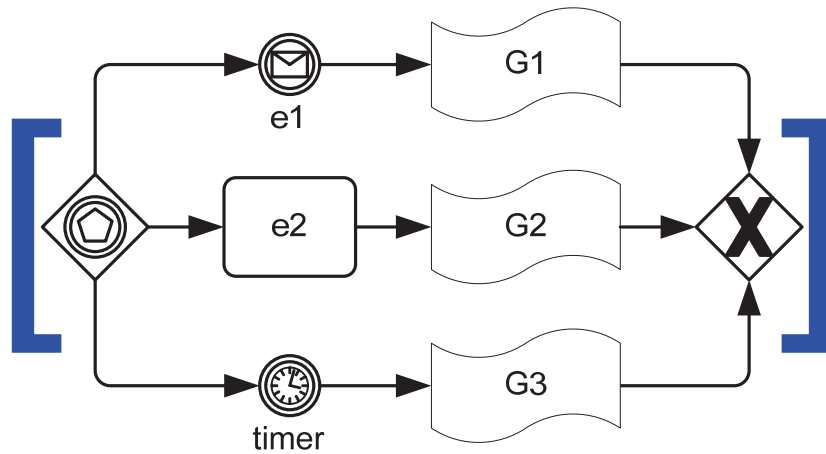
## BPEL Mapping Example – If-Elseif-Else



=

```
<if><condition>[p1]</condition>  
  [G1]  
<elseif><condition>[p2]</condition>  
  [G2]  
</elseif>  
<else>  
  [G3]  
</else>  
</if>
```

## BPEL Mapping Example – Pick



=

```
<pick createInstance="[instantiate? 'yes':'no']">
  <onMessage partnerLink="[e1-operation-interface]"
    operation="[e1-operation]">
    [G1]
  </onMessage>
  <onMessage partnerLink="[e2-operation-interface]"
    operation="[e2-operation]">
    [G2]
  </onMessage>
  <onAlarm>
    [timer-spec]
    [G3]
  </onAlarm>
</pick>
```