An ontology for the Business Process Modelling Notation

Marco Rospocher

Fondazione Bruno Kessler, Data and Knowledge Management Unit
Trento, Italy
rospocher@fbk.eu :: http://dkm.fbk.eu/rospocher

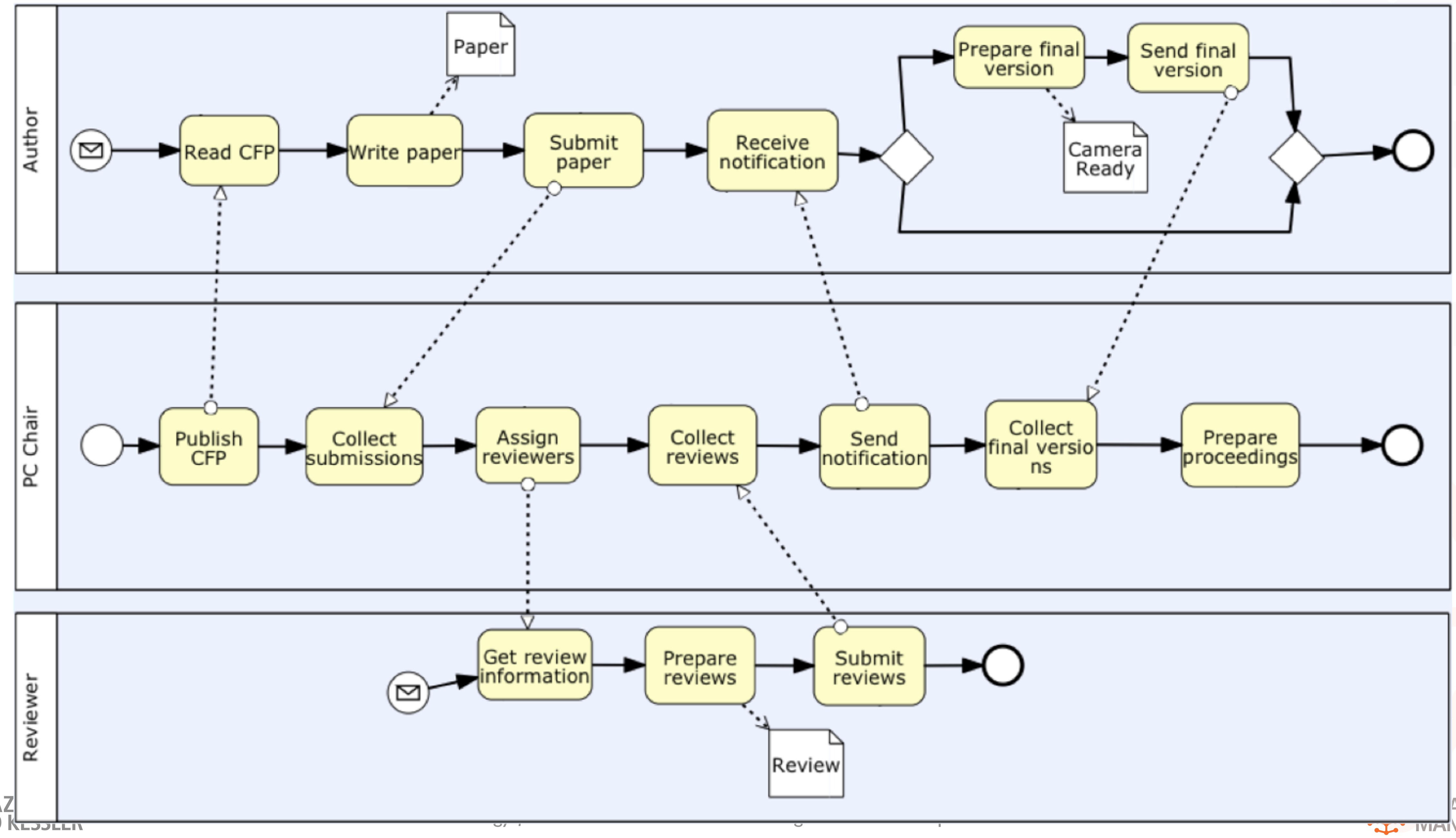
joint work with:

Chiara Ghidini, Luciano Serafini

8th International Conference on Formal Ontology in Information Systems (FOIS 2014)
September, 22-25, 2014 - Rio de Janeiro, Brazil

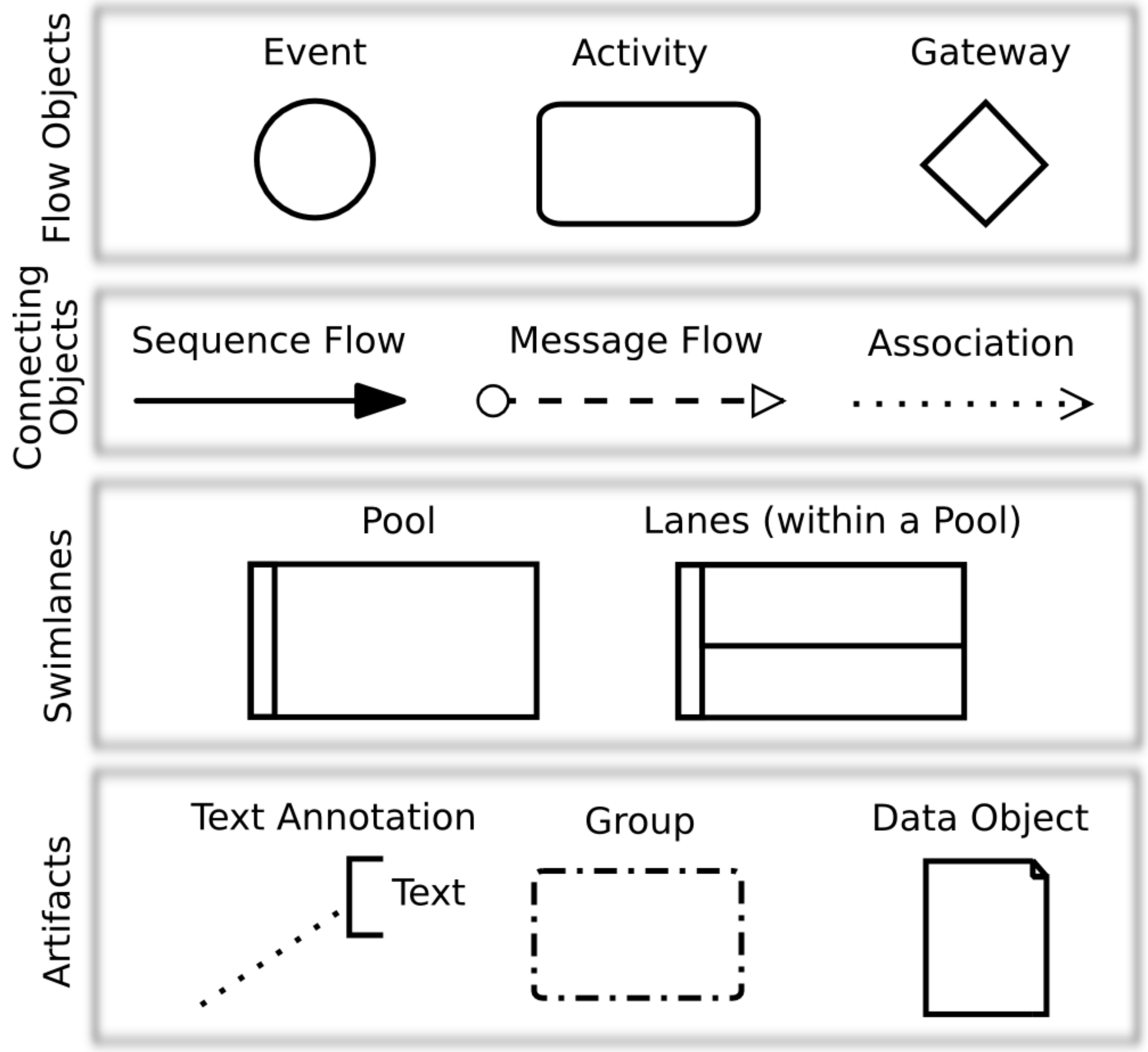
Business Process Modelling Notation (BPMN)

- State of the art graphical language for the specification of business processes



Business Process Modelling Notation (BPMN)

Core Element Set



for representing something that happens (event), work to be performed (activity), and control flow elements (gateway);

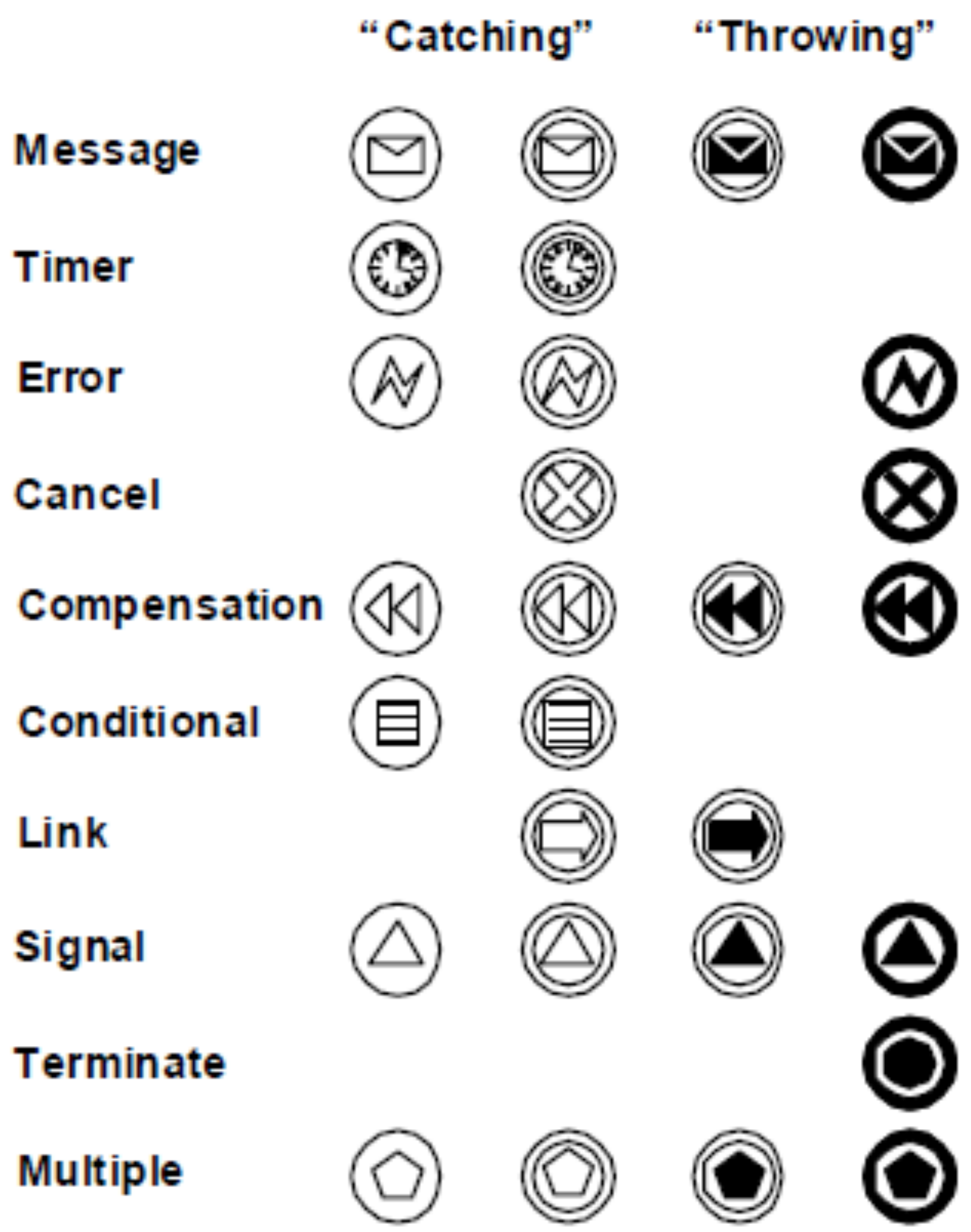
for showing the order in which activities are performed (sequence flow), ...

for describing participants in a process (pool), and to organize and categorize activities (lane);

for representing data processed/produced by activities (data object), informal grouping of activities (group), ...

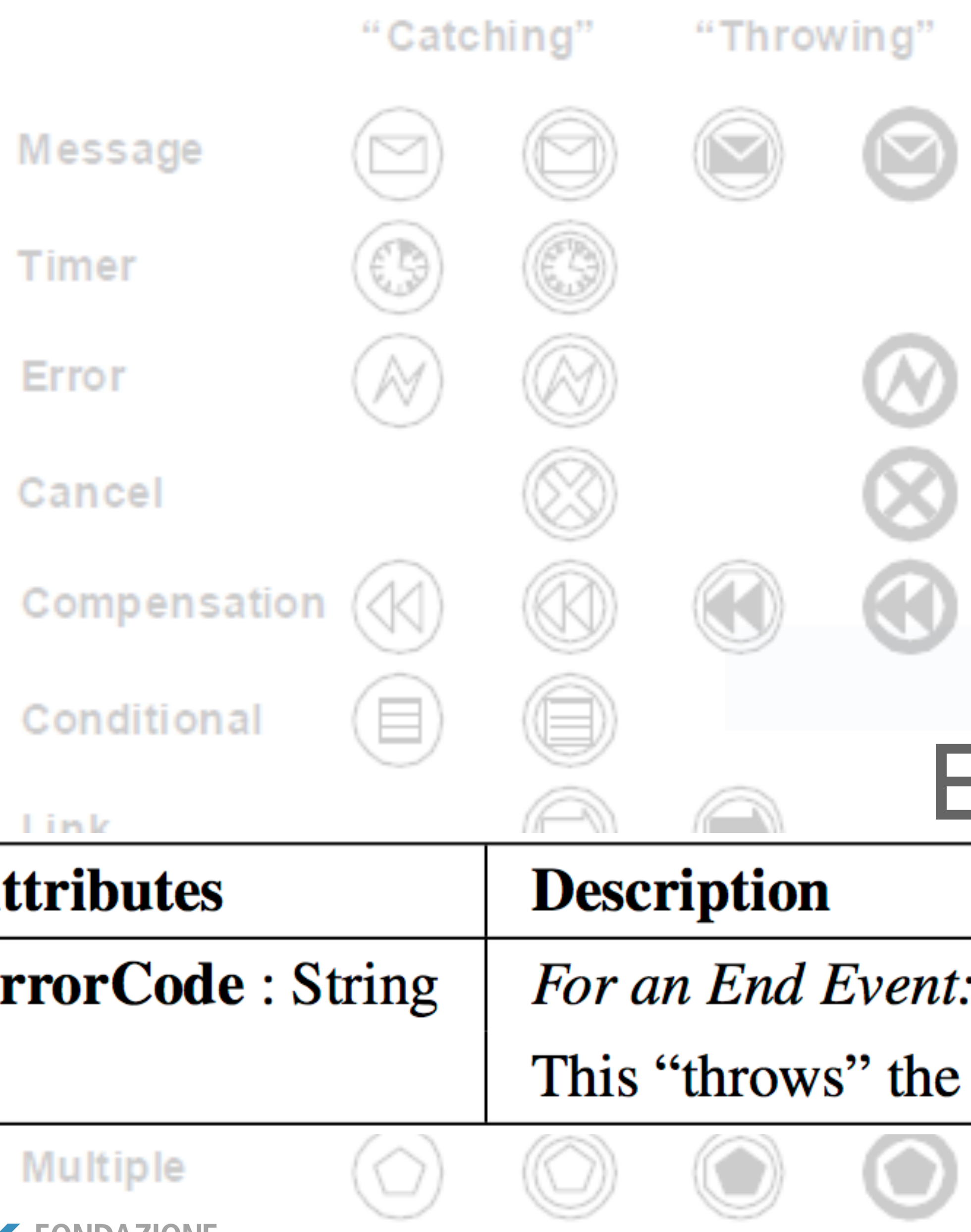
Business Process Modelling Notation (BPMN)

Extended Element Set (e.g. Event types)



Business Process Modelling Notation (BPMN)

Extended Element Set (e.g. Event types



ErrorCode attribute for Error Event

Attributes	**Description**
ErrorCode : String	*For an End Event:* If the Result is an Error, then the ErrorCode MUST be supplied. This "throws" the error. [..]

Our Contribution: An ontology for BPMN

- An OWL-DL formalization of the BPMN specification
- It accurately encodes:
  - the classification of all the elements of the BPMN language
  - the formal representation of the attributes and conditions describing how the elements can be combined to obtain a "valid" BPMN business process
- The proposed formalization:
  - provides a terminological description of the language;
  - enables representing any actual BPMN diagram as a DL A-Box
    - enables several reasoning-based services
- It covers BPMN v1.1…
  - we are in the process of updating to BPMN 2.02

Disclaimer

- The BPMN Ontology…
  - …is not intended to model the dynamic behaviour (behavioural semantics) of a BPMN process
    - Ontology languages are not particularly suited for this
    - better look at YAWL, PetriNets, …
  - …it provides an ontological formalization of BPMN as a graphical language, and not an ontological analysis in a foundational fashion
    - better look at works analysing BPMN wrt to
      - ABDESO/UFO (Guizzardi and Wagner)
      - Dolce (Sanfilippo, Borgo, and Masolo - see conference next session)

Modelling Process

Scope and Boundaries: Ontology Intended Uses

- Checking the compliance of a process diagram against the BPMN specification
  - e.g., the process diagram has at least one starting event and one end event, constructs are combined in the correct way
- Checking additional application-specific design guidelines
  - guidelines to guarantee process diagram readability (e.g., diagram should not contain more than ten sub-processes, every gate should have at most three out-going flows)
- Semantic description and retrieval of process diagrams (or process diagram elements)
  - e.g., to state that a certain sub-process is of type "privacy critical", and to be able to retrieve all process diagrams that contains privacy critical sub-processes, or all privacy critical activities within a diagram

Modelling Process

Scope and Boundaries: Competency Questions (excerpt)

- How many flow elements does process X contain?
- What is the error code associated to error event W?
- What type of BPMN elements does sub-process Y in process X contains?
- What is the BPMN element connected by a sequence flow to activity Z?
- Is there a path of sequence flows connecting activity Z_1 to activity Z_2?
- Is process XYZ a valid process according to the BPMN specification?
- ….

Modelling Process

Our Trusted Friend: BPMN Specification Document

- For each element, it provides:
  - an introductory description of the element, with some general properties and conditions
  - a compact tabular description of each element's attribute
    - name, value type, multiplicity details, conditions for instantiation
  - conditions holding for connecting the current element with other elements of the language
  - additional details on execution level aspects of the element
- Free text document, with some structure

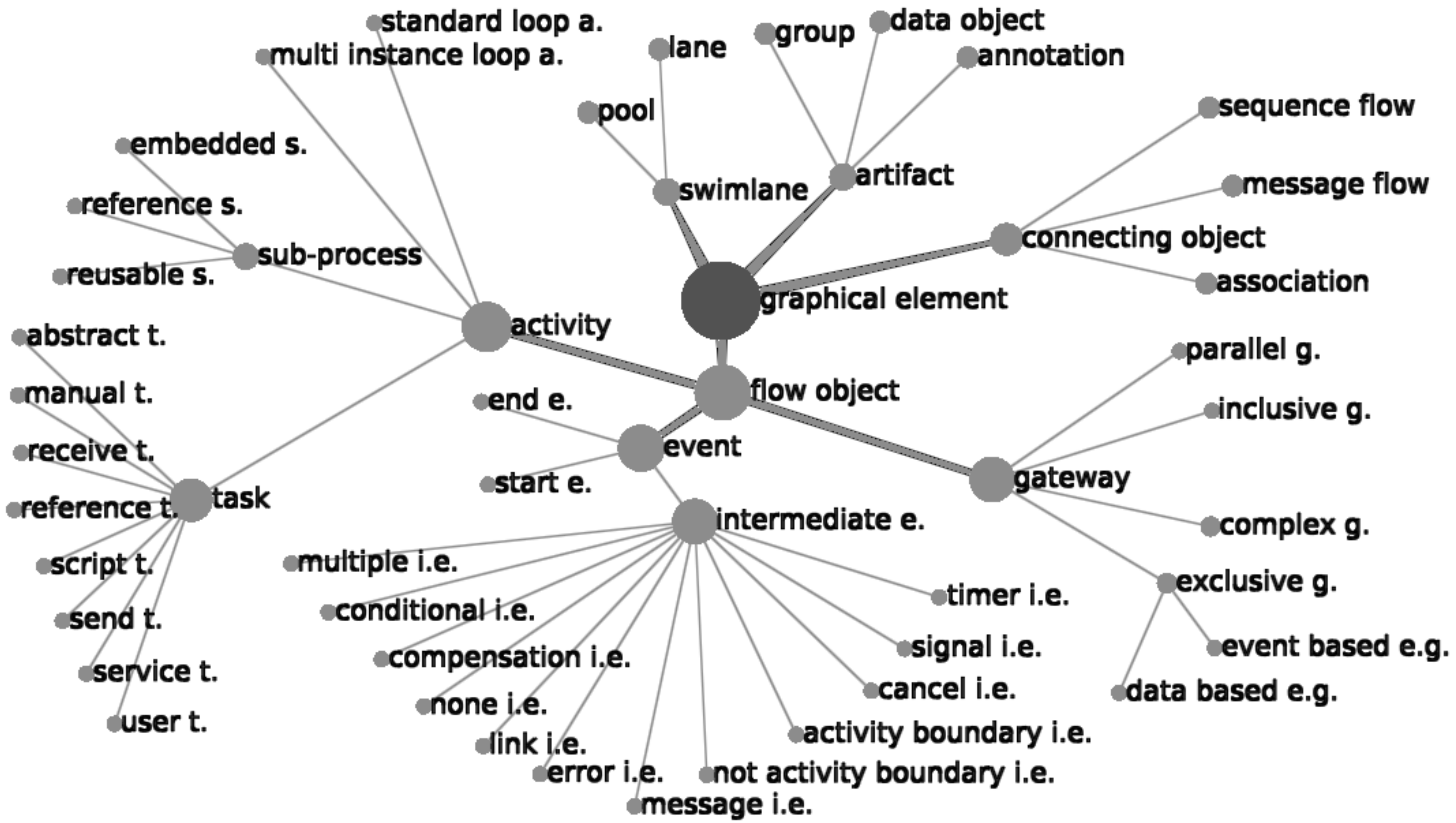
Modelling Process

Step 1 of 3: Signature Identification

- Identification and classification of all the elements of the language
  - We associated each element of the language to a class in the ontology
  - Guided by the classification of these elements provided by the BPMN Specification, we formalized the initial taxonomy of the ontology

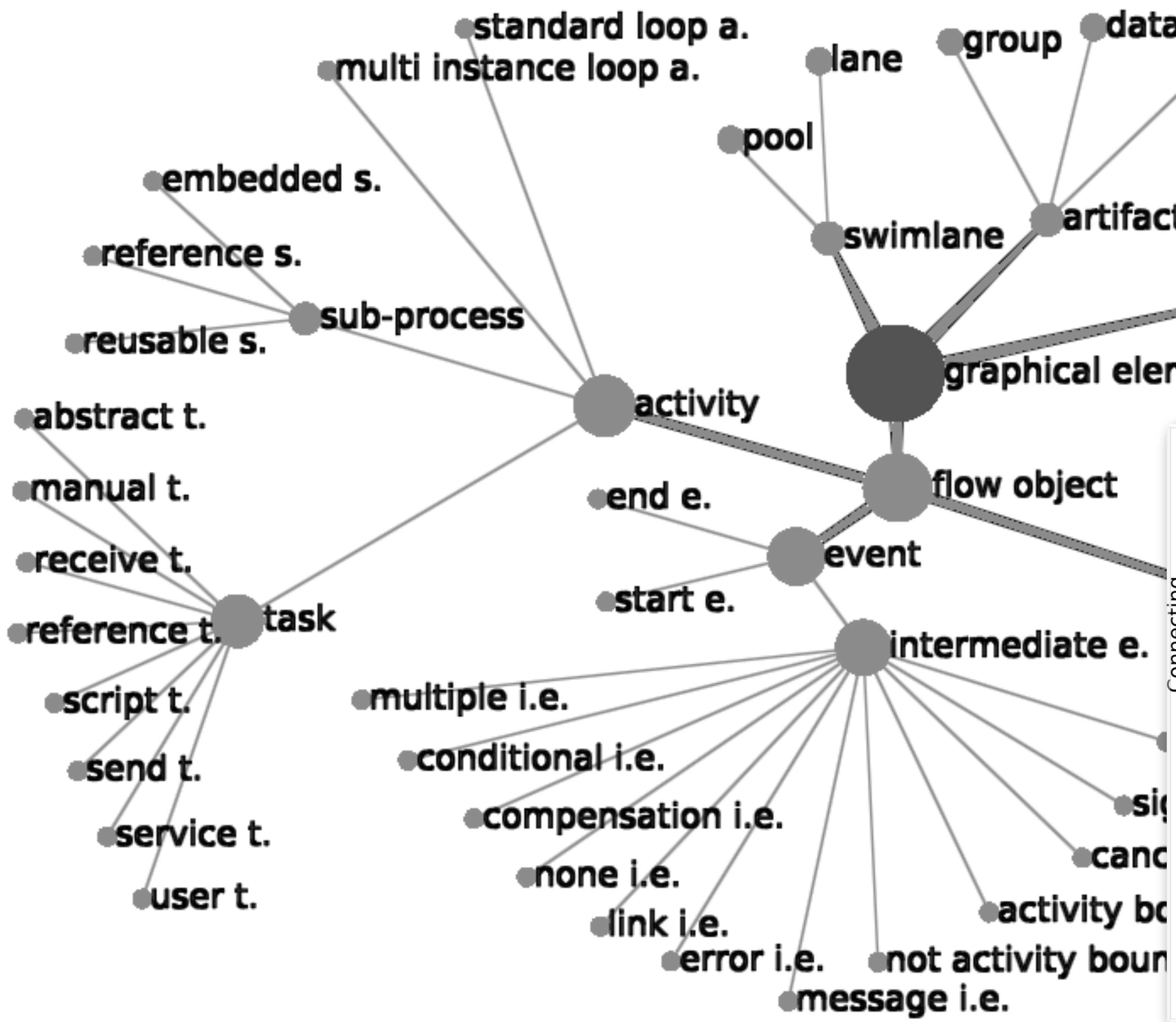
Modelling Process

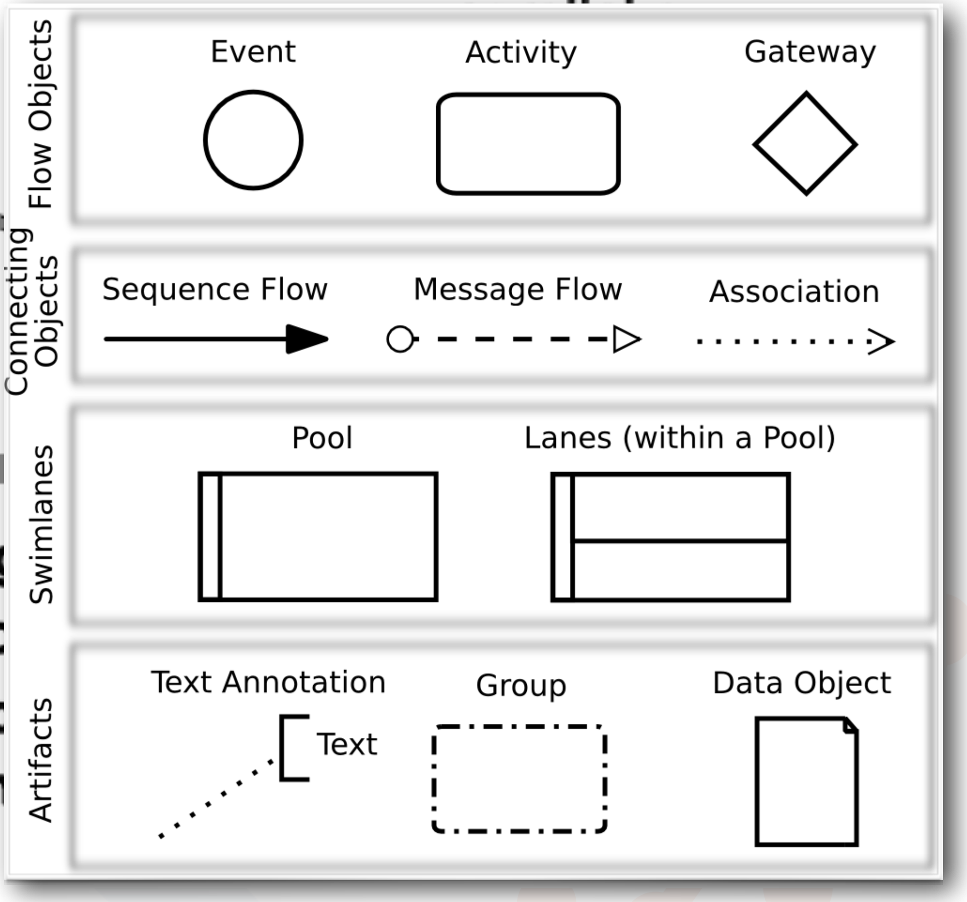
Step 1 of 3: Signature Identification



Modelling Process

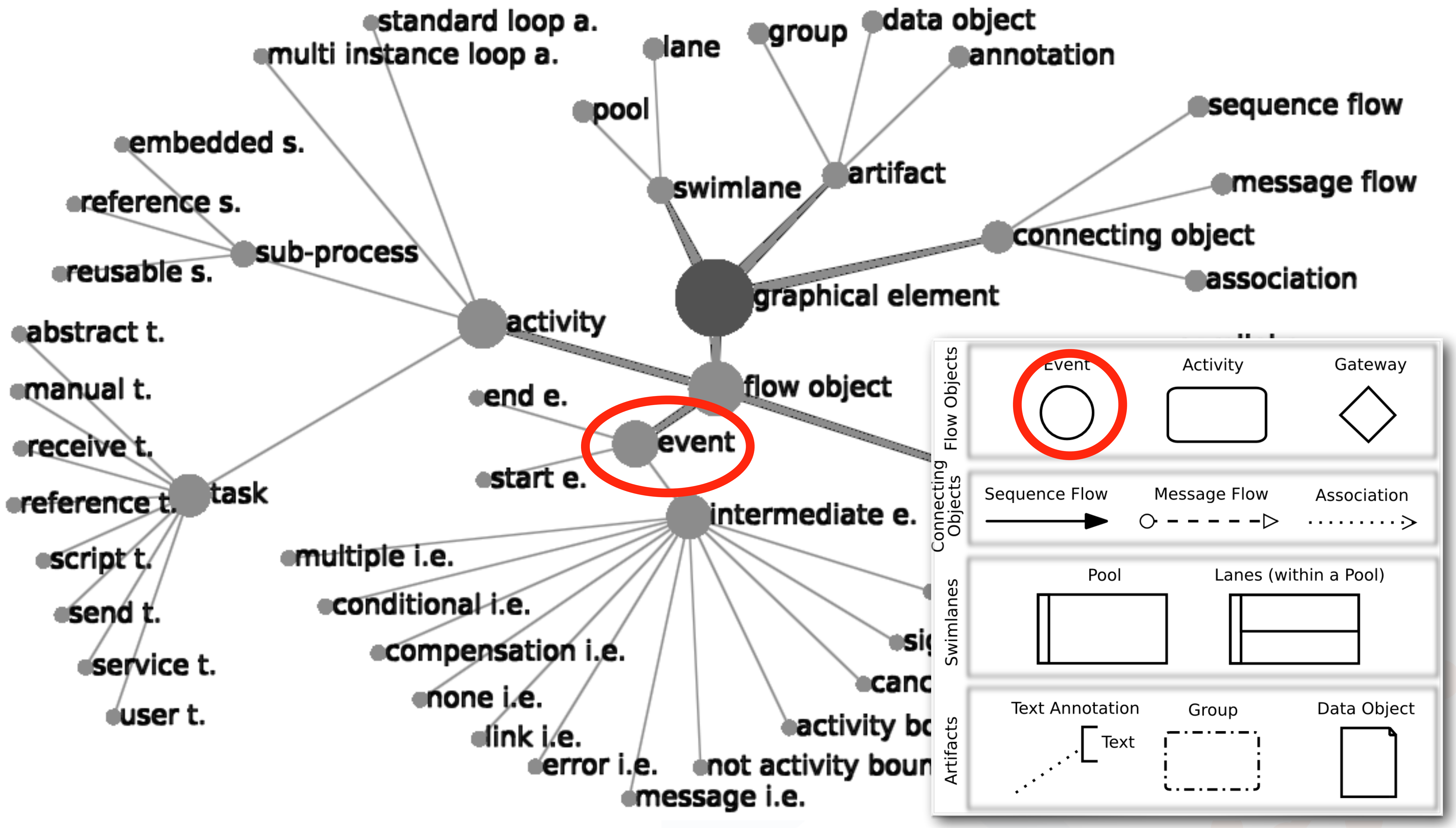
Step 1 of 3: Signature Identification





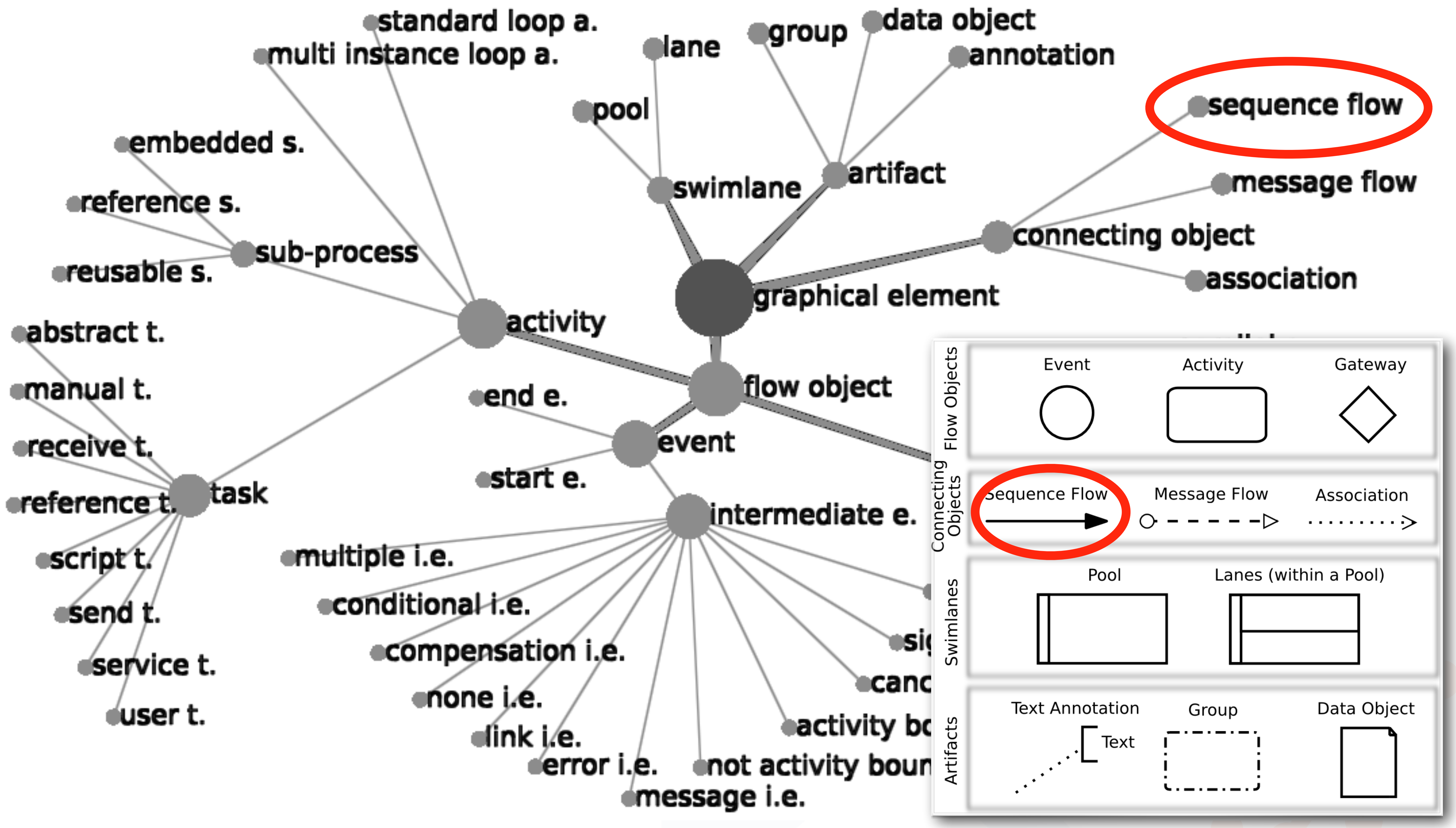
Modelling Process

Step 1 of 3: Signature Identification



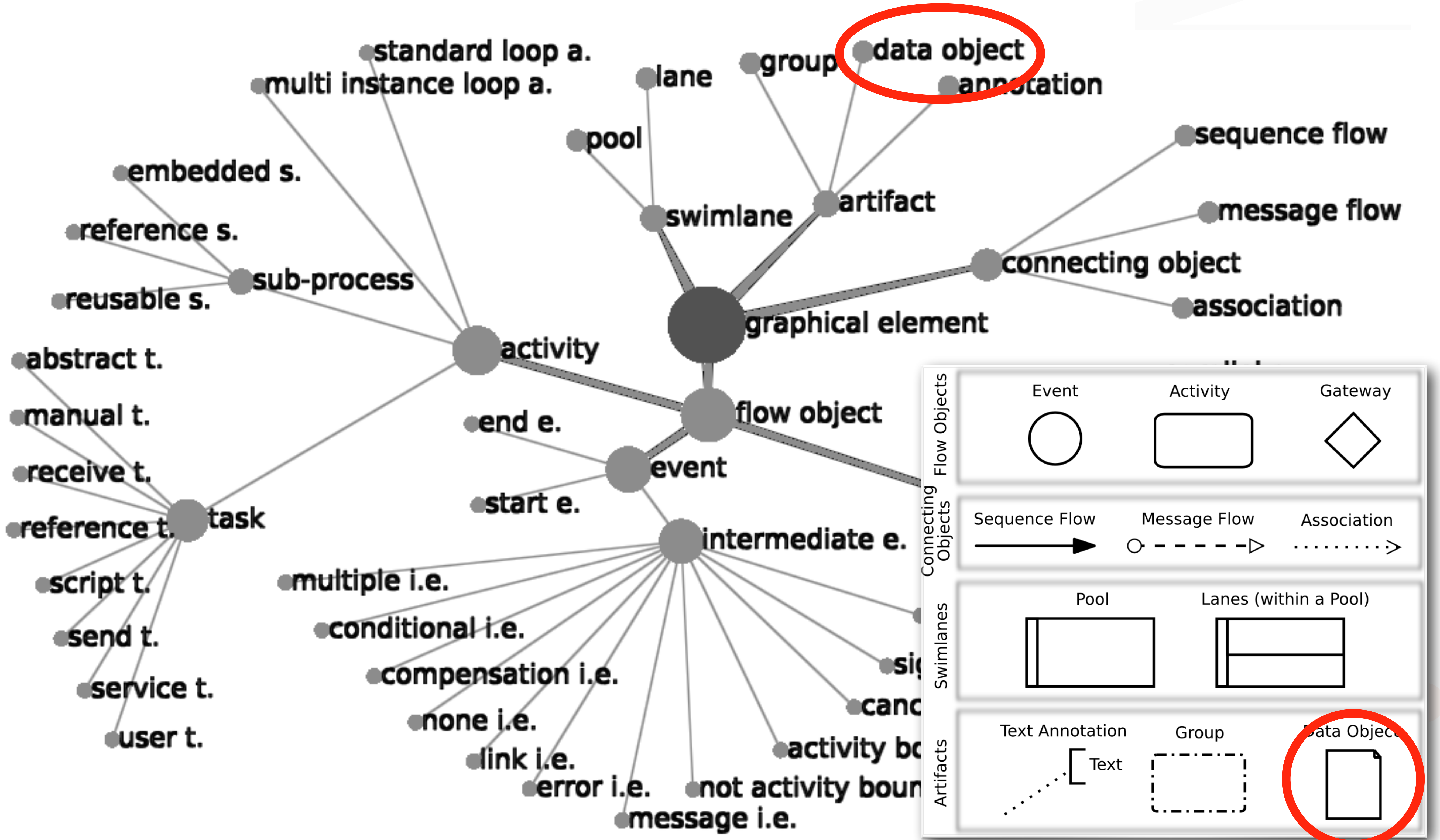
Modelling Process

Step 1 of 3: Signature Identification



Modelling Process

Step 1 of 3: Signature Identification



Modelling Process

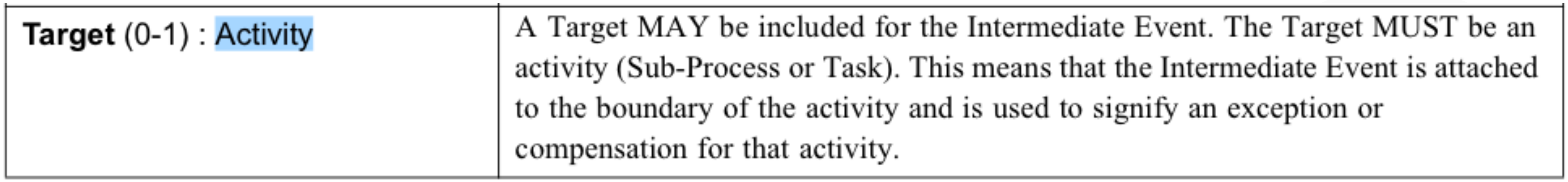
Step 2 of 3: Attribute Restrictions

- An attribute is formalized either as datatype property or as an object property
- Three situations considered:
  1. the value type of the attribute is another BPMN element
  2. the value type of the attribute is a datatype, but only an enumerated set of options is allowed and some conditions may apply to these options
  3. the value type of the attribute is a datatype with no restriction

Modelling Process

Step 2 of 3: Attribute Restrictions (cont'd)

- Case I: The value type of the attribute is another BPMN element
- Example:
  - Target attribute of Intermediate Event [p47]



Target (0-1) : Activity	A Target MAY be included for the Intermediate Event. The Target MUST be an activity (Sub-Process or Task). This means that the Intermediate Event is attached to the boundary of the activity and is used to signify an exception or compensation for that activity.

- Formalization: as object property
  - domain: the class having the attribute
  - range: the class of the element mentioned as value type of the attribute

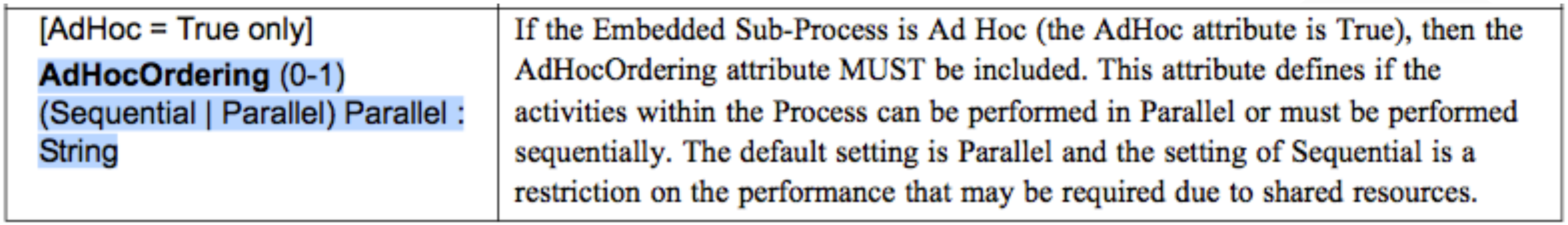
$$\exists hasIntermediateEventTarget.\top \sqsubseteq IntermediateEvent$$

$$\top \sqsubseteq \forall hasIntermediateEventTarget.Activity$$

Modelling Process

Step 2 of 3: Attribute Restrictions (cont'd)

- Case II: The value type of the attribute is a datatype, but only an enumerated set of options is allowed and some conditions may apply to these options
- Example:
  - AdHocOrdering attribute of Embedded SubProcess [p47]



[AdHoc = True only] **AdHocOrdering** (0-1) (Sequential | Parallel) Parallel : String	If the Embedded Sub-Process is Ad Hoc (the AdHoc attribute is True), then the AdHocOrdering attribute MUST be included. This attribute defines if the activities within the Process can be performed in Parallel or must be performed sequentially. The default setting is Parallel and the setting of Sequential is a restriction on the performance that may be required due to shared resources.

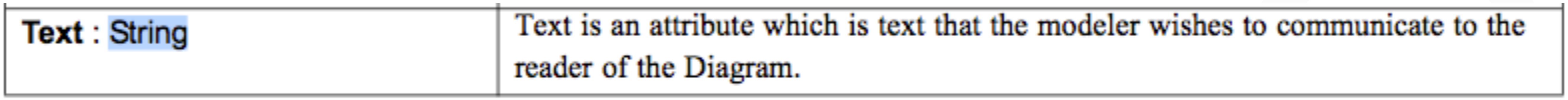
- Formalization: as object property
  - domain: the class having the attribute
  - range: a new class enumerating all possible values of the attribute

$$\exists hasESPAdHocOrdering.\top \sqsubseteq EmbeddedSubProcess$$
$$\top \sqsubseteq \forall hasESPAdHocOrdering.AdHocOrderingType$$

Modelling Process

Step 2 of 3: Attribute Restrictions (cont'd)

- Case III: The value type of the attribute is a datatype with no restriction
- Example:
  - Text attribute of Text Annotation [p95]



Text : String	Text is an attribute which is text that the modeler wishes to communicate to the reader of the Diagram.

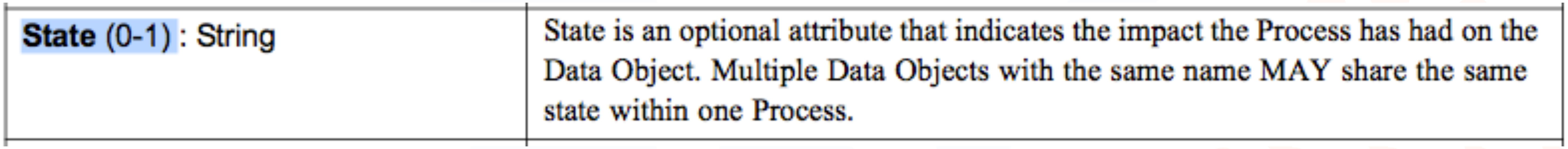
- Formalization: as datatype property
  - domain: the class having the attribute
  - range: a datatype compatible with the value type of the attribute

$$\exists hasTextAnnotationText.\top \sqsubseteq TextAnnotation$$
$$\top \sqsubseteq \forall hasTextAnnotationText.DT\{string\}$$

Modelling Process

Step 2 of 3: Attribute Restrictions (cont'd)

- For each attribute, we formalized its multiplicity details as an OWL cardinality restriction on the class having the attribute
  - (0..1) multiplicity is encoded as "at most one" OWL cardinality restriction
  - (1) multiplicity is encoded as "exactly one" OWL cardinality restriction
  - (1..n) multiplicity is encoded as "at least one" OWL cardinality restriction
  - (0..n) multiplicity is not encoded at all
- Example:
  - State attribute of Data Object [p94]



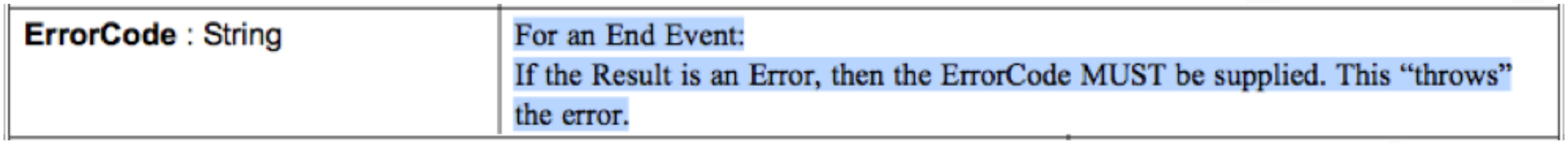
State (0-1) : String	State is an optional attribute that indicates the impact the Process has had on the Data Object. Multiple Data Objects with the same name MAY share the same state within one Process.

$$DataObject \sqsubseteq (\leq 1) hasState$$

Modelling Process

Step 2 of 3: Attribute Restrictions (cont'd)

- For each attribute, we also encode additional conditions ruling the usage of the attribute
- Example:
  - ErrorCode attribute of Error, in case the Error is a result of an End Event [p.94]



ErrorCode : String	For an End Event: If the Result is an Error, then the ErrorCode MUST be supplied. This "throws" the error.

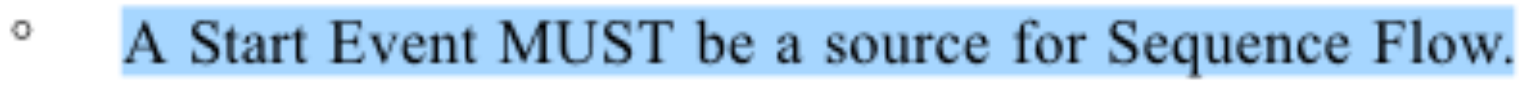
- Formalization: case by case

$$EndEvent \sqsubseteq \neg\exists hasResult.Error \sqcup \exists hasResult.(Error \sqcap \exists hasErrorCode)$$

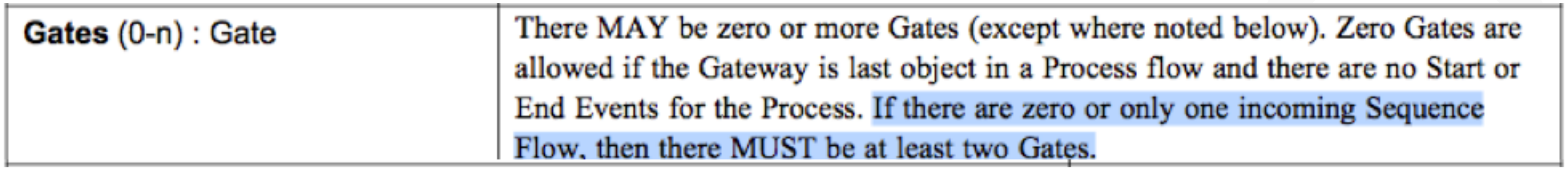
Modelling Process

Step 3 of 3: Structural Constraints Formalization

- Formalization of the conditions concerning the usage of the elements of the language to compose a BPMN diagram
- Examples: [p48,p72]



- A Start Event MUST be a source for Sequence Flow.

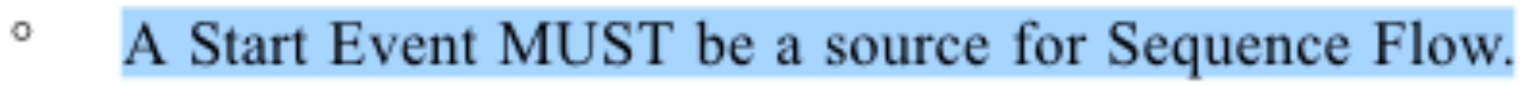


Gates (0-n) : Gate	There MAY be zero or more Gates (except where noted below). Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow, then there MUST be at least two Gates.

- Formalization: case by case

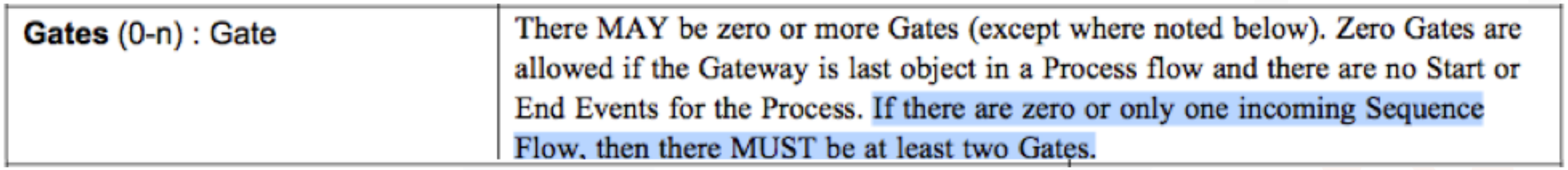
Modelling Process

Step 3 of 3: Structural Constraints Formalization



◦ A Start Event MUST be a source for Sequence Flow.

$$StartEvent \sqsubseteq \exists hasConnectingObjectSource^{-1}.SequenceFlow$$



Gates (0-n) : Gate	There MAY be zero or more Gates (except where noted below). Zero Gates are allowed if the Gateway is last object in a Process flow and there are no Start or End Events for the Process. If there are zero or only one incoming Sequence Flow, then there MUST be at least two Gates.

$$\begin{aligned} Gateway \sqsubseteq & (\geq 2)hasSequenceFlowTarget^{-1} \sqcup \\ & ((\leq 1)hasSequenceFlowTarget^{-1} \sqcap \\ & (\geq 2)hasGatewayGate) \end{aligned}$$

The BPMN Ontology

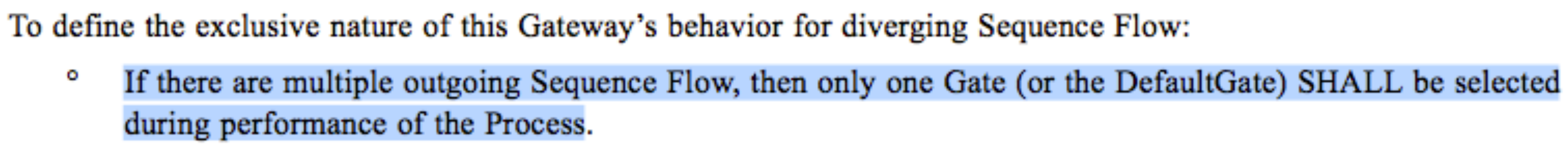
Limitations

- A few documented properties and conditions are not encoded in the BPMN Ontology:
  - Execution level properties
  - Attribute default values
  - “Undecideable” conditions

The BPMN Ontology

Limitations: Execution level properties

- These properties specifies the behavioral nature of the graphical elements in a BPMN diagram
- Example: [p.77]



To define the exclusive nature of this Gateway's behavior for diverging Sequence Flow:

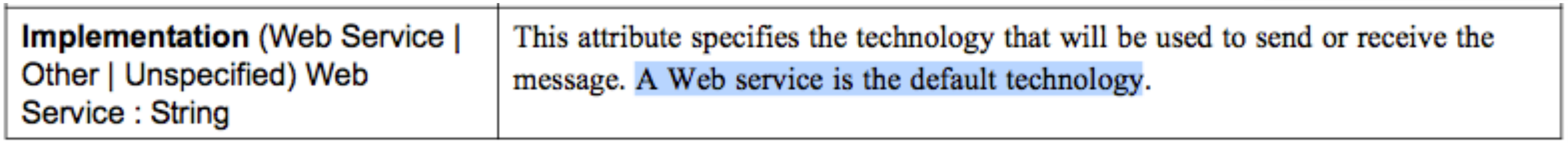
- If there are multiple outgoing Sequence Flow, then only one Gate (or the DefaultGate) SHALL be selected during performance of the Process.

- Not considered, as our BPMN Ontology is not intended to model the dynamic behaviour of BPMN diagrams

The BPMN Ontology

Limitations: Attribute default values

- Attributes default values: the value taken by the required attributes when not explicitly assigned in the model
- Example: [p.52]



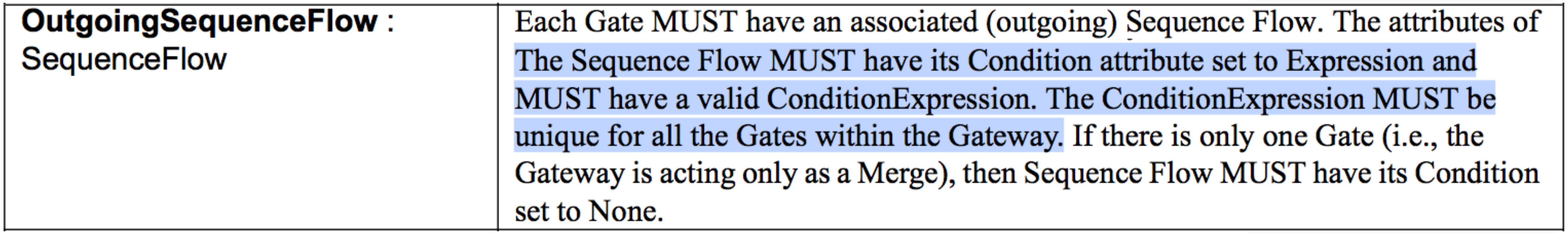
Implementation (Web Service | Other | Unspecified) Web Service : String	This attribute specifies the technology that will be used to send or receive the message. A Web service is the default technology.

- Attributes default values have not been formalized because OWL does not support the specification of properties default values

The BPMN Ontology

Limitations: “Undecideable” conditions

- There is a limited number of conditions which are not represented
  - we wanted to remain in a decidable fragment of OWL
- Example: [p73]



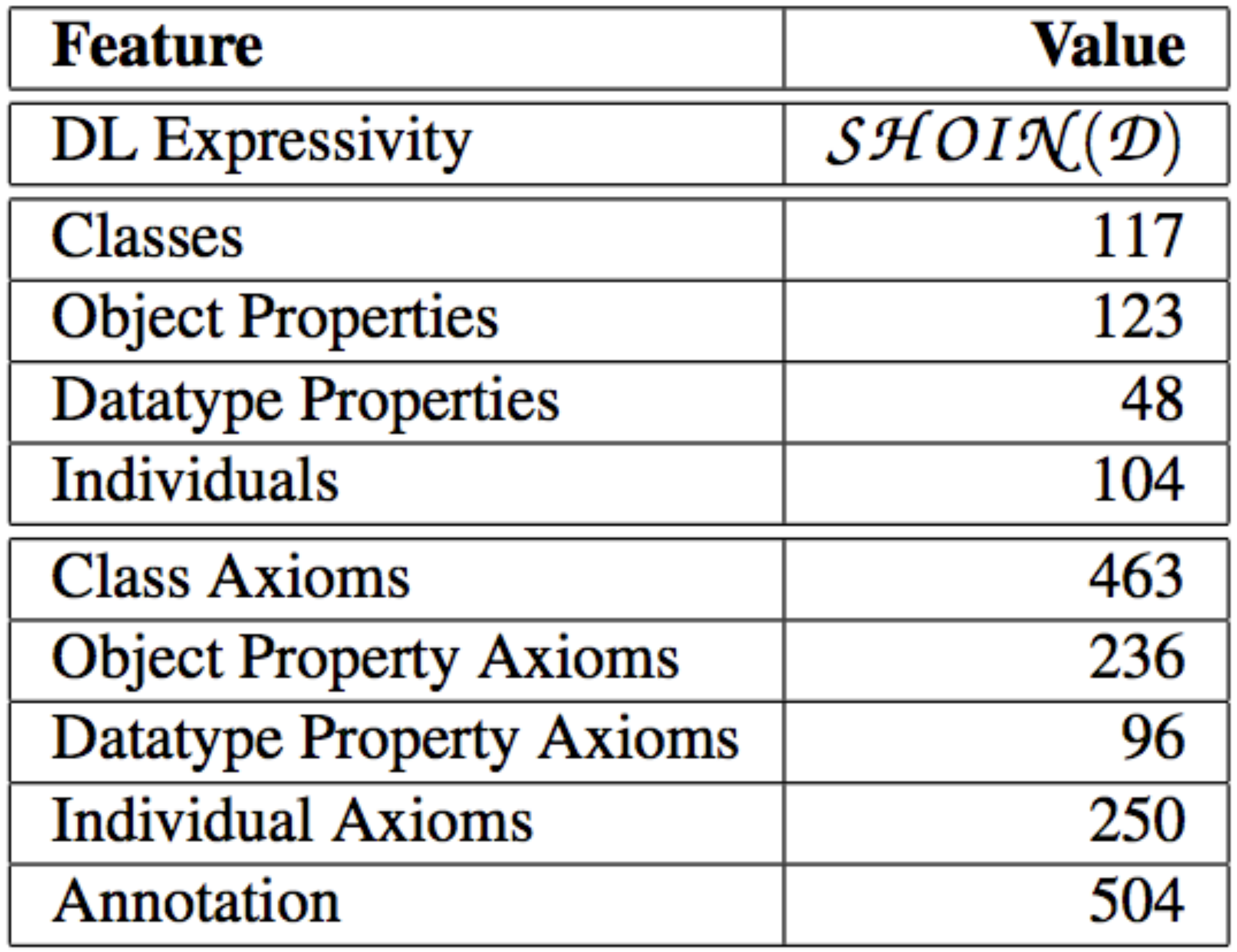
OutgoingSequenceFlow : SequenceFlow	Each Gate MUST have an associated (outgoing) Sequence Flow. The attributes of The Sequence Flow MUST have its Condition attribute set to Expression and MUST have a valid ConditionExpression. The ConditionExpression MUST be unique for all the Gates within the Gateway. If there is only one Gate (i.e., the Gateway is acting only as a Merge), then Sequence Flow MUST have its Condition set to None.

- This condition could be formalized by defining a functional property as the chain of two other properties
  - one connecting gateways to sequence flows
  - one associating conditions to sequence flows

but imposing number restrictions on property chains (or, more generally, on complex roles) lead to undecidability

The BPMN Ontology

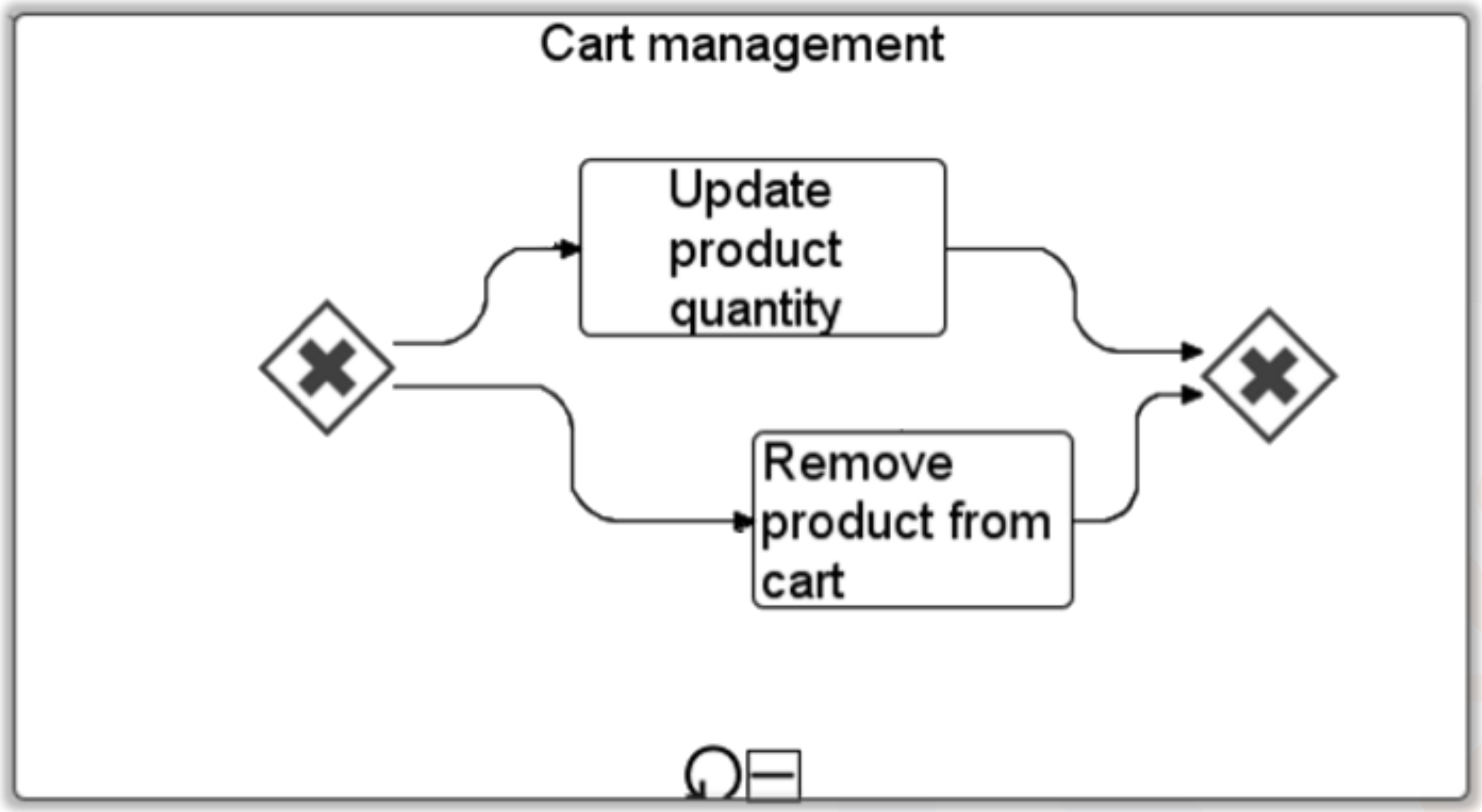
Ontology Metrics



Feature	Value
DL Expressivity	$\mathcal{SHOIN}(\mathcal{D})$
Classes	117
Object Properties	123
Datatype Properties	48
Individuals	104
Class Axioms	463
Object Property Axioms	236
Datatype Property Axioms	96
Individual Axioms	250
Annotation	504

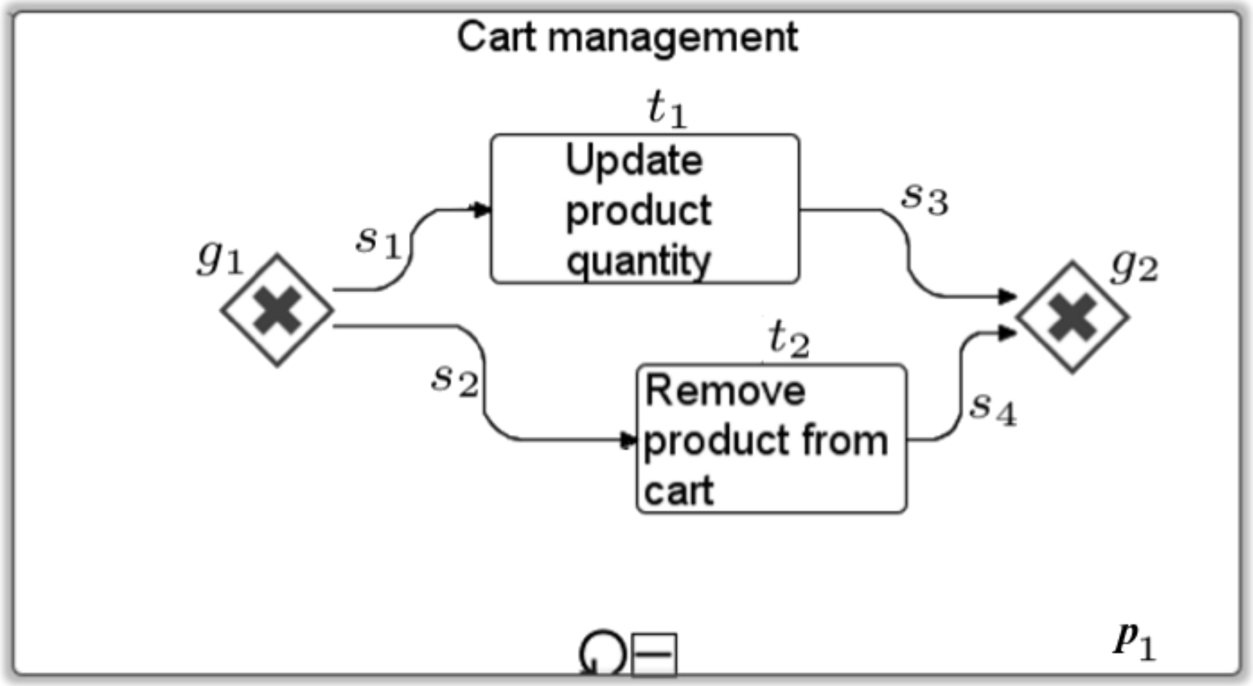
Instantiating the BPMN Ontology

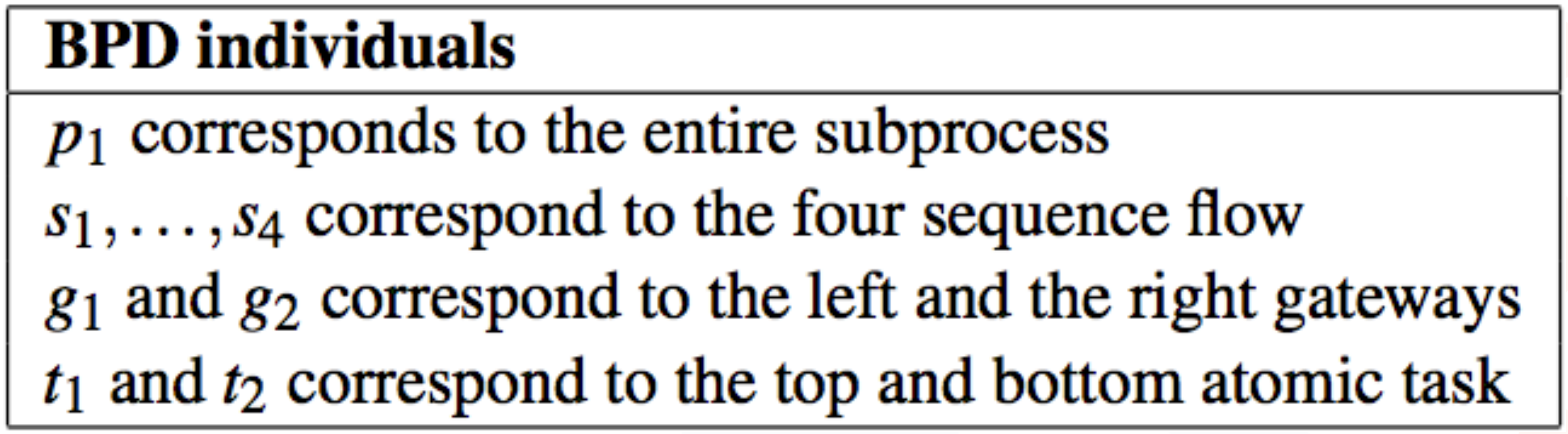
- Given a BPMN business process diagram (BPD), it is possible to represent it as an A-box in the language of the BPMN Ontology
- Example:



Instantiating the BPMN Ontology

Step 1. Create an individual for each element

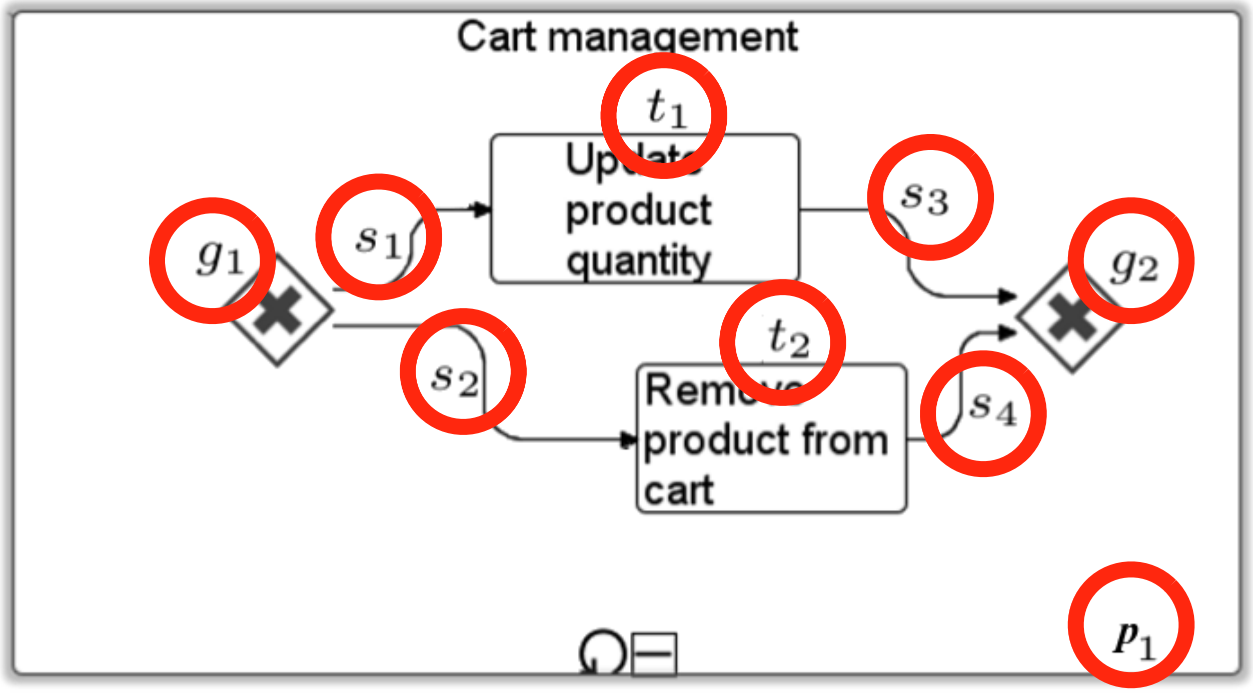




BPD individuals
p_1 corresponds to the entire subprocess
$s_1, \dots, s_4$ correspond to the four sequence flow
g_1 and g_2 correspond to the left and the right gateways
t_1 and t_2 correspond to the top and bottom atomic task

Instantiating the BPMN Ontology

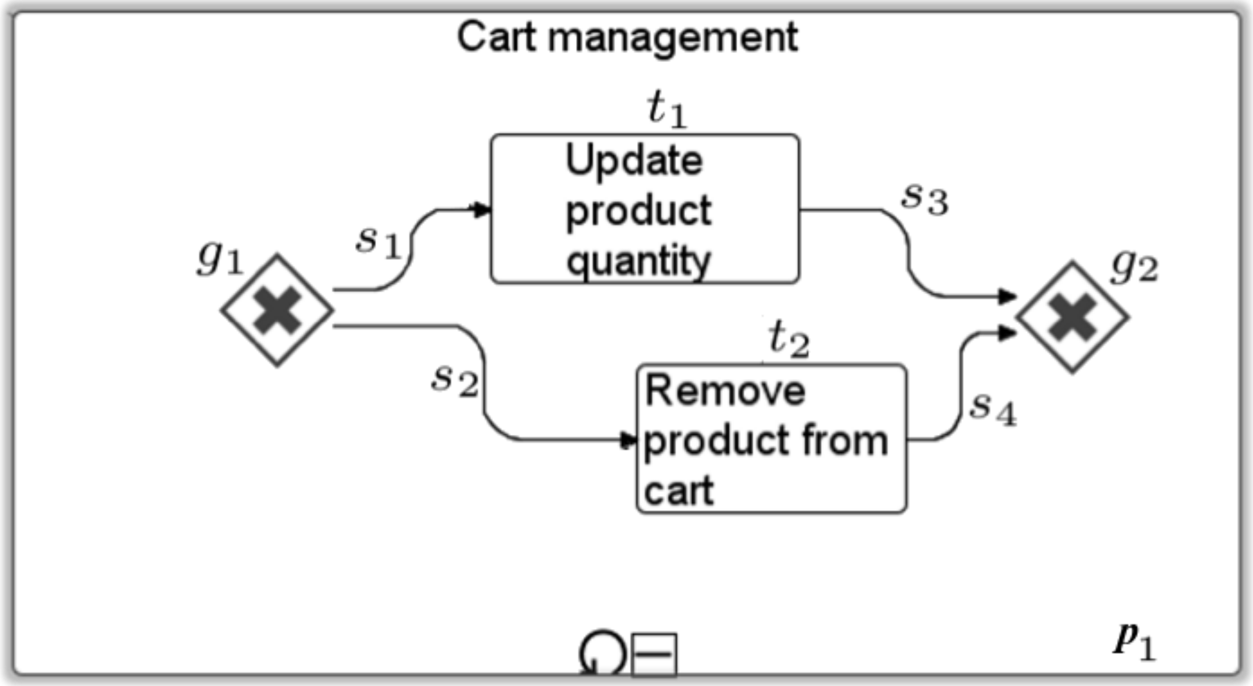
Step 1. Create an individual for each element



BPD individuals
p_1 corresponds to the entire subprocess
$s_1, \ldots, s_4$ correspond to the four sequence flow
g_1 and g_2 correspond to the left and the right gateways
t_1 and t_2 correspond to the top and bottom atomic task

Instantiating the BPMN Ontology

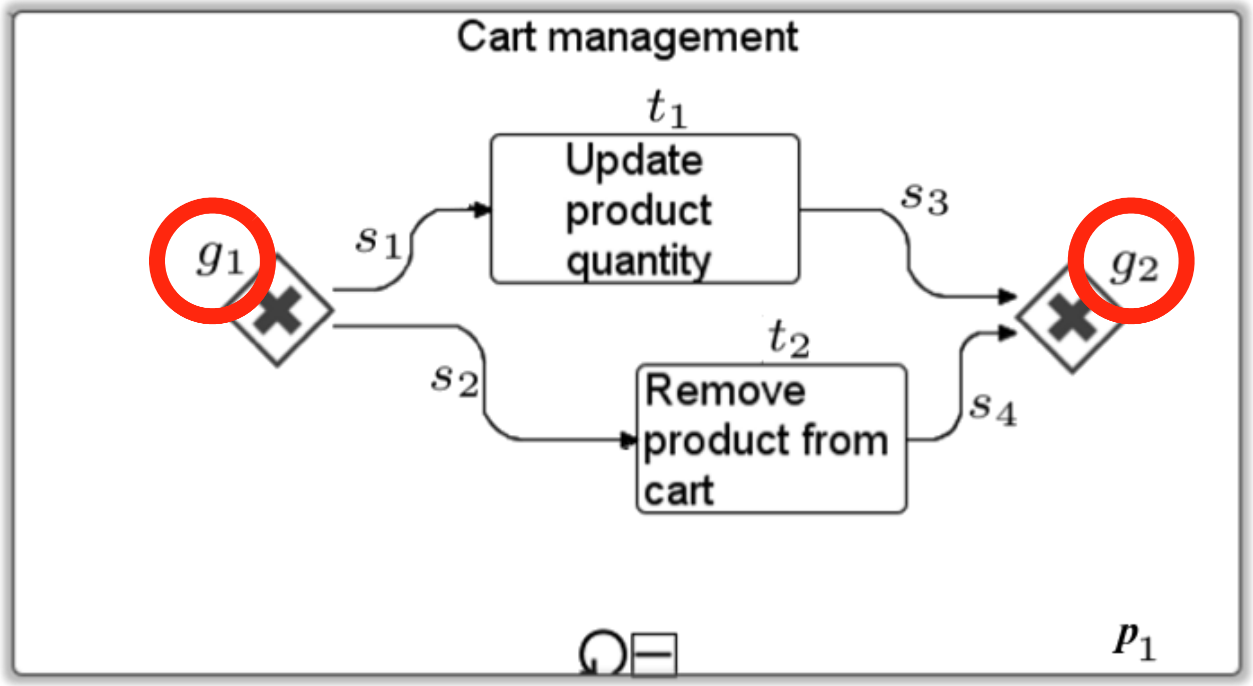
Step 2. Instantiate each individual wrt its BPMN class



Type assertions	
embedded_loop_sub_process(p_1)	
data_based_exclusive_gateway(g_i)	$i = 1,2$
sequence_flow(s_j)	$j = 1,..,4$
task(t_k)	$k = 1,2$

Instantiating the BPMN Ontology

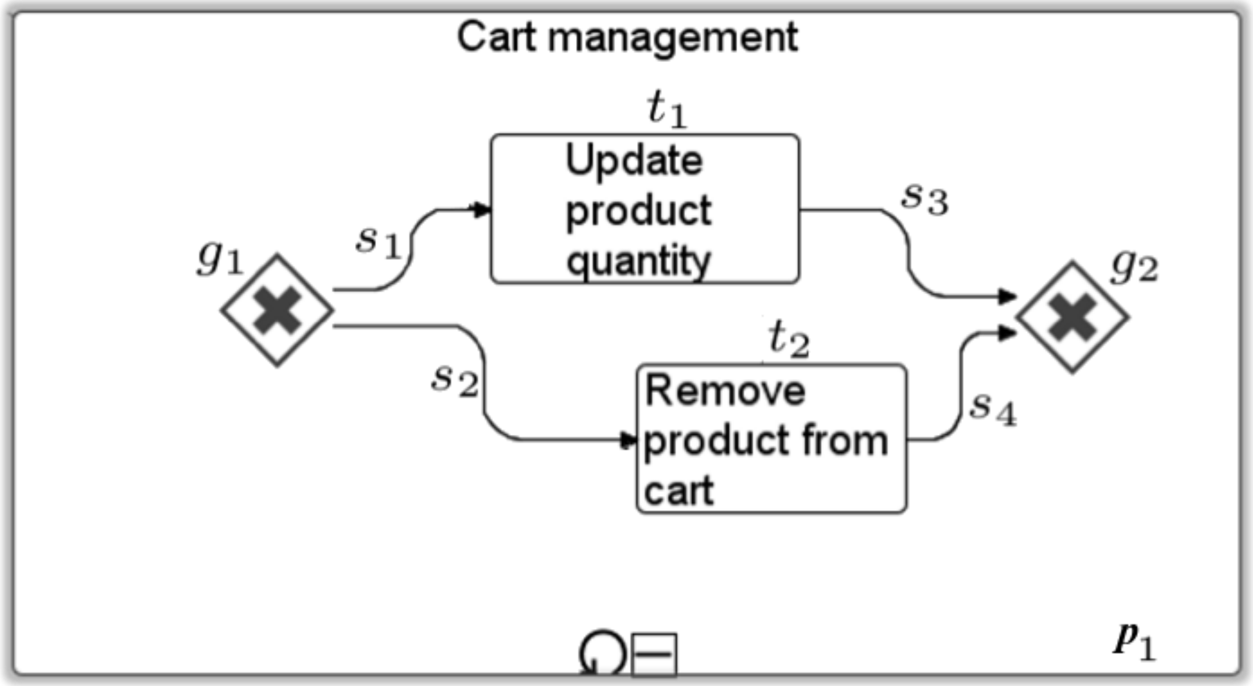
Step 2. Instantiate each individual wrt its BPMN class

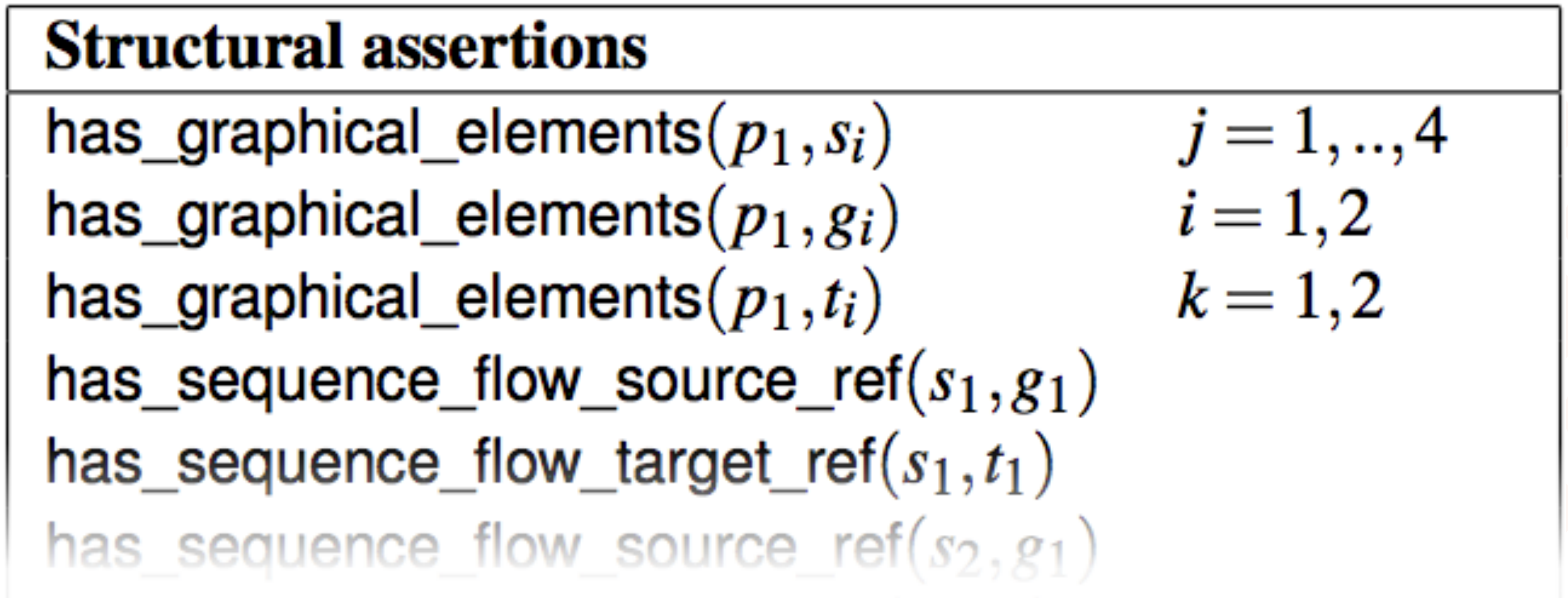


Type assertions	
embedded_loop_sub_process(p_1)	
data_based_exclusive_gateway(g_i)	$i = 1,2$
sequence_flow(s_j)	$j = 1,..,4$
task(t_k)	$k = 1,2$

Instantiating the BPMN Ontology

Step 3. "Compose" the diagram structure in the A-box

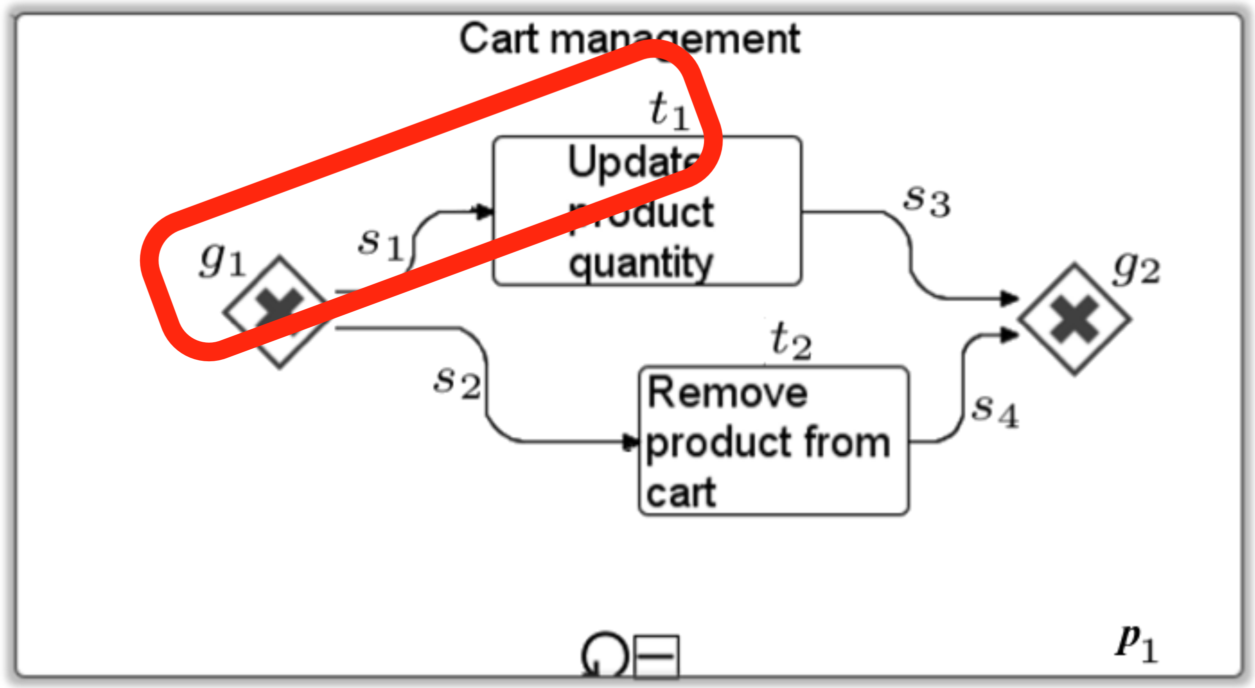




Structural assertions	
has_graphical_elements(p_1, s_i)	$j = 1,..,4$
has_graphical_elements(p_1, g_i)	$i = 1,2$
has_graphical_elements(p_1, t_i)	$k = 1,2$
has_sequence_flow_source_ref(s_1, g_1)	
has_sequence_flow_target_ref(s_1, t_1)	
has_sequence_flow_source_ref(s_2, g_1)	

Instantiating the BPMN Ontology

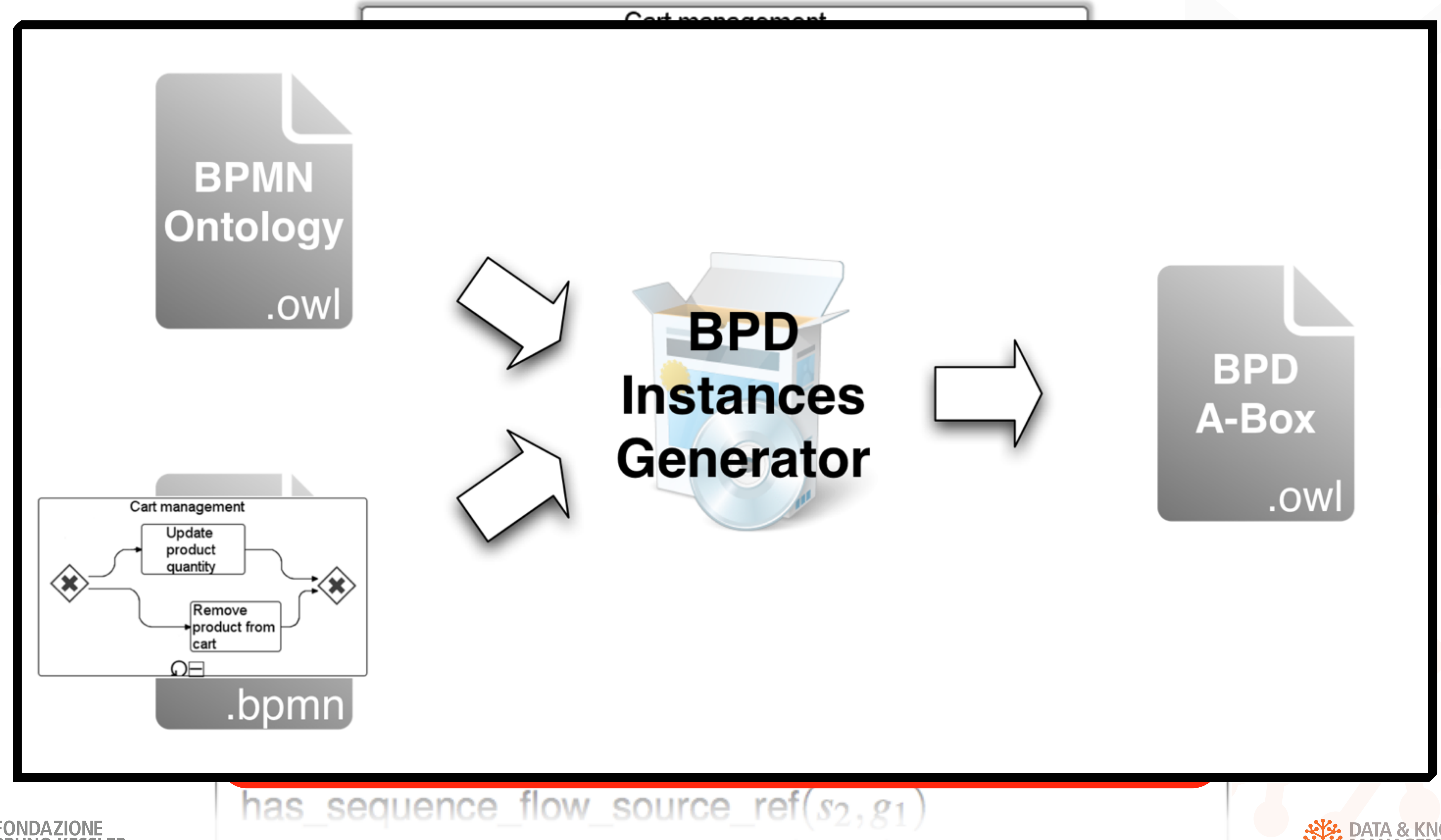
Step 3. "Compose" the diagram structure in the A-box



Structural assertions	
has_graphical_elements(p_1, s_i)	$j = 1,..,4$
has_graphical_elements(p_1, g_i)	$i = 1,2$
has_graphical_elements(p_1, t_i)	$k = 1,2$
has_sequence_flow_source_ref(s_1, g_1)	
has_sequence_flow_target_ref(s_1, t_1)	
has_sequence_flow_source_ref(s_2, g_1)	

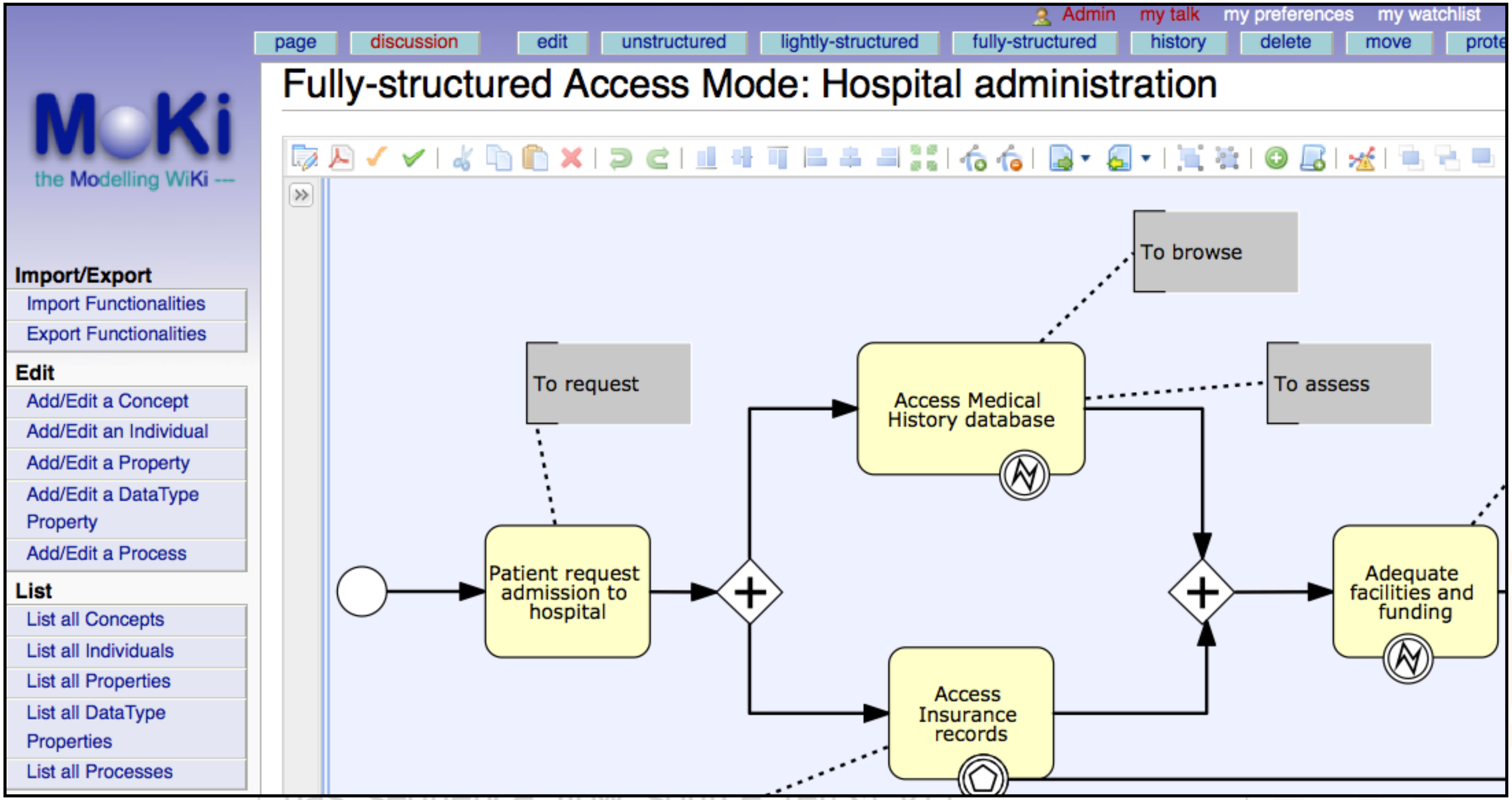
Instantiating the BPMN Ontology

Step 3. "Compose" the diagram structure in the A-box



Instantiating the BPMN Ontology

Step 3. "Compose" the diagram structure in the A-box



Fully-structured Access Mode: Hospital administration

Import/Export
- Import Functionalities
- Export Functionalities

Edit
- Add/Edit a Concept
- Add/Edit an Individual
- Add/Edit a Property
- Add/Edit a DataType Property
- Add/Edit a Process

List
- List all Concepts
- List all Individuals
- List all Properties
- List all DataType Properties
- List all Processes

To browse
To request
Access Medical History database
To assess
Patient request admission to hospital
Adequate facilities and funding
Access Insurance records

Instantiating the BPMN Ontology

Reasoning over an instantiated BPMN Ontology

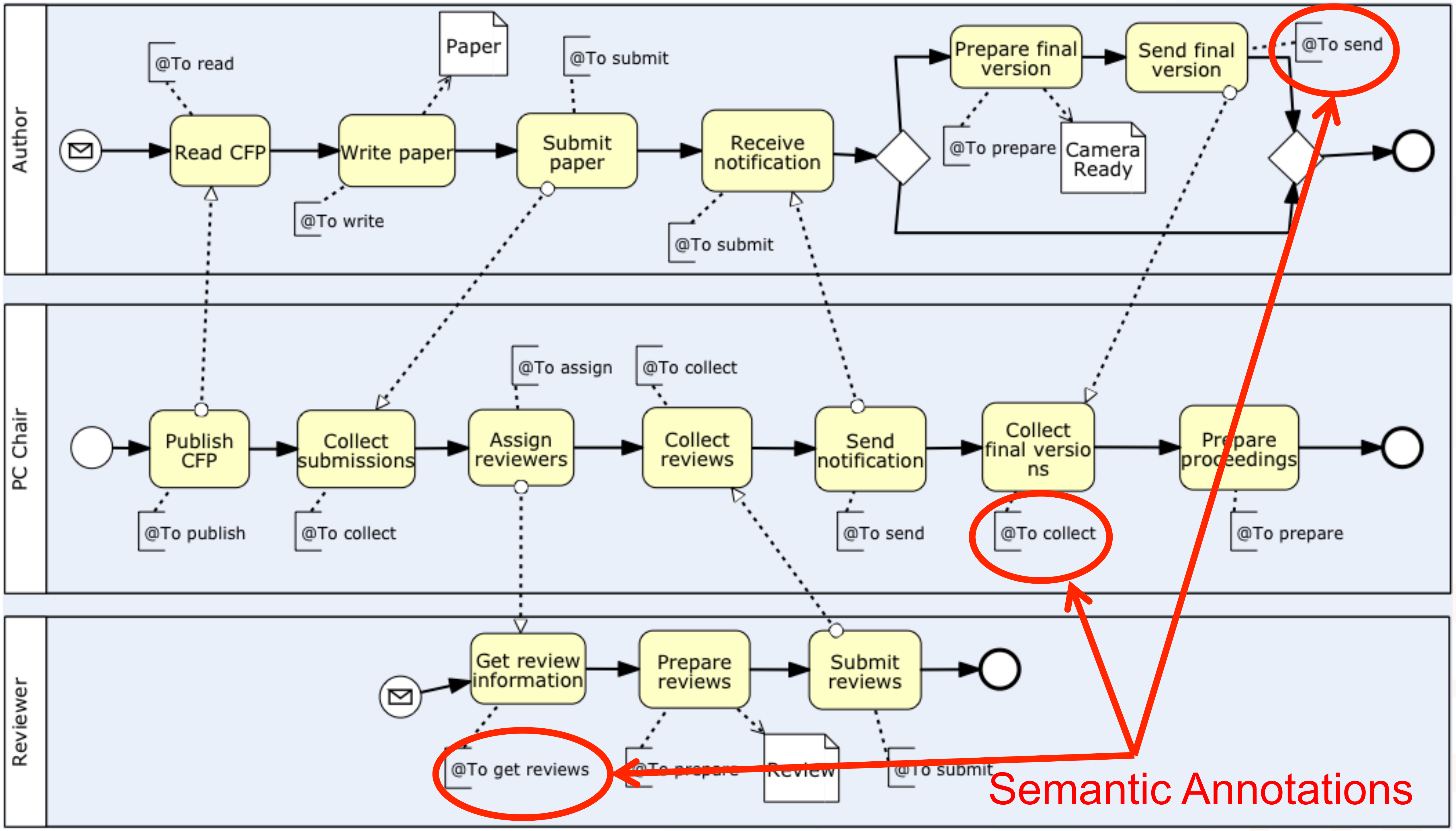
- Query answering on BPMN diagrams (via SPARQL)
  - "Which are the activities which follows gateways and produce a data object?”
  - "Are there sub-processes which do not contain start/end events?"
- Compliance checking of a BPMN diagram against the BPMN Specification
  - e.g.: $Gateway \sqsubseteq (\geq 2) hasSequenceFlowTarget^{-1} \sqcup$
    $((\leq 1) hasSequenceFlowTarget^{-1} \sqcap$
    $(\geq 2) hasGatewayGate)$
  - doable, but in closed-world assumption!

Application Scenarios

- In (Semantic Web) applications where a terminological description of BPMN is needed
  - e.g.: for information retrieval of resources annotated with respect to the elements of the language
- To favour knowledge interoperability (via ontology mappings)
  - e.g.: in a product lifecycle management environment
- To check the compliance of a BPD against the BPMN Specification
  - extendable to domain/application specific constraints
    - e.g., keep the model as simple and readable as possible
      - Gateways in the process may have at most two outgoing sequence flows
    - e.g., ensure that exceptional situations are always handled
      - for every error event attached to the boundary of an activity (i.e., capturing an exception), a sequence flow connecting it to an activity (i.e., an error handling activity) should be provided

Application Scenarios

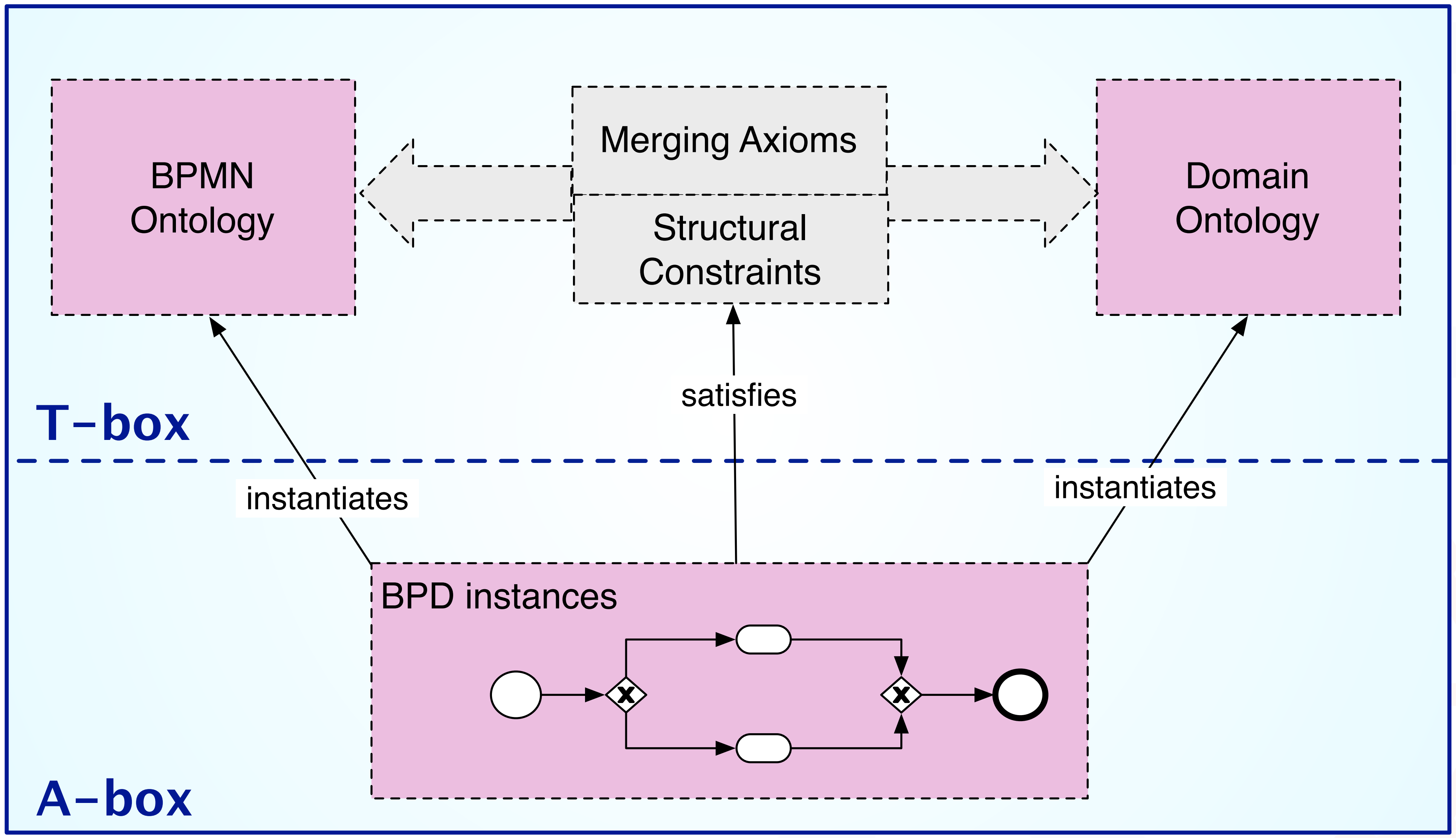
Reasoning on semantically annotated business processes



Application Scenarios

Reasoning on semantically annotated business processes

- Semantically annotated business processes can be encoded into a OWL knowledge base



Application Scenarios

Reasoning on semantically annotated business processes

- The framework also enables to define constraints for
  - correct/incorrect annotation of business process
    - e.g.: A BPMN activity is annotatable only with actions of the domain ontology (and not e.g., with documents)
  - valid critical patters
    - e.g.: the activity of reserving products in the On-line Shop pool has always to catch a "product unavailability" error event

- Using DL-reasoning we can
  - Check compatibility of process constraints
  - Verify constraints over an annotated process
  - Perform queries combining domain and BPMN semantics

Conclusions

- We presented an ontological (OWL-DL) formalization of the BPMN specification
- The proposed formalization:
  - provides a terminological description of the language;
  - enables representing any actual BPMN diagram as a set of individuals and assertions on them
    - enables several reasoning-based services
- Available on-line:
  - dkm.fbk.eu (~25/30 download per month)
  - ontohub.org
- Current / future work:
  - cover BPMN 2.02

FONDAZIONE BRUNO KESSLER

DKM DATA & KNOWLEDGE MANAGEMENT

Thank you! Questions?

Marco Rospocher

Fondazione Bruno Kessler, Data and Knowledge Management Unit
Trento, Italy
rospocher@fbk.eu :: http://dkm.fbk.eu/rospocher