Building the BPMN ontology

Marco Rospocher

http://dkm.fbk.eu/rospocher - rospocher@fbk.eu

DATA & KNOWLEDGE MANAGEMENT

FONDAZIONE BRUNO KESSLER

Outline

- The Business Process Modelling Notation (BPMN)
- Building The BPMN ontology
  - Formalization Process
  - Limitations
  - Details on the ontology
- Using the BPMN ontology

Business Process Modelling Notation (BPMN)

- State of the art (graphical) language for the specification of business processes.
- The development of BPMN was driven by two major requirements:
  - had to be **acceptable and usable** by business analysts in the business community;
  - had to support the generation of **executable processes** from the notation provided.
- Current Stable version: **v2.0** (this talk refers to **v1.1**).

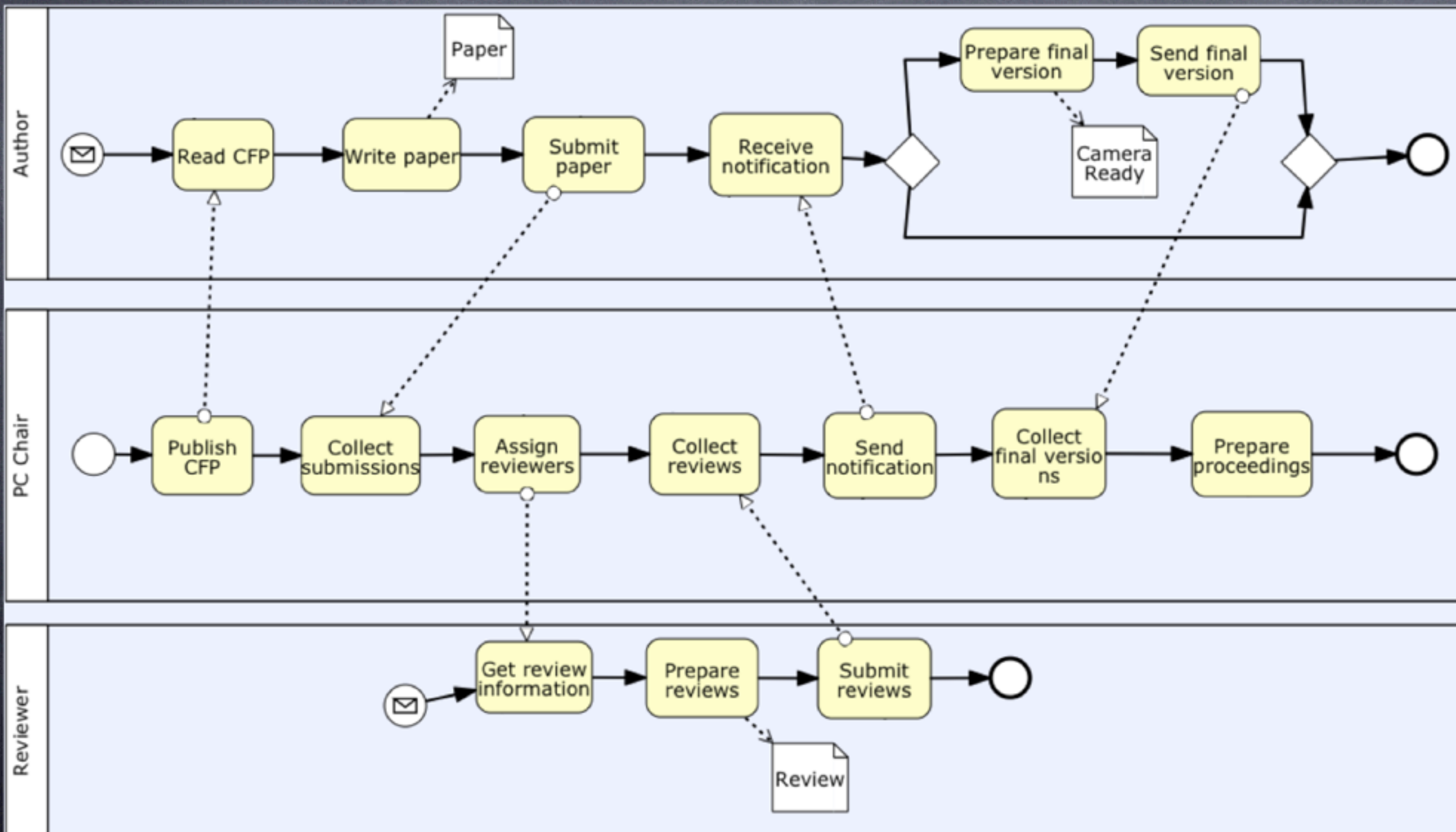
Business Process Modelling Notation (BPMN)

- State of the art (graphical) language for the specification of business processes.
- The development of BPMN was driven by two major requirements:
  - had to be **acceptable and usable** by business analysts in the business community;

    BPMN supports users in writing business processes using a simple and intuitive graphical notation
  - had to support the generation of **executable processes** from the notation provided.

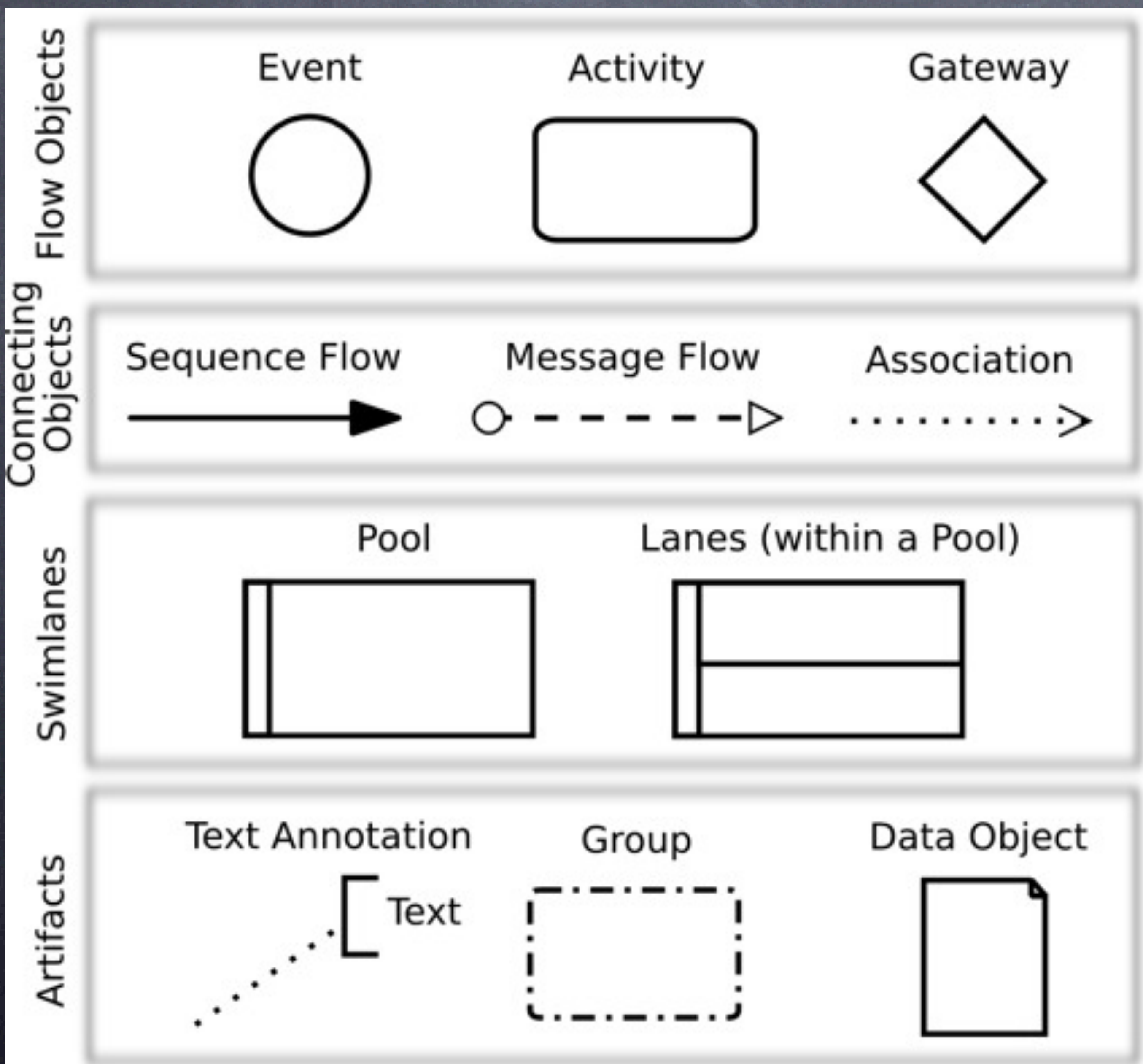
    Mapping of the language to the Business Process Execution Language (BPEL)
- Current Stable version: **v2.0** (this talk refers to **v1.1**).

Business Process Modelling Notation (BPMN)



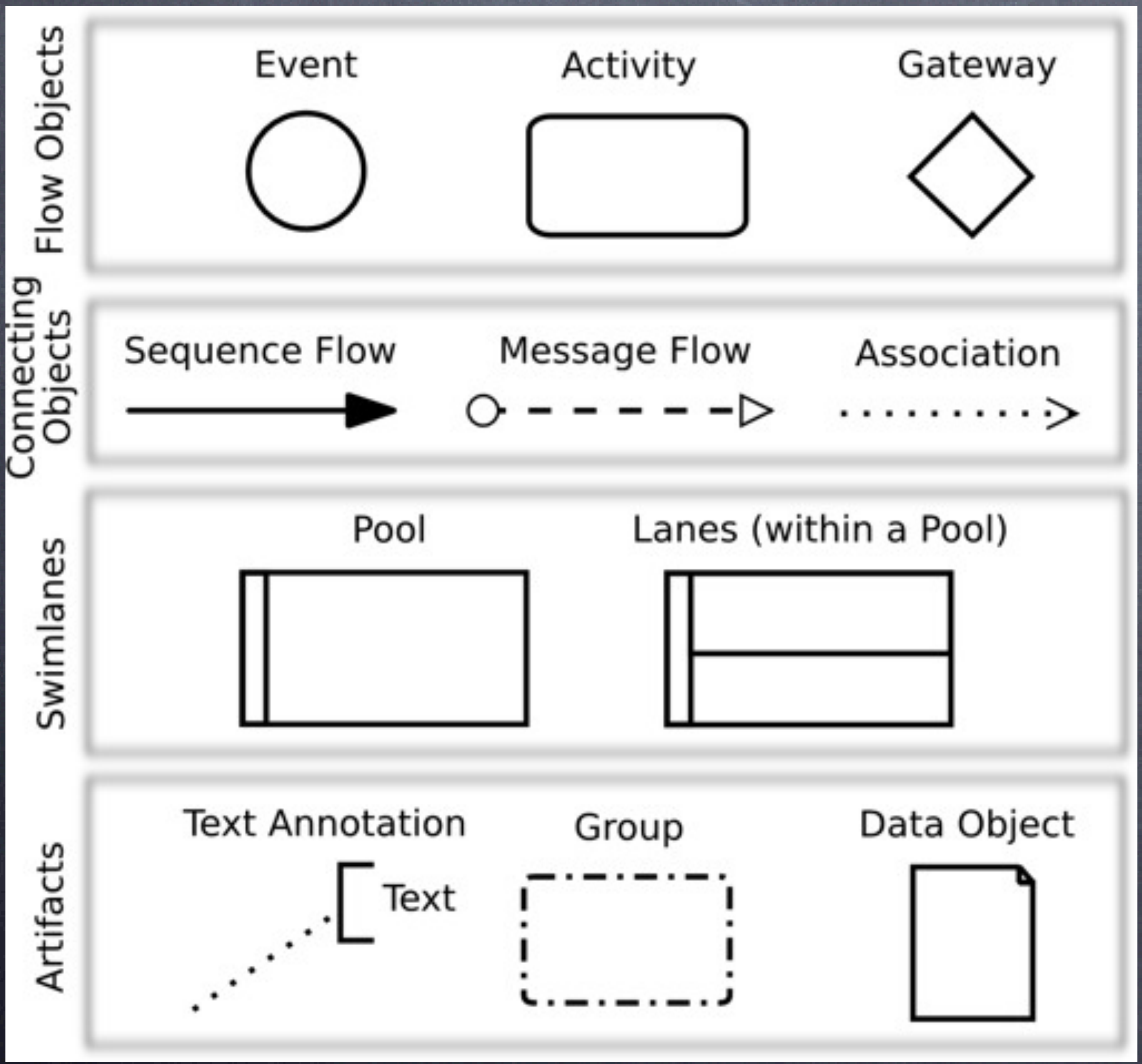
Business Process Modelling Notation (BPMN)

Core Element Set



Business Process Modelling Notation (BPMN)

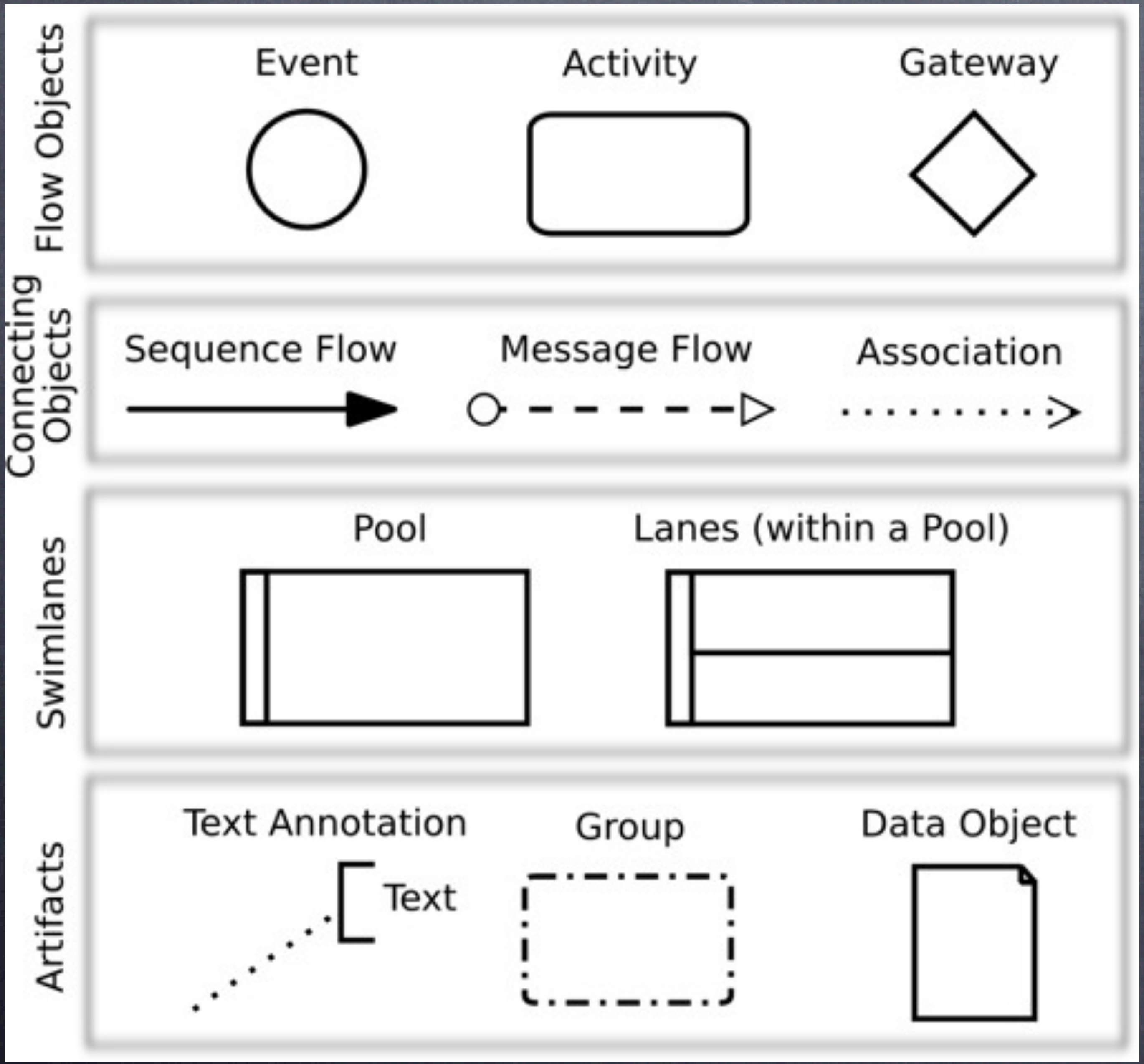
Core Element Set



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

Business Process Modelling Notation (BPMN)

Core Element Set

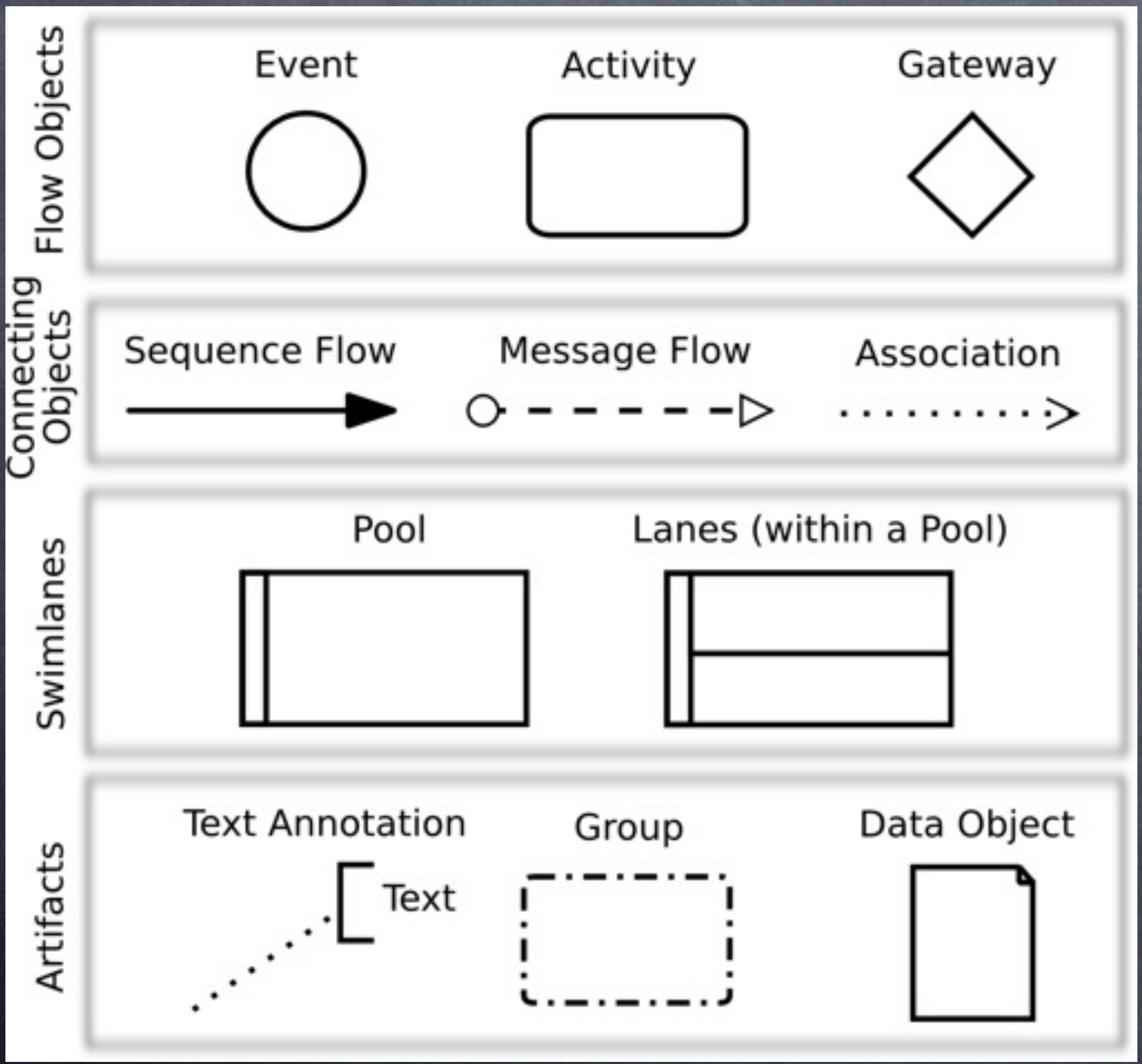


for representing something that happens (**event**), work to be performed (**activity**), and control flow elements (**gateway**);

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

Business Process Modelling Notation (BPMN)

Core Element Set



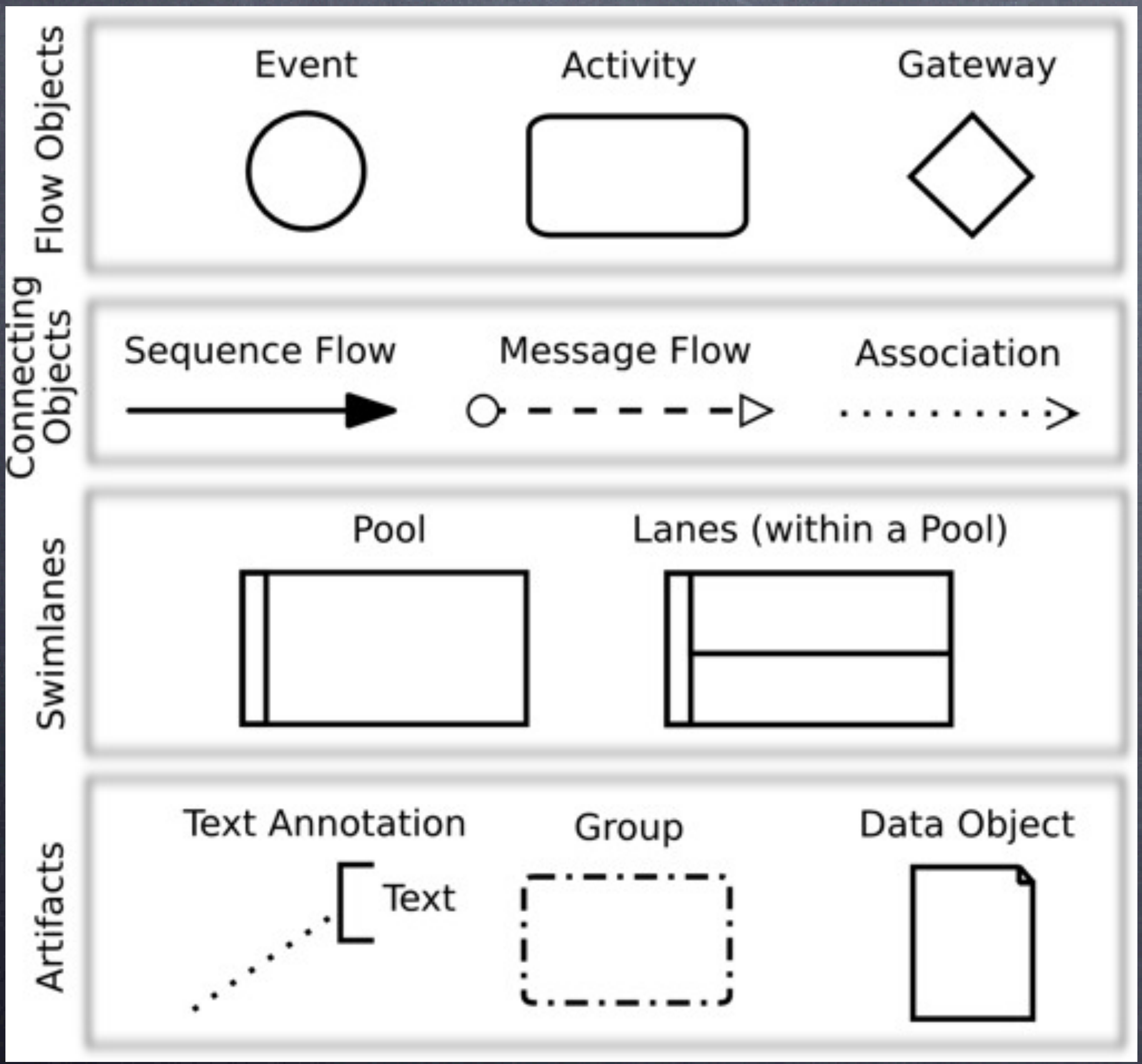
for representing something that happens (**event**), work to be performed (**activity**), and control flow elements (**gateway**);

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**);

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**), ...

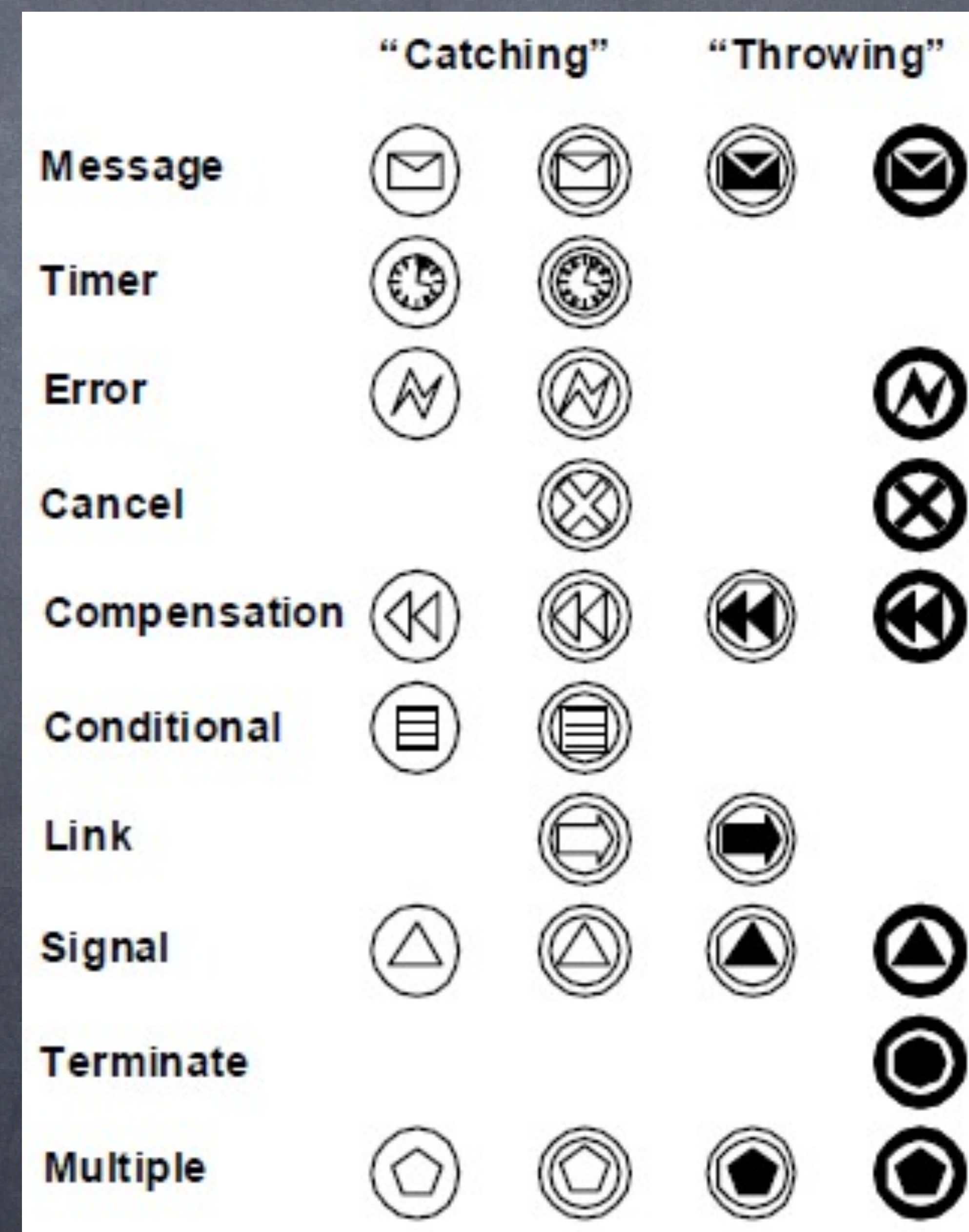
for describing participants in a process (**pool**), and to organize and categorize activities (**lane**);

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)

- For each element, the **BPMN Specification Document (OMG)** provides:
  - an **introductory description** of the element, together with some general properties and conditions about it;
  - a compact tabular description of each **element's attribute** (its name, its value type, its multiplicity details, conditions to for its instantiation);
  - a detailed description of the **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**.
- In this talk: BPMN v1.1 [OMG] 2008-01-17.

An ontology for BPMN

- An **ontological** (OWL-DL) **formalization** of BPMN.
- 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 BPMN business process** compliant with the BPMN Specification.
- The proposed formalization:
  - provides a **terminological description** of the language;
  - enables representing any actual **BPMN process as a set of individuals** and assertions on them (for e.g. **compliance checking** of a process to the **BPMN specification** via ontological reasoning).

Building The BPMN ontology

- A **three phases** formalization process, supported by the BPMN Specification document:

    1. **identification and classification** of all the elements of the language;
    2. formalization of the **attributes** of each element;
    3. formalization of the **conditions** concerning the usage of the elements of the language to **compose a BPMN diagram**.

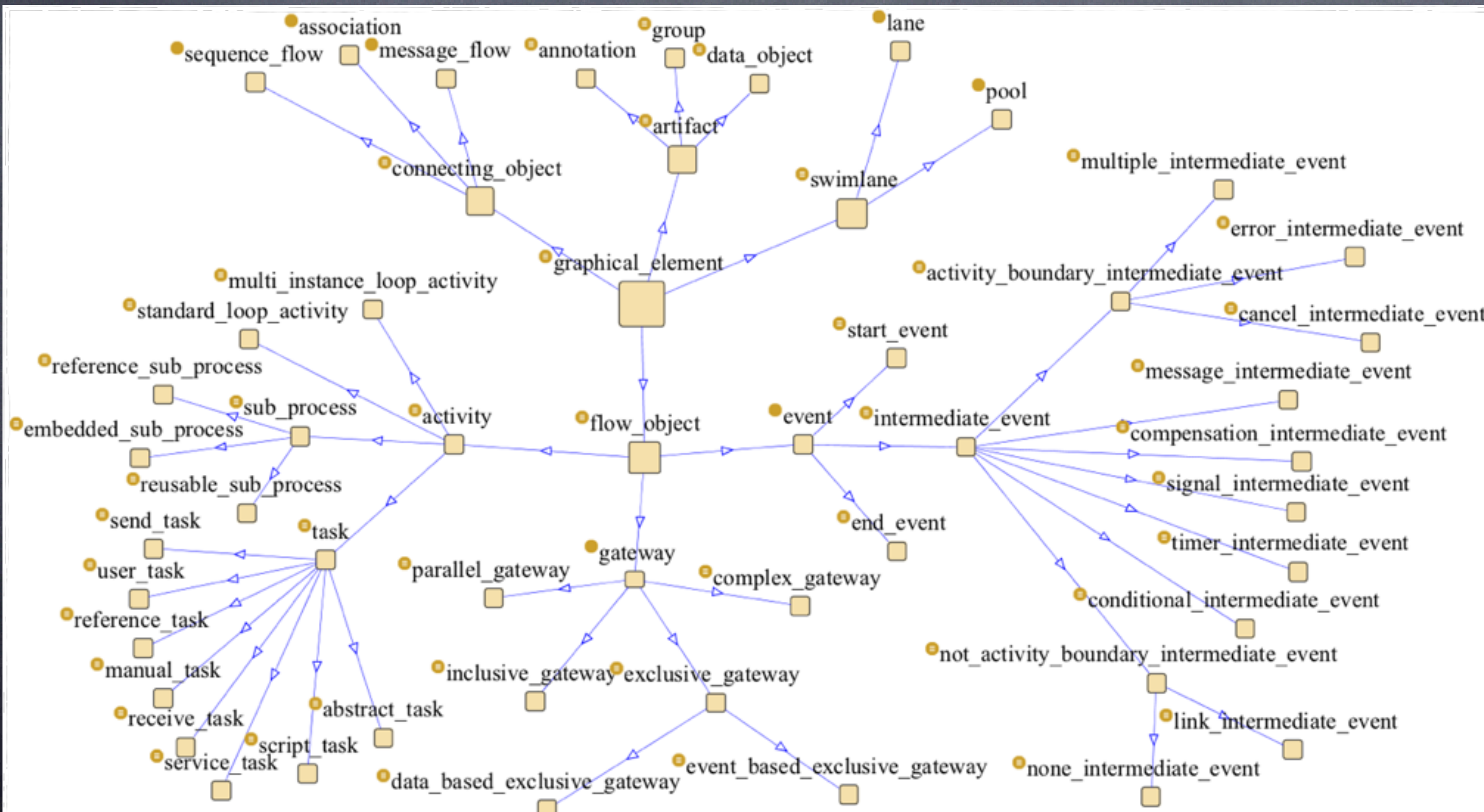
Building The BPMN ontology

Formalization Process - PHASE 1

- 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.

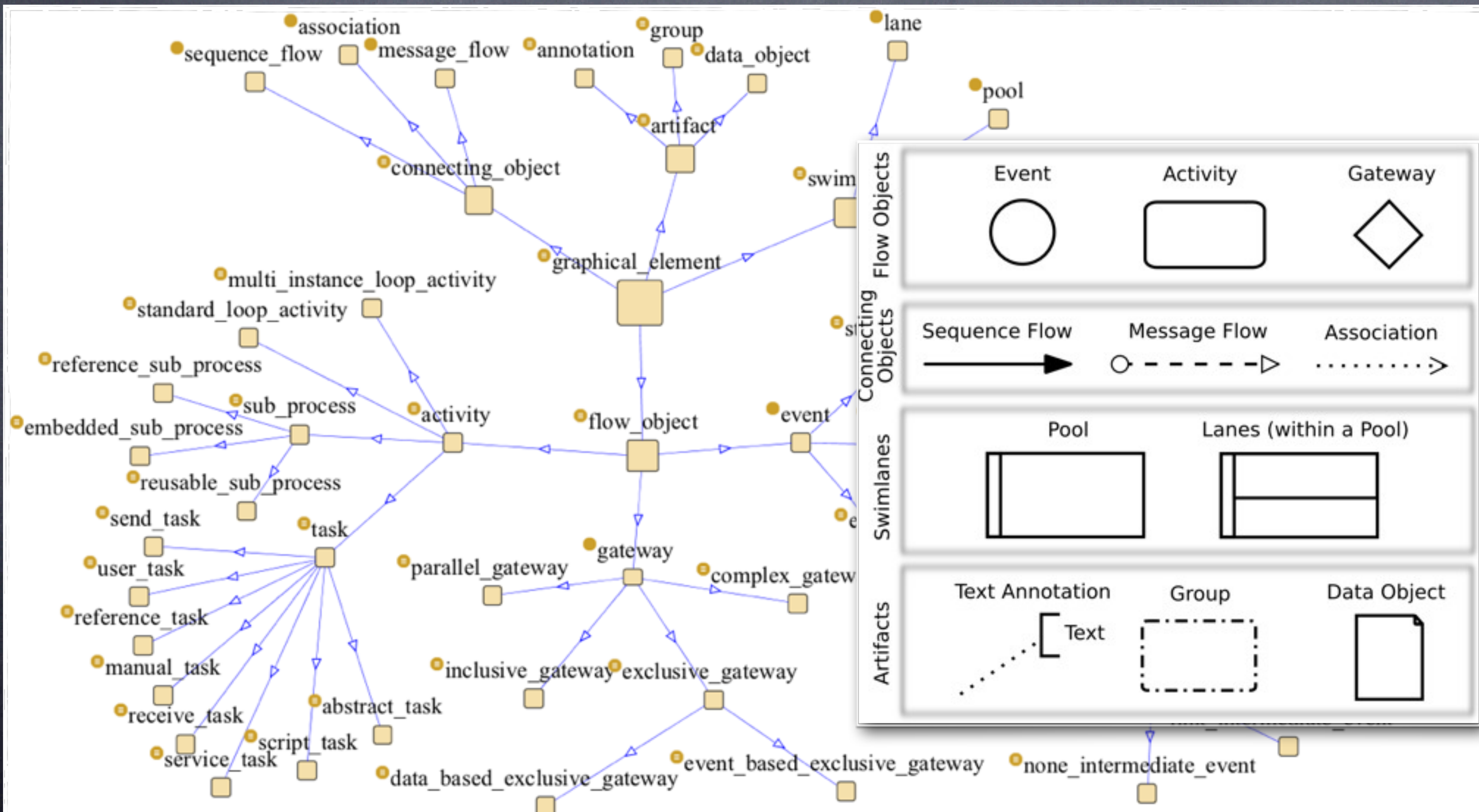
Building The BPMN ontology

Formalization Process - PHASE 1



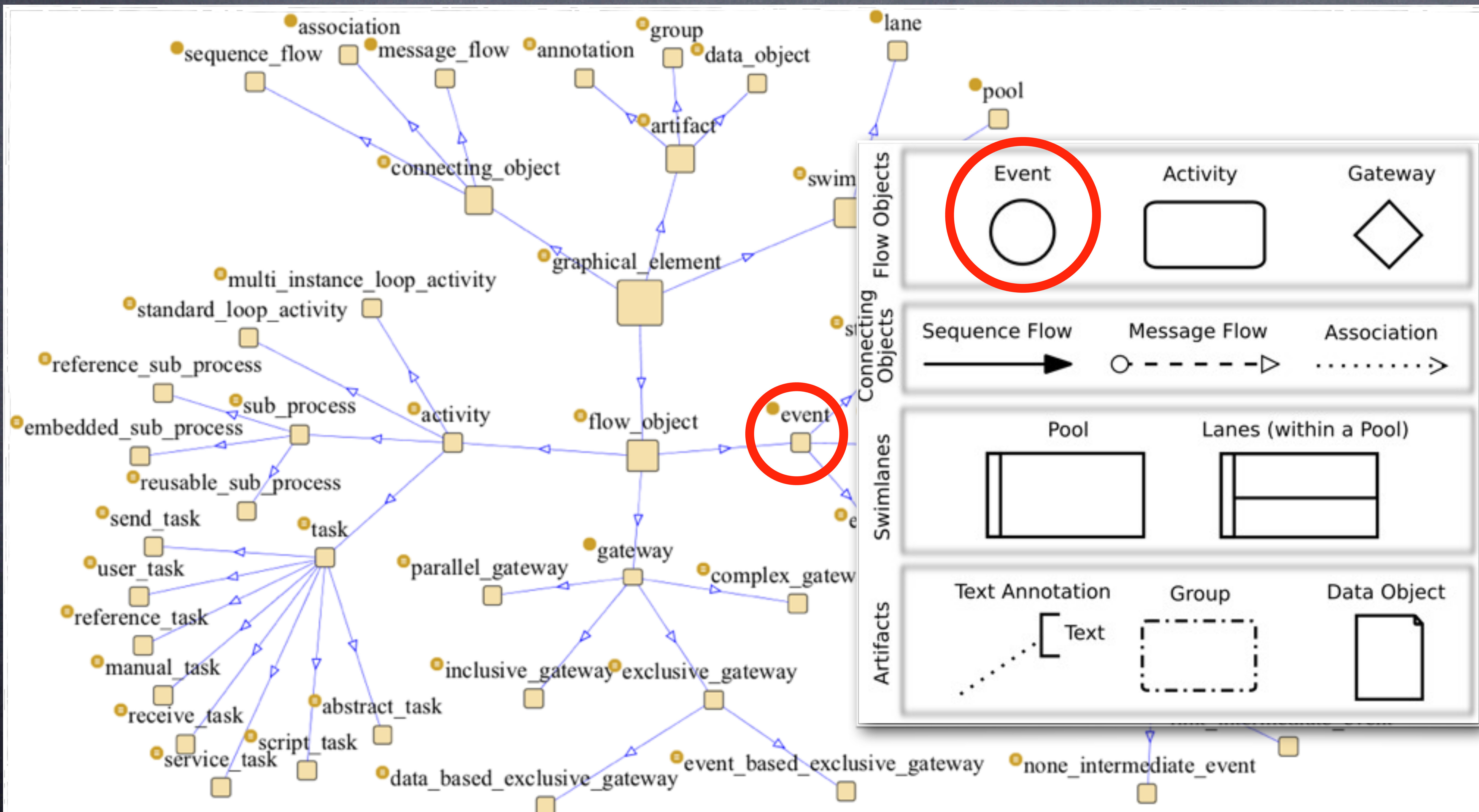
Building The BPMN ontology

Formalization Process - PHASE 1



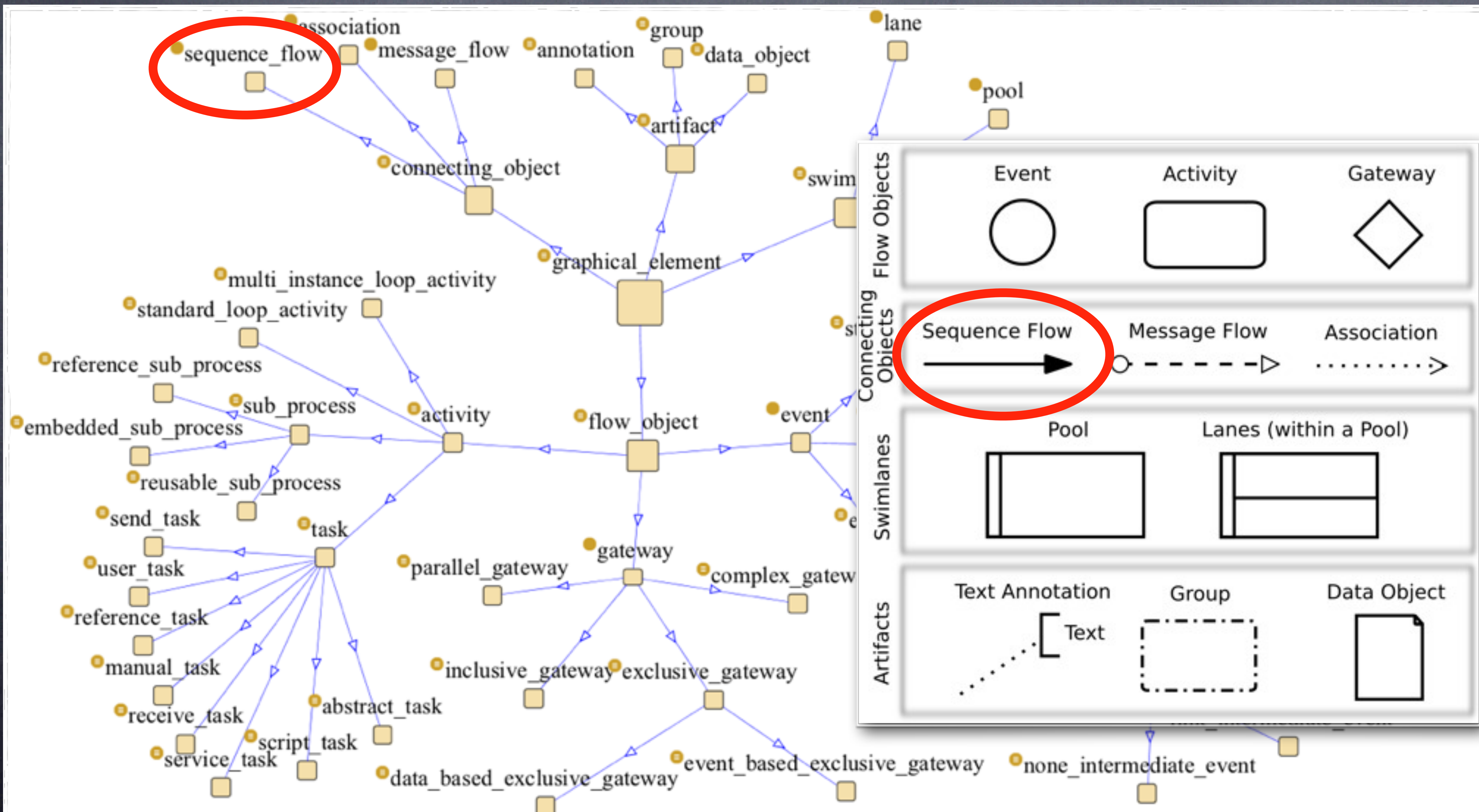
Building The BPMN ontology

Formalization Process - PHASE 1



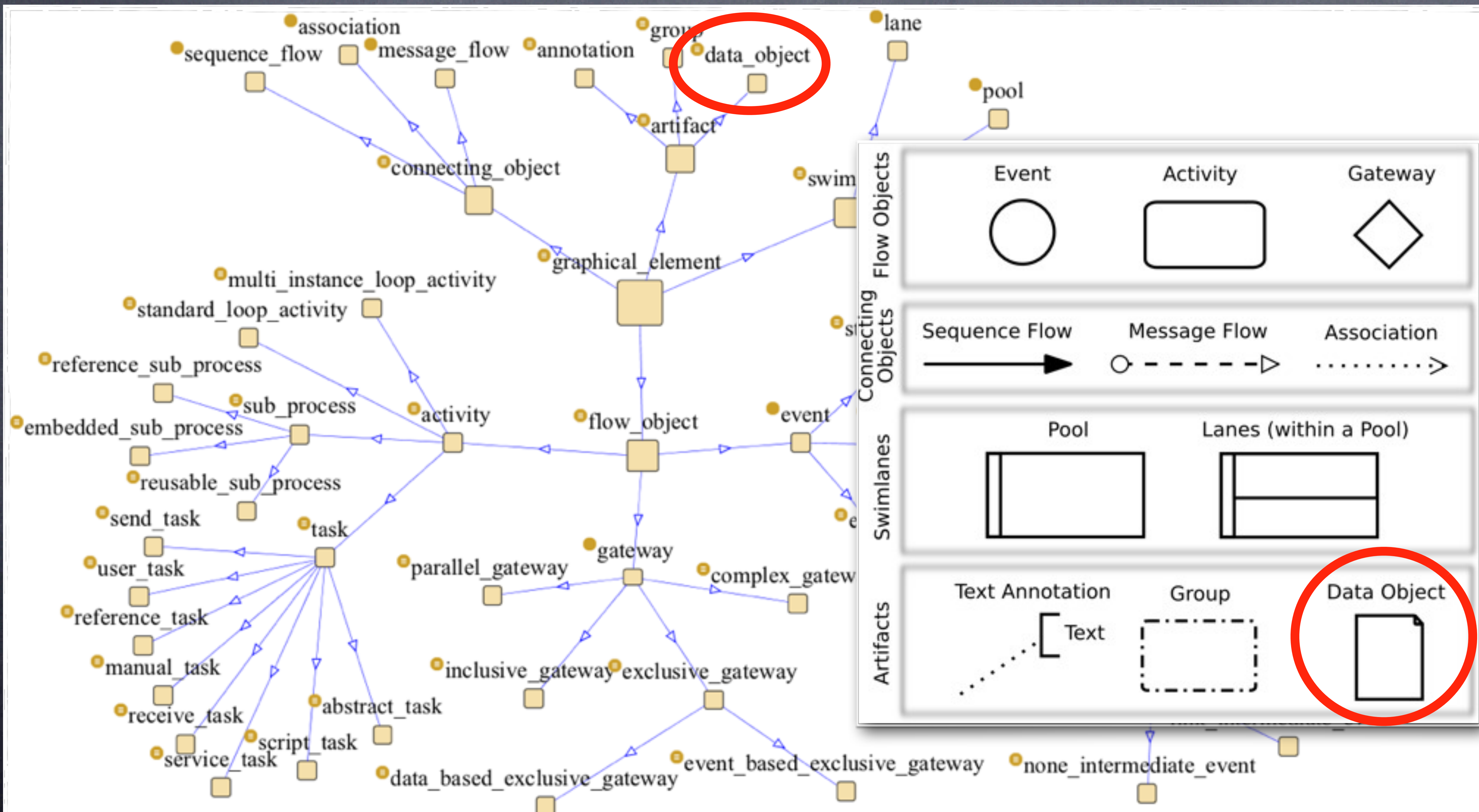
Building The BPMN ontology

Formalization Process - PHASE 1



Building The BPMN ontology

Formalization Process - PHASE 1



Building The BPMN ontology

Formalization Process - PHASE 2

- Formalization of the **attributes of each element**

[p.121]

10.1.1 Common Connecting Object Attributes

The following table displays the set of attributes common to Connecting Objects (Sequence Flow, Message Flow, and Association), and which extends the set of common BPMN Element attributes (see Table 10.1):

Table 10.1 - Common Connecting Object Attributes

Attributes	Description
Name (0-1) : String	Name is an optional attribute that is text description of the Connecting Object.
SourceRef : Graphical Element	SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected *from*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.
TargetRef : Graphical Element	TargetRef is an attribute that identifies which Graphical Element the Connecting Object is connected *to*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

Building The BPMN ontology

Formalization Process - PHASE 2

- Formalization of the **attributes of each element**

[p.121]

10.1.1 Common Connecting Object Attributes

The following table displays the set of attributes common to Connecting Objects (Sequence Flow, Message Flow, and Association), and which extends the set of common BPMN Element attributes (see Table 10.1):

Table 10.1 - Common Connecting Object Attributes

Attributes	Description
Name (0-1) : String	Name is an optional attribute that is text description of the Connecting Object.
SourceRef **Graphical Element**	SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected *from*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.
TargetRef : Graphical Element	TargetRef is an attribute that identifies which Graphical Element the Connecting Object is connected *to*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

Building The BPMN ontology

Formalization Process - PHASE 2

- Formalization of the **attributes of each element**

[p.121]

10.1.1 Common Connecting Object Attributes

The following table displays the set of attributes common to Connecting Objects (Sequence Flow, Message Flow, and Association), and which extends the set of common BPMN Element attributes (see Table 10.1):

Table 10.1 - Common Connecting Object Attributes

Attributes	Description
Name (0-1) : String	Name is an optional attribute that is text description of the Connecting Object.
SourceRef **Graphical Element**	SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected *from*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.
TargetRef : Graphical Element	TargetRef is an attribute that identifies which Graphical Element the Connecting Object is connected *to*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

Building The BPMN ontology

Formalization Process - PHASE 2

- Formalization of the **attributes of each element**

[p.121]

10.1.1 Common Connecting Object Attributes

The following table displays the set of attributes common to Connecting Objects (Sequence Flow, Message Flow, and Association), and which extends the set of common BPMN Element attributes (see Table 10.1):

Table 10.1 - Common Connecting Object Attributes

Attributes	Description
Name (0-1) : String	Name is an optional attribute that is text description of the Connecting Object.
SourceRef : **Graphical Element**	SourceRef is an attribute that identifies which Graphical Element the Connecting Object is connected *from*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.
TargetRef : Graphical Element	TargetRef is an attribute that identifies which Graphical Element the Connecting Object is connected *to*. Note: there are restrictions as to what objects Sequence Flow and Message Flow can connect. Refer to the Sequence Flow Connections section and the Message Flow Connections section for each Flow Object, Swimlane, and Artifact.

$$\exists hasConnectingObjectSourceRef.\top \sqsubseteq ConnectingObject$$

$$\top \sqsubseteq \forall hasConnectingObjectSourceRef.GraphicalElement$$

Building The BPMN ontology

Formalization Process - PHASE 2

- An **attribute** is formalized either as **datatype** property or as an **object** property.
- **Three situations** considered:
  - I. the value type of the attribute is another BPMN element;
  - 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;
  - III. the value type of the attribute is a datatype with no restriction.

Building The BPMN ontology

Formalization Process - PHASE 2

- **Case I**: The value type of the attribute is another **BPMN element.**
- Formalization: as **object property:**
  - domain: the class having the attribute;
  - range: the class of the element mentioned as value type of the attribute.
- Example (**Target** attribute of **Intermediate Event** [p.47]):

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.

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

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

Building The BPMN ontology

Formalization Process - PHASE 2

- **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.
- Formalization: as **object property:**
  - domain: the class having the attribute;
  - range: a new class enumerating all possible values of the attribute.
- Example (**AdHocOrdering** attribute of **Embedded Sub-Process** [p.47]):

[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.

$$\exists hasESPAdHocOrdering.\top \sqsubseteq EmbeddedSubProcess$$
$$\top \sqsubseteq \forall hasESPAdHocOrdering.AdHocOrderingType$$
$$AdHocOrderingType \equiv \{parallel, sequential\}$$

Building The BPMN ontology

Formalization Process - PHASE 2

- **Case III**: The value type of the attribute is **a datatype with no restriction.**
- Formalization: as **datatype property:**
  - domain: the class having the attribute;
  - range: a datatype compatible with the value type of the attribute.
- Example (**Text** attribute of **Text Annotation** [p.95]):

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

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

Building The BPMN ontology

Formalization Process - PHASE 2

- 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** [p.94]):

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.

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

Building The BPMN ontology

Formalization Process - PHASE 2

- For each attribute, we also encode **additional conditions ruling the usage** of the attribute.
  - formalization case by case;
- 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.

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

Building The BPMN ontology

Formalization Process - PHASE 3

- Formalization of the **conditions** concerning the usage of the elements of the language **to compose a BPMN diagram**.
- Examples [p.48,p.72]:

◦ 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.

- The formalization of these conditions was performed case by case (due to their variety).

Building The BPMN ontology

Formalization Process - PHASE 3

◦ 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.

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

Building The BPMN ontology

Limitations

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

Building 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.

- Ontologies are **not** particualrly **suitable** to model the dynamic behaviour of business process, **other formalism** are more adequate (e.g. **PetriNets**).

Building The BPMN ontology

Limitations - Attributes default values

- Attributes default values: **the value taken** by the required attributes **when** they are **not explicitly assigned** in the process 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** (we recall that attributes are formalized as properties in the BPMN Ontology).

Building The BPMN ontology

Limitations - “Undecideable” conditions

- Due to **DL expressiveness limitation** and the fact that we wanted to remain in a **decidable fragment of OWL,** there is a **limited number of conditions** which are **not represented**.
- Example [p.73]

OutgoingSequenceFlow : SequenceFlow	For Exclusive Data-Based, and Inclusive Gateways: 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, the one connecting the gateways to sequence flows and the one associating a condition to a sequence flow, but imposing **number restrictions on property chains** (or, more generally, on complex roles) **lead to undecidability**.

Building The BPMN ontology

Details on the ontology

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

Building The BPMN ontology

Details on the ontology

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

The BPMN ontology

Details on the ontology (on-line version)

- The **BPMN Ontology** [Ghidini, Rospocher, Serafini] is **publicly available**:

  https://dkm.fbk.eu/index.php/BPMN_Ontology

- A **textual description** in terms of DL is also available.

- Built using the **latex2owl** syntax:

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

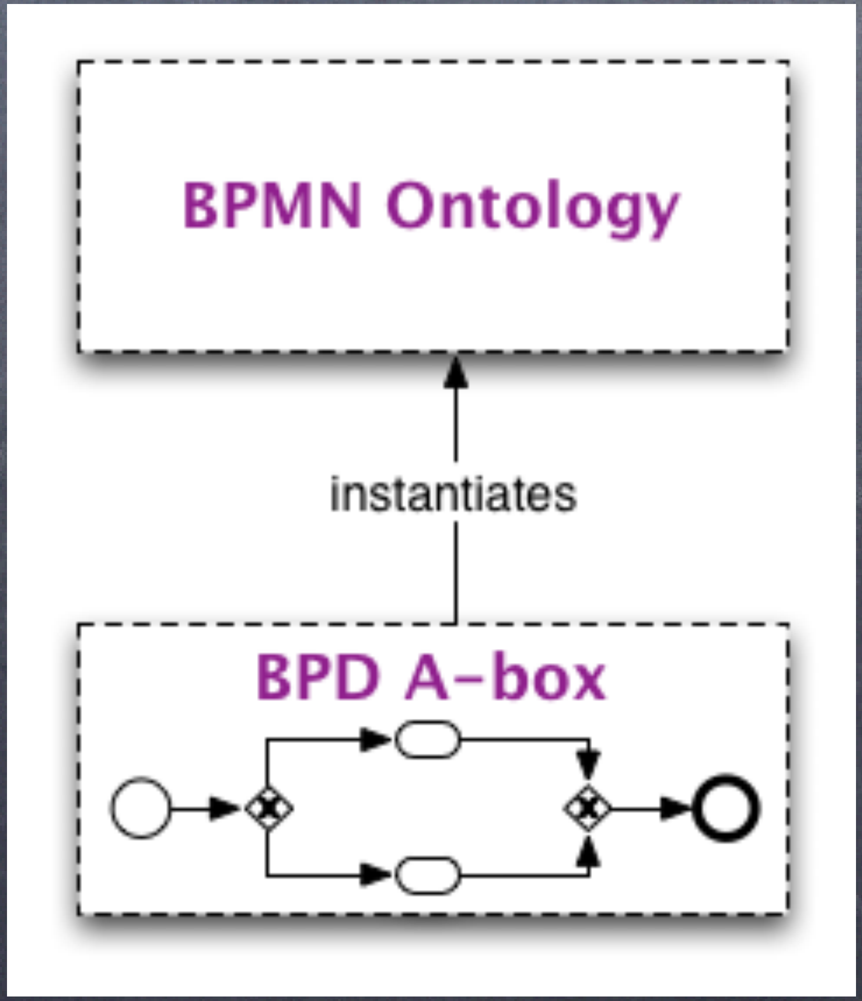
```
EndEvent \cisa (\not \exists hasResult.Error) \cor
                (\exists hasResult.(Error \cand \exists hasErrorCode))
```

- It covers **version 1.1** of the BPMN Specification (version 2.0 in on the way...).

Using The BPMN ontology

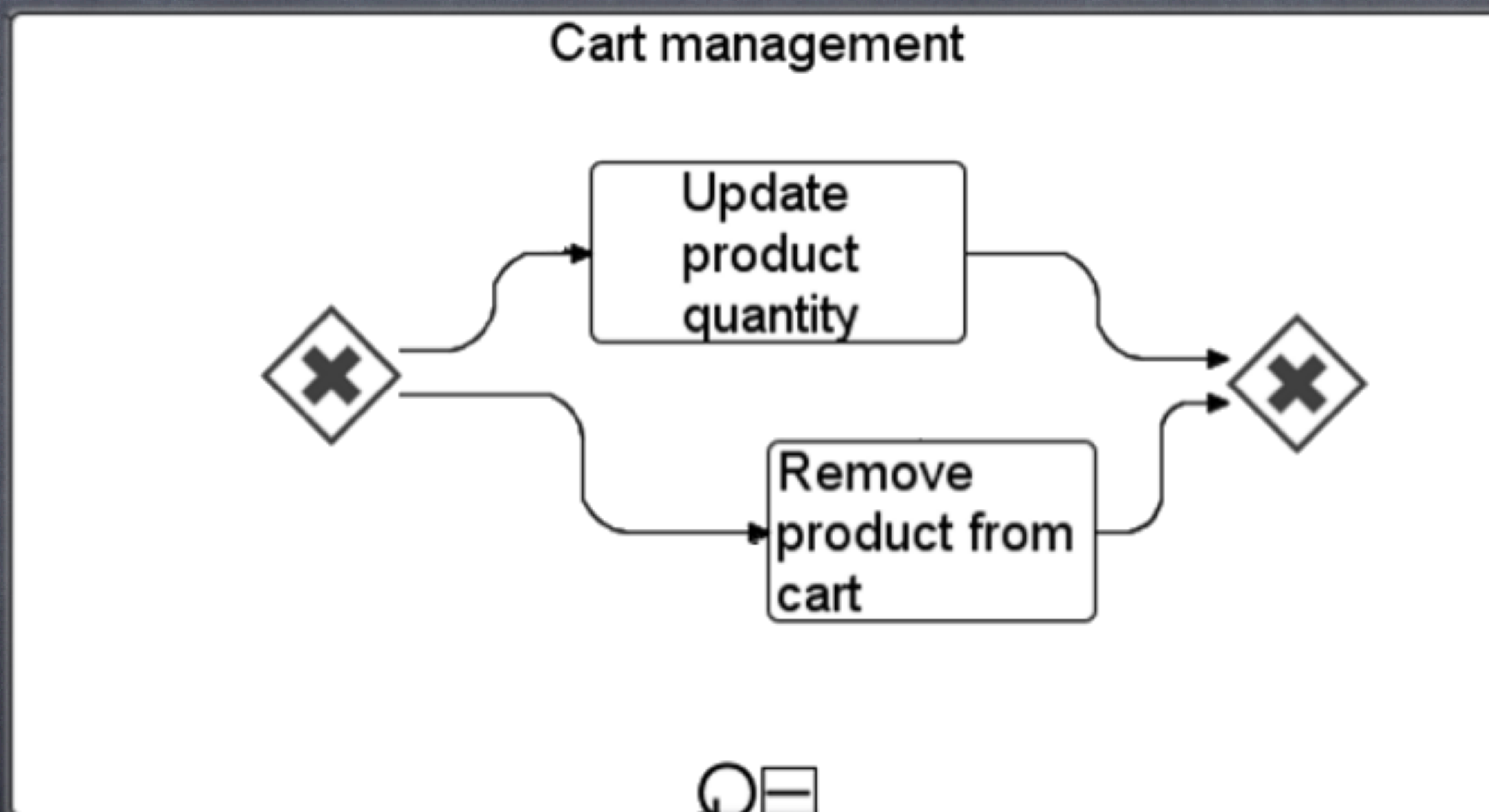
Instantiating a BPMN diagram

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



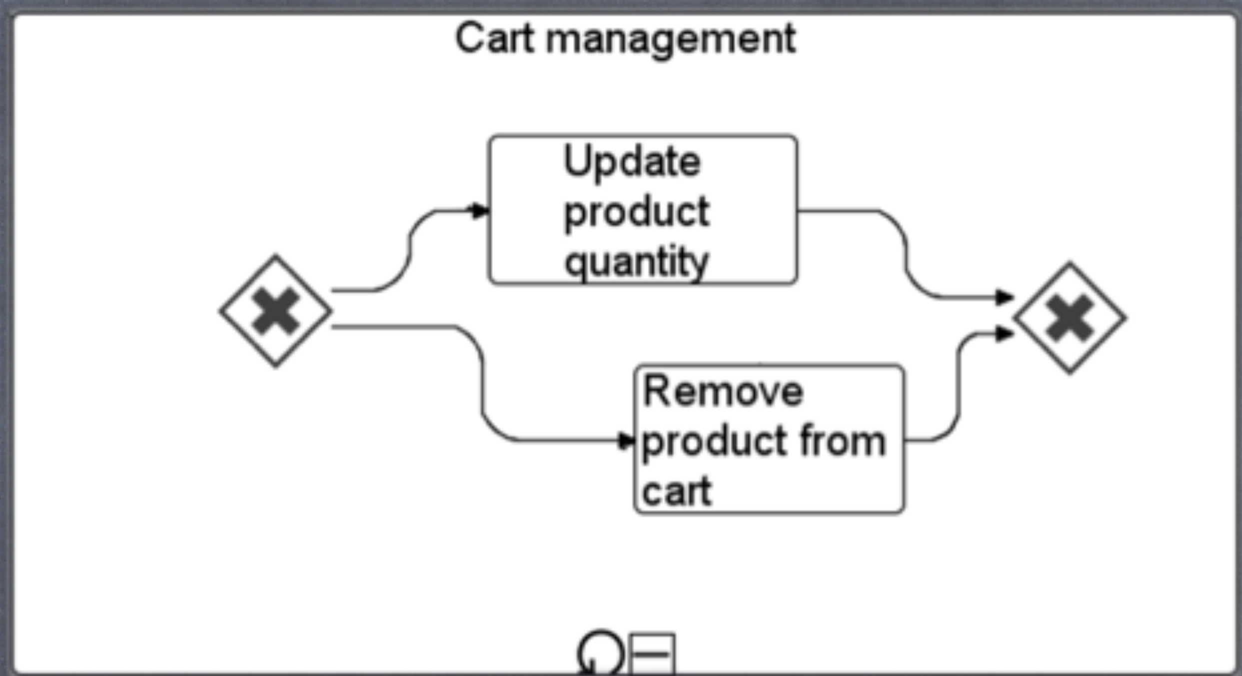
Using The BPMN ontology

Instantiating a BPMN diagram - example



Using The BPMN ontology

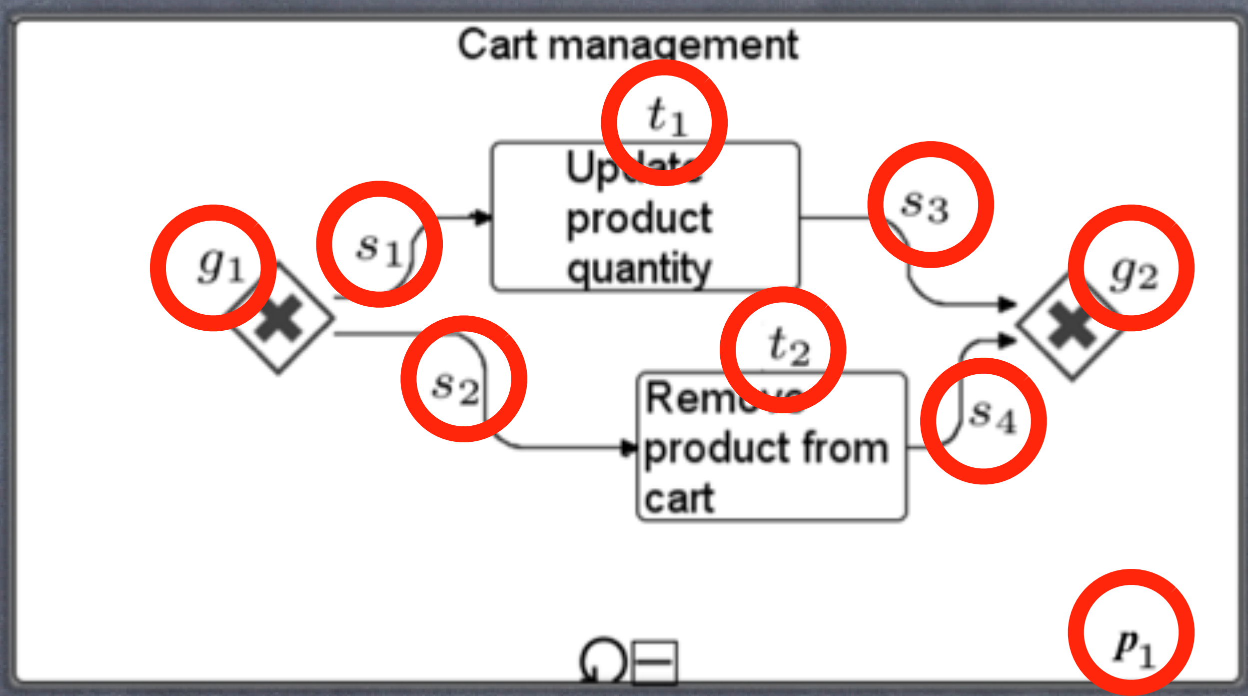
1. Create an individual for each element



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

Using The BPMN ontology

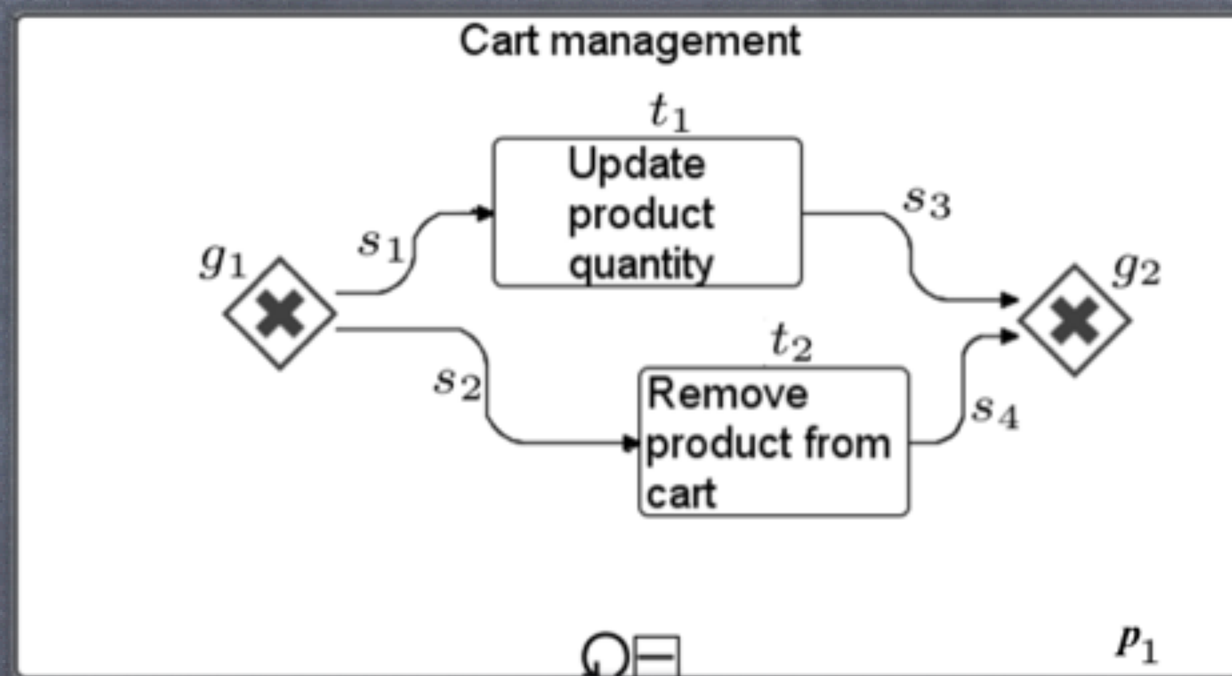
1. Create an individual for each element



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

Using The BPMN ontology

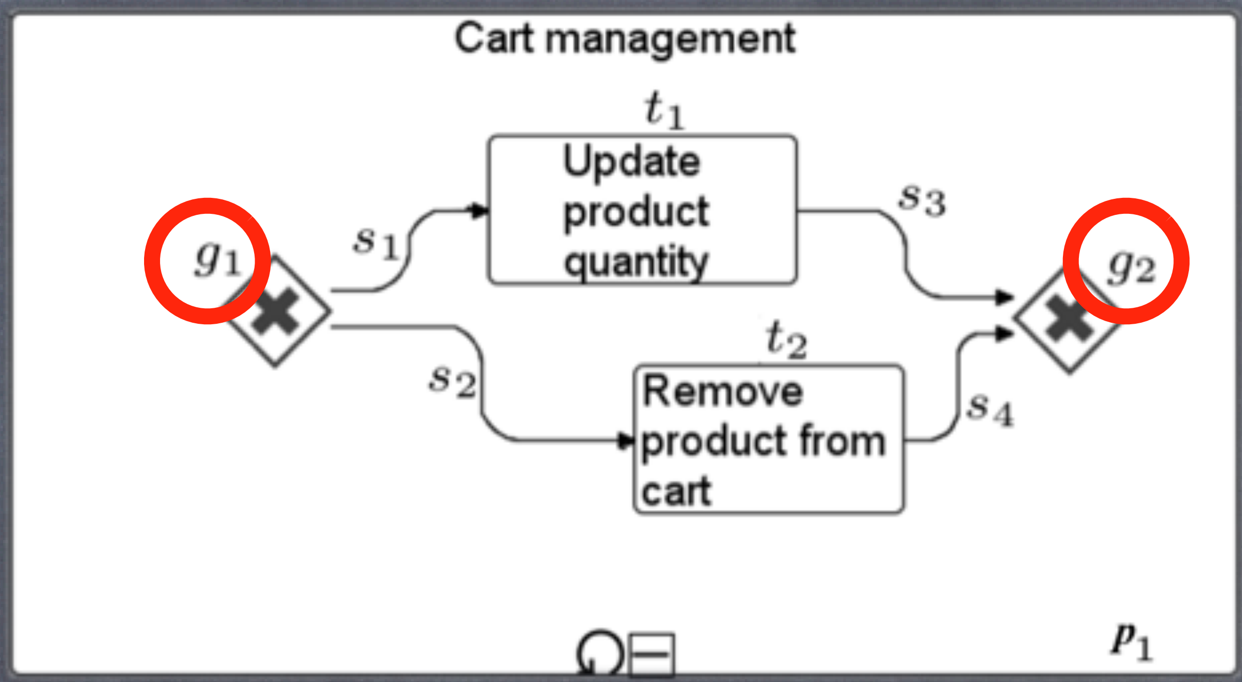
2. Instantiate each individual wrt its BPMN element class



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)	$k = 1,2$

Using The BPMN ontology

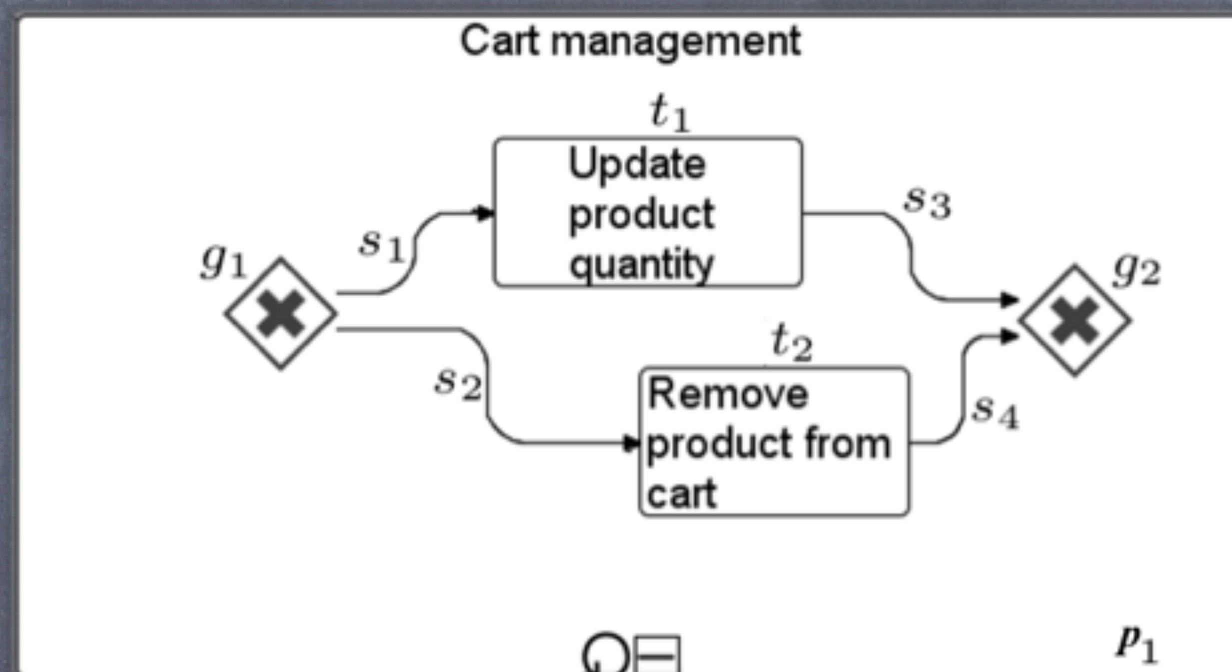
2. Instantiate each individual wrt its BPMN element class



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)	$k = 1, 2$

Using The BPMN ontology

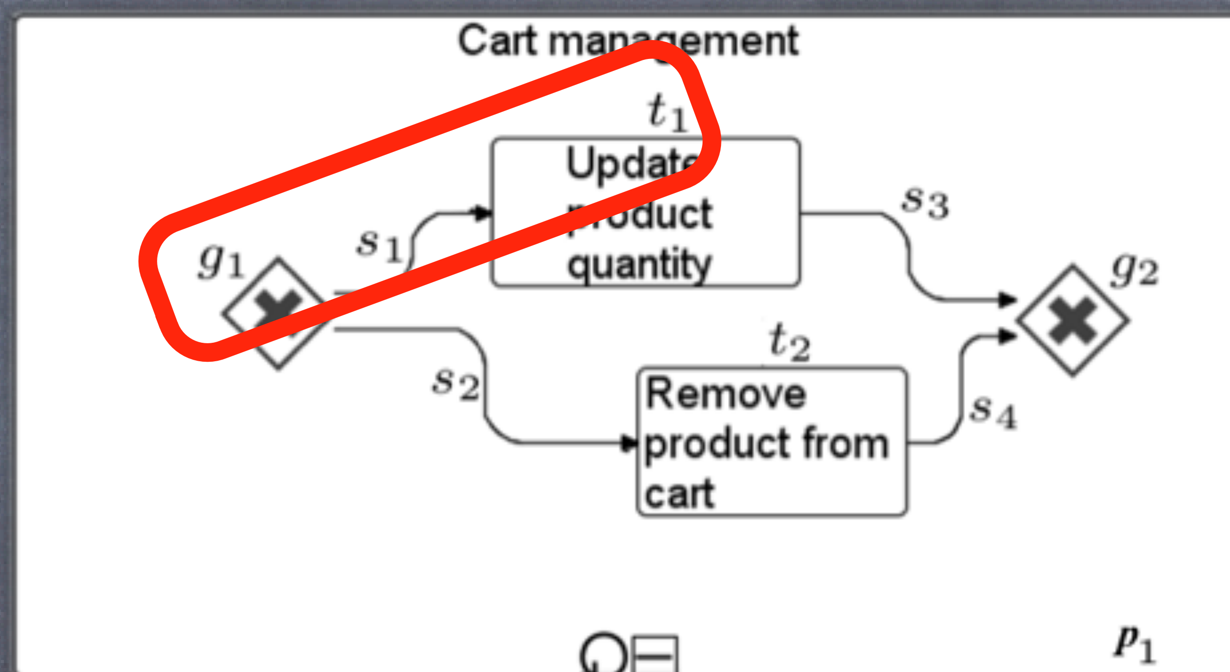
3. "Compose" the diagram structure in the A-box



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)	

Using The BPMN ontology

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

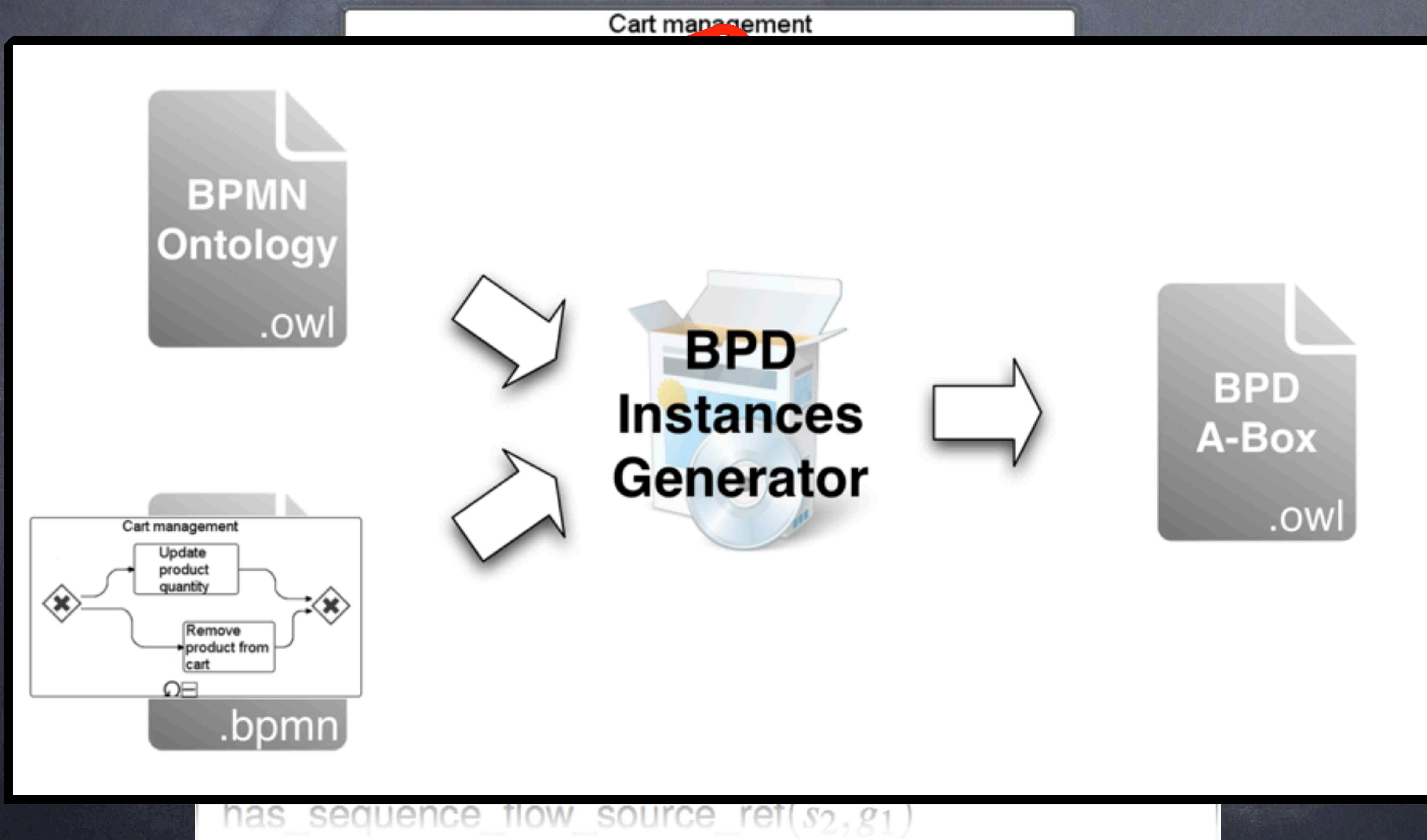


Structural assertions

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)	

Using The BPMN ontology

3. “Compose” the diagram structure in the A-box



Using The BPMN ontology

Reasoning services over an instantiated BPMN ontology

- By encoding a BPMN diagram as a set of instances of the BPMN Ontology, several **reasoning services** can be **implemented**:
  - **Query answering on BPMN diagrams**: it is possible to provide process querying mechanisms (via SPARQL) that exploit via reasoning the information formalized in the BPMN Ontology: e.g. "Which are the activities which follows gateways and produce a data object?" and "Are there sub-processes which do not contain start/end events?";
  - **Compliance checking of a BPMN diagram against the BPMN Specification**: to verify the compliance of a process with the structural conditions enforced by BPMN on diagrams. (via OWL reasoning, but in **closed-world assumption!**).

Thank you! Questions?

Marco Rospocher

http://dkm.fbk.eu/rospocher - rospocher@fbk.eu

DATA & KNOWLEDGE MANAGEMENT

FONDAZIONE BRUNO KESSLER