Exercise: Representing Process Knowledge in an Ontology

In this exercise an ontology about process knowledge is created in Protégé using the method Ontology Developmetn 101 (Noy & McGuinness, 2001).

The exercise was done during the lecture.

Determine Domain and Scope

The objective of the knowledge base is to support the process manager in designing the process flow and assigning tasks to actors. Therefore, knowledge about process flows and responsibilities for tasks shall be represented in the ontology.

Competency Questions

To determine the scope of the ontology, the following competency questions were defined:

- Who executes task X?
- Which task is executed after task X?
- When can task X start?

Reusing existing ontologies

Thee are no ontologies, which were imported. However, one can use concepts from modeling languages as basis to identify relevant terms. In case of business processes, the concepts of the BPMN standard can be used as reference

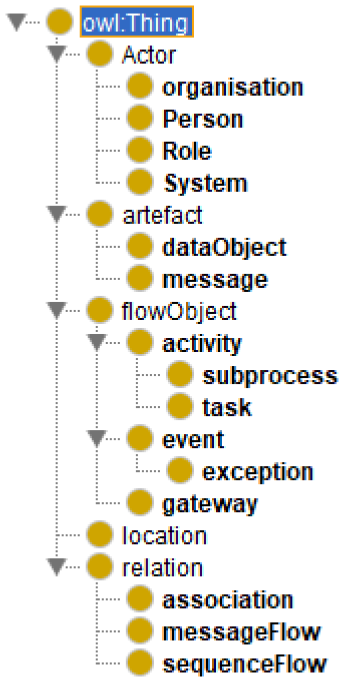
Enumerate important terms

BPMN was used to identify important terms. As the focus is on process flow and responsibilities, important terms are: process, task, role, event, sequence, gateway, department, application, and execute,

Define classes and class hierarchy

The identified terms ask, role, event, sequence, gateway, department, application are represented as classes. A bottom-up approach was chosen to structure them in a hierarchy.

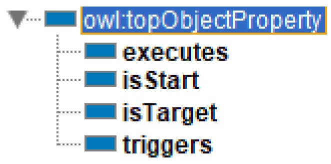
A subset of the hierarchy is shown in the following figure:



- owl:Thing
  - Actor
    - organisation
    - Person
    - Role
    - System
  - artefact
    - dataObject
    - message
  - flowObject
    - activity
      - subprocess
      - task
    - event
      - exception
    - gateway
  - location
  - relation
    - association
    - messageFlow
    - sequenceFlow

Define data properties and object properties

The object properties represent relations between elements. The following object properties were represented:



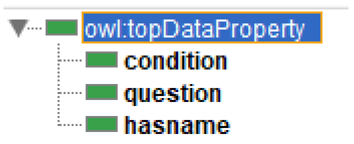
- owl:topObjectProperty
  - executes
  - isStart
  - isTarget
  - triggers

The relation executes connects an actor and a task.

Trigger is a relation between an event and a task. As soon as an event occurs, it triggers the execution of a task.

IsStart and is Target are connecting the sequenceFlow relation to the elements it is connected to: isStart is the flow Object in which the sequenceFlow relation start and the isTarget is the flowObject in which the sequenceFlow ends.

The data properties are shown in the following figure:



- owl:topDataProperty
  - condition
  - question
  - hasname

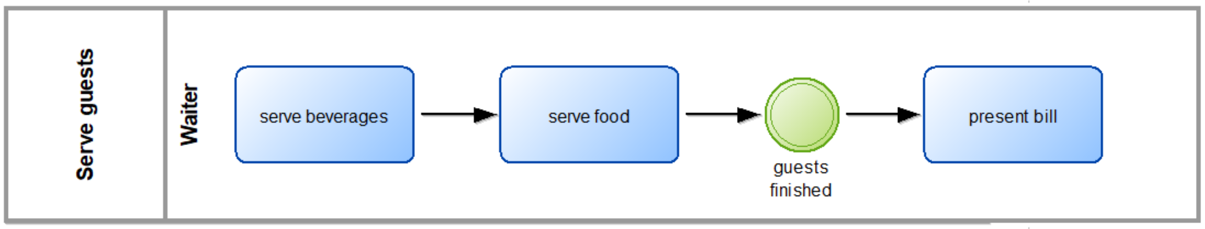
A performer and a task can have a name, a gateways has a question and a sequence flow can have a condition.

Create instances

Instances were created to represent the following process:

The waiter serves the beverages. Then the waiter serves the food. When the guests are finished, the waiter presents the bill.

This is the graphical model of the process:

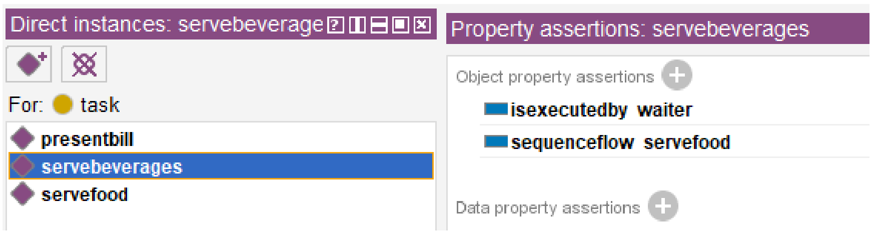


There are three tasks:
- servebeverages
- servefood
- presentbill

one performer:
- waiter

one event:
- guestsfinished

The following figure shows the properies of the task serververages:

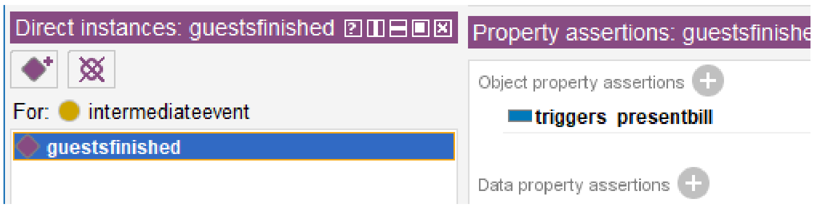

Direct instances: servebeverage
For: task
- presentbill
- servebeverages
- servefood

Property assertions: servebeverages
Object property assertions
- isexecutedby waiter
- sequenceflow servefood

Data property assertions

The task is executed by the waiter and has a sequenceflow to servefood, representing that servefood is executed after servebeverages.

The following figure shows that the event guestsfinished triggers the task presentbill:


Direct instances: guestsfinished
For: intermediateevent
- guestsfinished

Property assertions: guestsfinishe
Object property assertions
- triggers presentbill

Data property assertions

Querying the knowledge base

The following queries show that the with the knowledge base the competency questions can be answered

Query 1: Who performs task “Serve food”

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX po: <http://www.semanticweb.org/knut.hinkelmann/ProcessOntology#>
SELECT ?performer
      WHERE { ?performer po:executes  po:servefood }
```

Answer: ?performer = waiter

Query 2: When can task "Present Bill" start?

The query is equivalent to: What is the trigger for task "Present Bill"?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX po: <http://www.semanticweb.org/knut.hinkelmann/ProcessOntology#>
SELECT ?event
       WHERE { ?event po:triggers  po:presentbill}
```

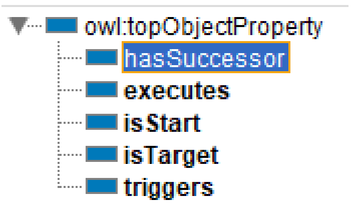
Answer: ?event = guestsfinished

Rules

With rules we can derive implicit property values.

The sequence flow of tasks is represented with a relation that has a start and target. With a rule we want to make a direct relation between the start and target element of the sequence flow.

We create a new object property: hasSuccessor



```
owl:topObjectProperty
    hasSuccessor
    executes
    isStart
    isTarget
    triggers
```

Domain and range of this object property are flowObjects.

The following rule looks for the start and target of a sequenceFlow and make adds the target element as value for the property hasSuccesor of the element at the start of the sequenceFlow

```
po:isStart(?sequenceflow, ?start) ^ po:isTarget(?sequenceflow, ?target) ->
po:hasSuccessor(?start, ?target)
```

Query 3: Which task es executed after task "serve beverages"?

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX po: <http://www.semanticweb.org/knut.hinkelmann/ProcessOntology#>
SELECT ?task
       WHERE { po:servebeverages po:hasSuccessor  ?task }
```

Answer: ?task = servefood