A Project Management Ontology in OWL

Roberto Casadei
May, 2014

Course: Semantic Web
Computer Science and Engineering LM
Alma Mater Studiorum – Università di Bologna

Summary

Introduction

WHAT → modelling the knowledge in a domain of interest using OWL

HOW → trying to follow a methodology for ontology development

With respect to the previous assignment

- More emphasis on the process (rather than on the product)
- Striving to follow a more systematic approach
- Deeper awareness about what an ontology is

Ontology development: the methodology

Iterative process

1) Determine **domain**, **scope** and **objectives**

2) Informal/semi-formal **knowledge acquisition**

3) Refine **requirements** and **tests**

4) **Design** and **implementation**

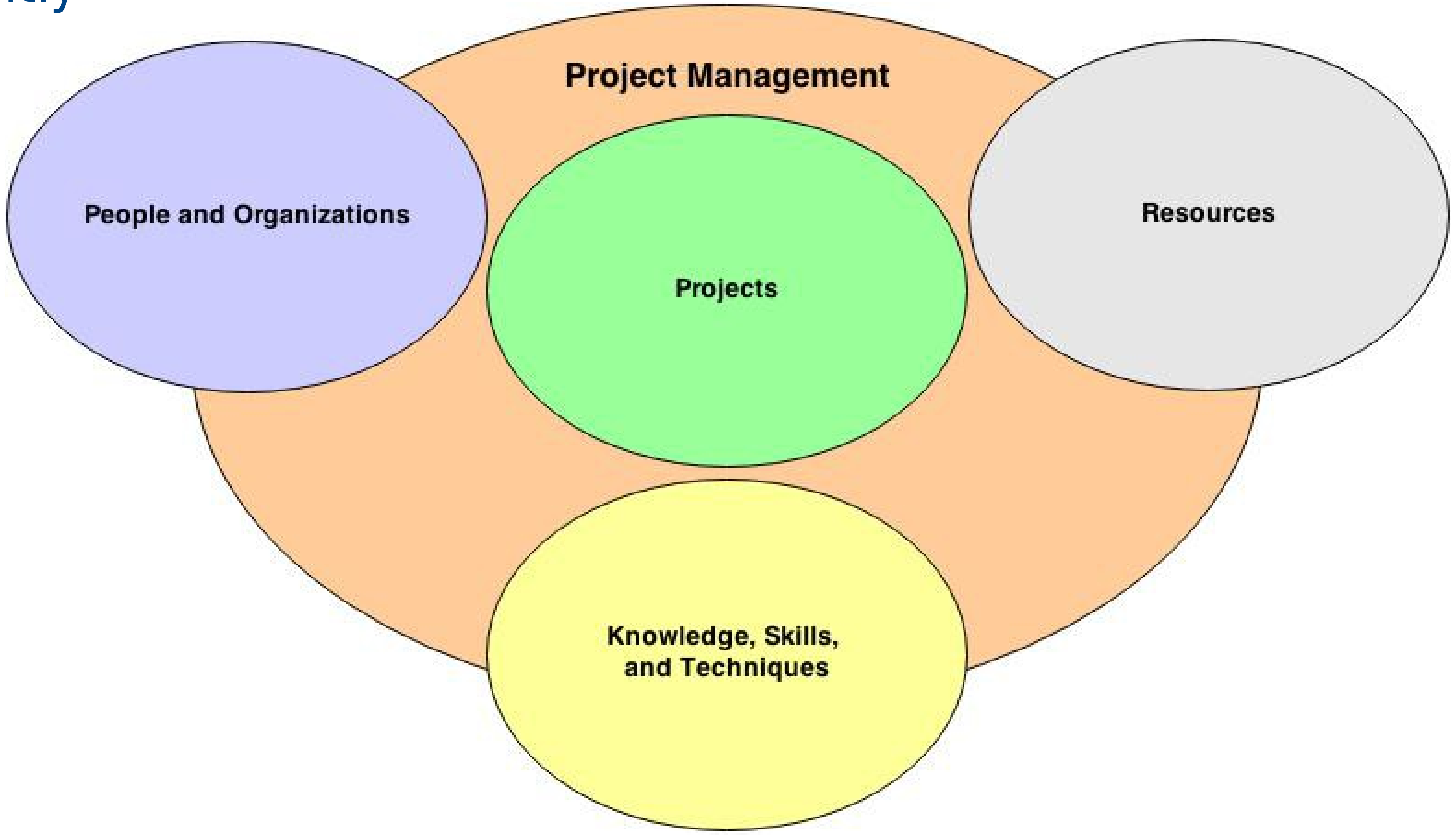
5) **Evaluation** and **quality assurance**

6) **Maintainance**: usage monitoring and evolution

1a) Domain and scope

Domain → **project management**

- I.e., the application of knowledge, skills and techniques to execute projects effectively and efficiently



1b) Domain and scope

Are we sure we need an ontology?

- Actually, this phase might conclude that an ontology is not the right tool for us

High-level objectives

- To provide a **description tool** and support **unambiguous communication**
- To provide a model upon which **project management software tools** can be built
    - Typical features: collaborative tools, issue tracking, scheduling, project portfolio management, resource management, document management tools, workflow management, reporting&analysis

Two abstraction levels

- Project management concepts and models
- Project models

1c) Domain and scope

Scope

1) Project management models

2) People and organizations

3) Resources

4) Projects

But re-using when possible!

2a) Informal/semi-formal knowledge acquisition

Sources

- The previous assignment
- The ''Project Management'' course **;)**
- Wikipedia, the PMBOK (Project Management Body of Knowledge) & literature

Steps

- **Collect the terms** and **informally organize** them
- **Paraphrase**/clarify terms to produce **informal concept definitions**
- **Inspect** the terms

2b) Informal/semi-formal knowledge acquisition

Collect the terms and organize them informally

Project management model
Project management concept

Resource
Time
Budget
Tool
Human-resource
Deliverable
Artifact
Documentation

Project management effort
Project
Software project

Process
Activity

Organization
Agent
Person

Participant
Stakeholder
Group
Team
Role
Customer
Membership
Project manager
Team member

2c) Informal/semi-formal knowledge acquisition

Paraphrase the terms to produce informal concept definitions

For example:

- **Organization** = group of individuals organized to work for some purpose or mission.
- **Process** = set of interrelated actions and activities performed to create a pre-specified product, service, or result. Each process is characterized by its inputs, the tools and techniques that can be applied, and the resulting outputs.
- **Deliverable** = any unique and verifiable product, result or capability to perform a service that is required to be produced to complete a process, phase, or project
- … … …

2d) Informal/semi-formal knowledge acquisition

What do we want to say about these terms? Which properties they have?

For example:

- A resource can be the **input** or the **output** of a project/process/activity
- A process can be **organized** into **subprocesses** and activities
- An agent or person can **play** a certain role in a project **during** a time interval
- … … …

Do these concepts have some **common characteristics**? Are there **additional things** with those characteristics which might be considered?

For example:

- Projects as well as processes and activities have goals, inputs, outputs, and participants
- Organizations as well as projects, processes, and activities can define roles to be played by agents or people
- … … …

3a) Refine requirements and tests: functional requirements

We should define better what we want to do so to be able to move towards design

For example, via use cases

- A project management expert **defines** a new project management model by **extending** a basic model with both concepts already defined somewhere and new concepts
- A project manager **uses** a software tool which allows him to graphically **create** a project description that is based on the MPMO ontology concepts
- A software tool may be used to **check** if all the concepts instantiated into a project description are defined within the associated project management model
- … … …

3b) Refine requirements and tests: non-functional requirements

What properties should the ontology exhibit?

- **Generality** & **Flexibility**: adaptivity to contexts/situations which can be very diverse
- **Simplicity**: providing a model which is easy to understand and put into action
- **Extensibility**: providing support for specializations and tailoring

→ These suggest goal-driven tests by the perspective of ontology usage

Is OWL the right language to build the ontology? Which OWL flavor?

→ We compare what we want to express in our ontology to OWL expressivity and decidability boundaries

- For example: a design decision is that of having things being both an instance and a class
  - E.g.: Organization is both an instance of PMConcept and a subclass of org:Organization
- This would put the ontology into OWL Full (which is undecidable) – but OWL 2 solves this problem using the so-called punning where a URI can be used in multiple roles

4a) Design & implementation

Inputs for this phase

- Goals, requirements
- Informal knowledge previously collected

Design / modelling / implementation

- Delve into concepts, relationships, properties
- Organize the concepts
- Formalize the definitions

Conventions

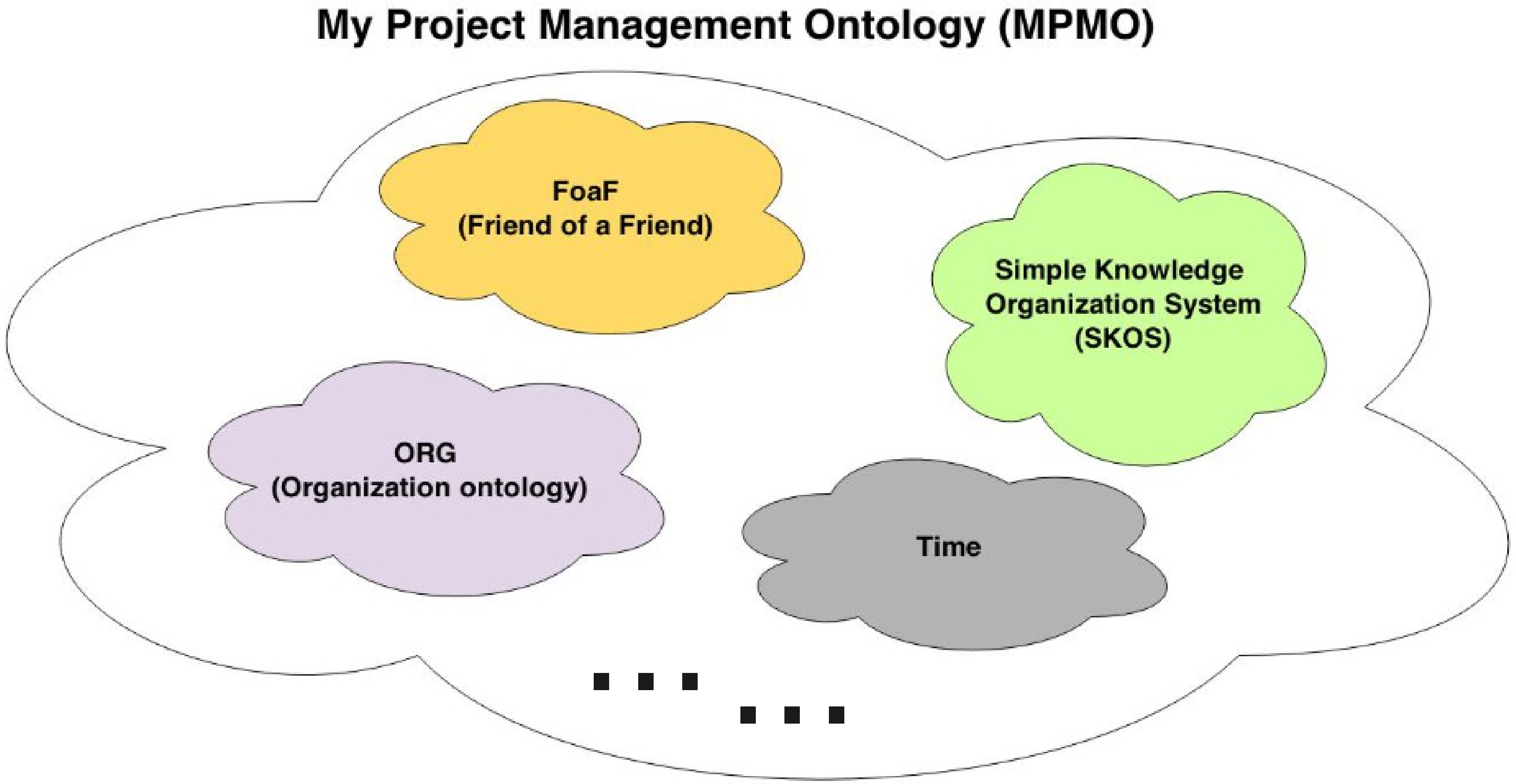
- Class names are singular

4b) Design & implementation

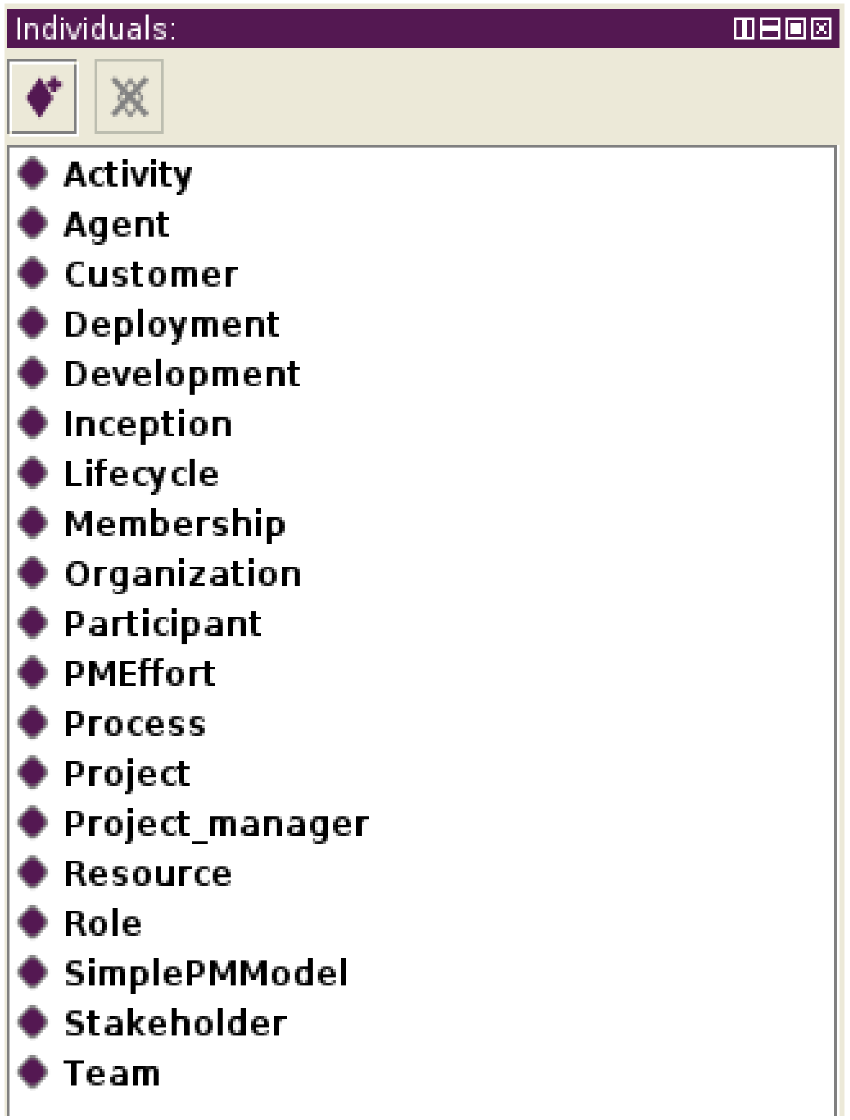
Ontology editor: **Protégé**

Imported ontologies

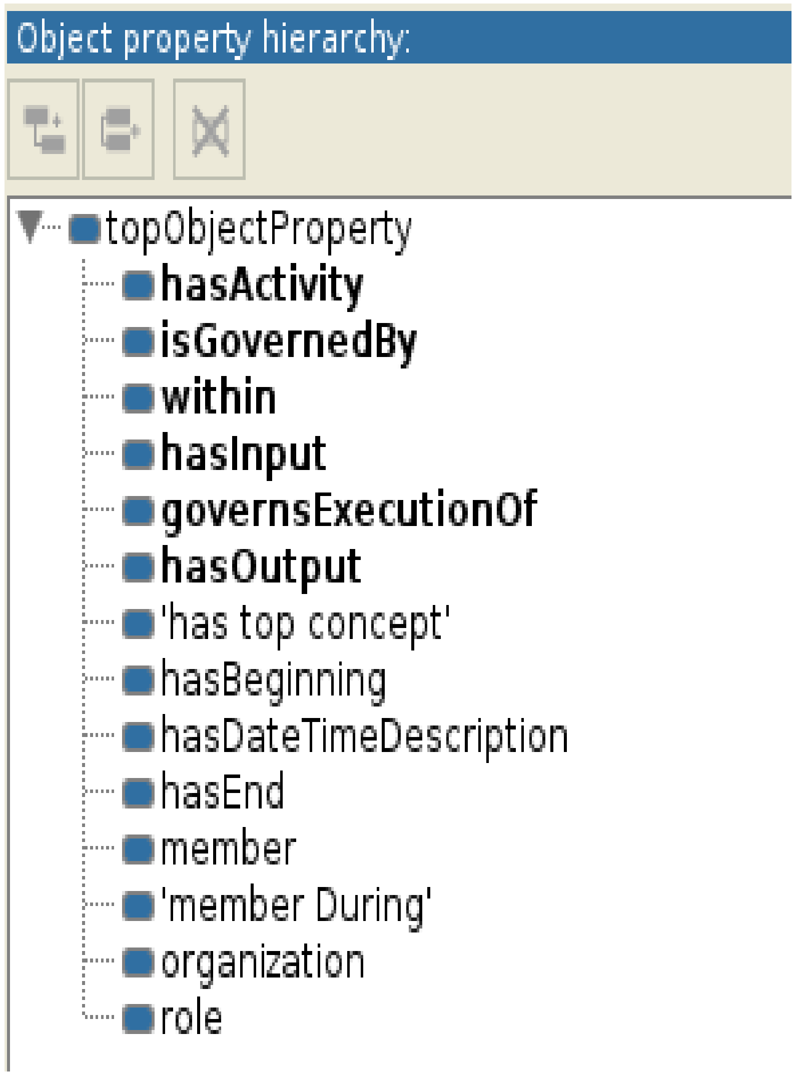
- SKOS
- ORG
- FoaF
- Time
- … … …



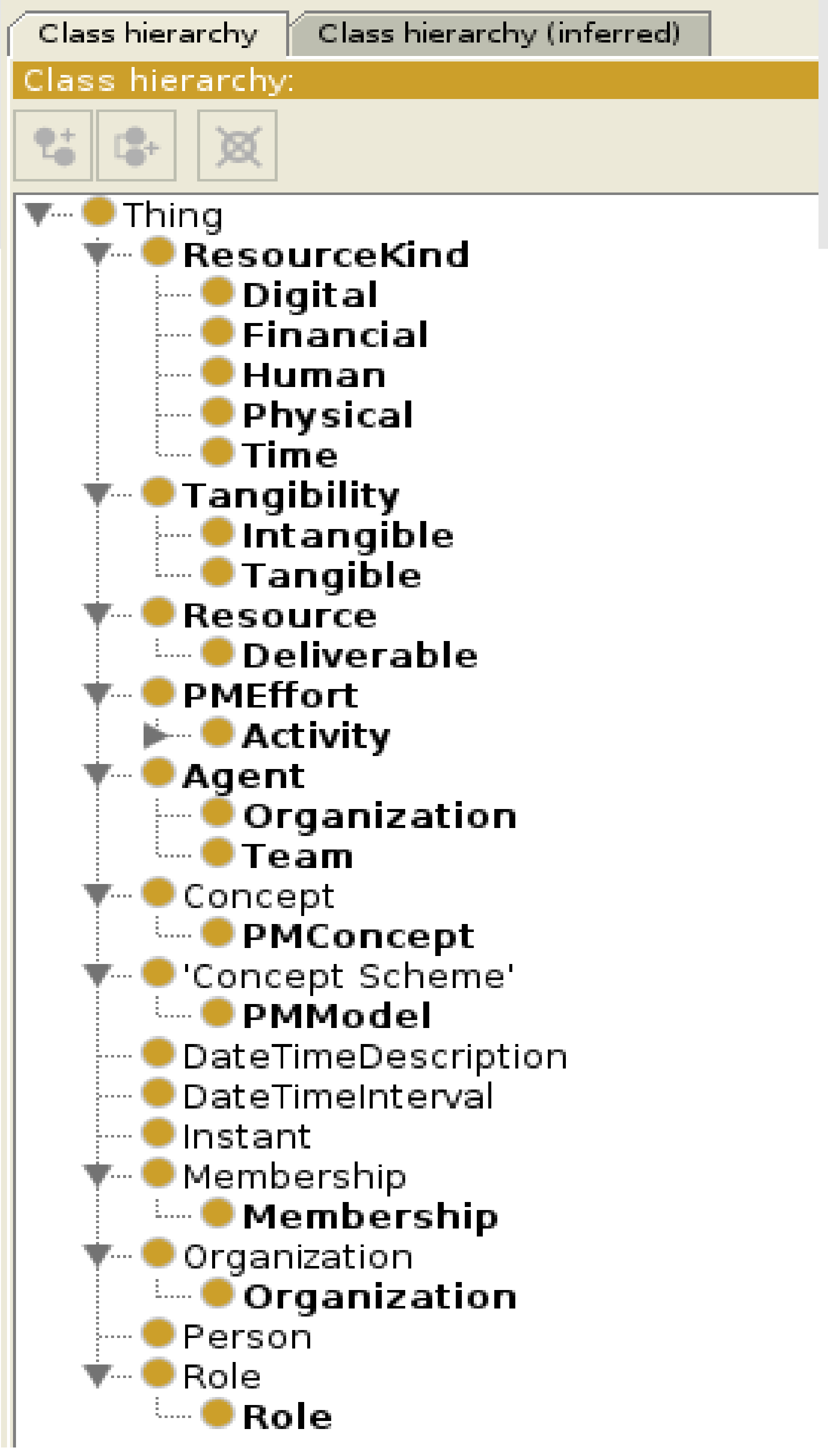
4c) Design & implementation


Individuals:

- Activity
- Agent
- Customer
- Deployment
- Development
- Inception
- Lifecycle
- Membership
- Organization
- Participant
- PMEffort
- Process
- Project
- Project_manager
- Resource
- Role
- SimplePMModel
- Stakeholder
- Team


Object property hierarchy:

- topObjectProperty
  - hasActivity
  - isGovernedBy
  - within
  - hasInput
  - governsExecutionOf
  - hasOutput
  - 'has top concept'
  - hasBeginning
  - hasDateTimeDescription
  - hasEnd
  - member
  - 'member During'
  - organization
  - role



Class hierarchy:

- Thing
  - ResourceKind
    - Digital
    - Financial
    - Human
    - Physical
    - Time
  - Tangibility
    - Intangible
    - Tangible
  - Resource
    - Deliverable
  - PMEffort
    - Activity
  - Agent
    - Organization
    - Team
  - Concept
    - PMConcept
  - 'Concept Scheme'
    - PMModel
  - DateTimeDescription
  - DateTimeInterval
  - Instant
  - Membership
    - Membership
  - Organization
    - Organization
  - Person
  - Role
    - Role

4d) Design & implementation

Some notes about design/modelling decisions

- At one abstraction level we have PMModel, PMConcept, PMEffort, and at another abstraction level we have the project management concepts (instances of PMConcept!)
  - A PMModel **governsExecutionOf** PMEffort (inverse functional property)
- Project is a subclass of Process, Process is a subclass of Activity
  - A Process must produce in output one or more Deliverables
    - This cannot be expressed in RDFS. How is this expressed in OWL?
    - Unqualified cardinality restrictions in OWL 1: owl:minCardinality + owl:someValuesFrom
    - Qualified cardinality restrictions in OWL 2 : owl:minQualifiedCardinality + owl:onClass
  - A Project is a temporary endeavour → it must have a duration
    - time:hasBeginning min 1 time:Instant
    - time:hasEnd min 1 time:Instant
- Resource has a subclass Deliverable, whereas the kind of resource is expressed using a value partition ResourceKind which is totally covered by disjointed subclasses: Digital, Financial, Human, Physical, Time

5a) Evaluation and quality assurance

Based on goals and requirements!!!

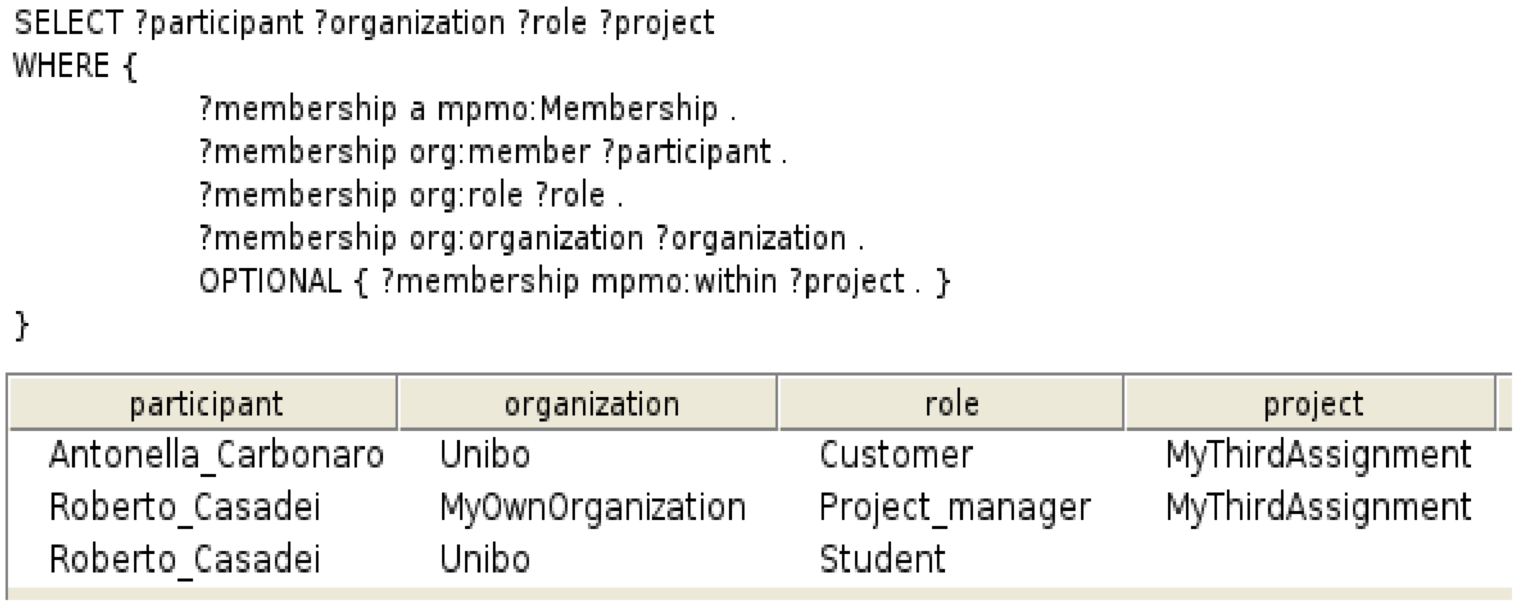
Checklist for non-functional ontology properties

- Generality & Flexibility
- Simplicity
- Extensibility

Evaluation through "**acceptance testing**"

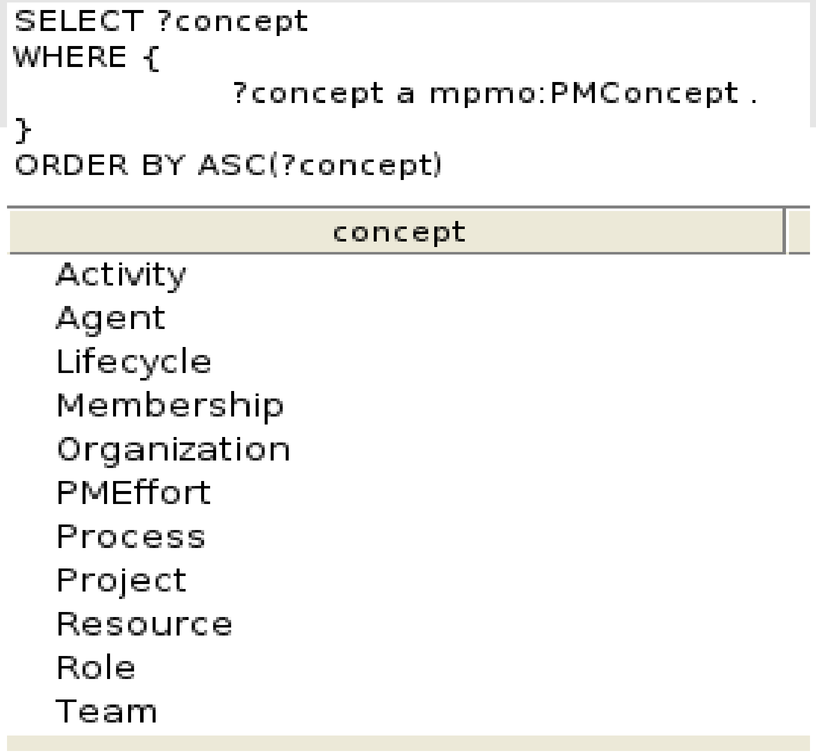
- No applications are built upon this ontology
- But we can perform some **query** upon data

5b) Evaluation and quality assurance



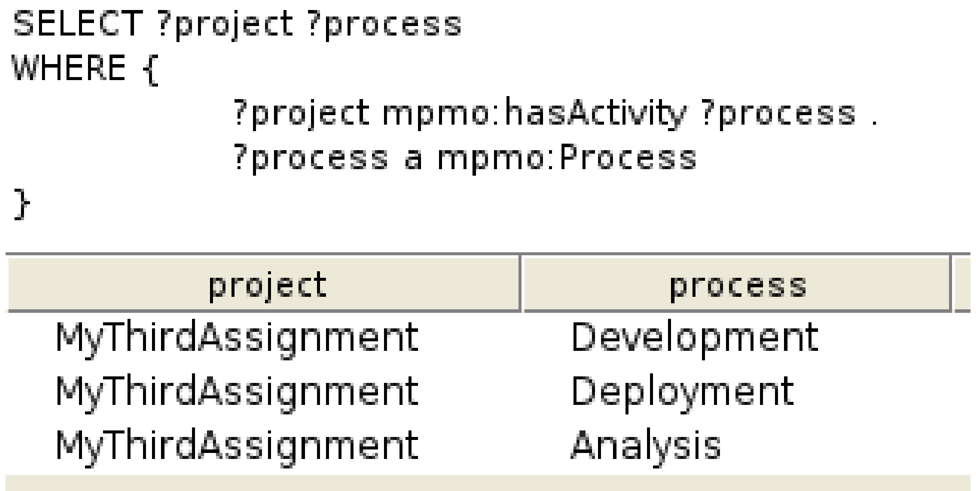
```
SELECT ?participant ?organization ?role ?project
WHERE {
            ?membership a mpmo:Membership .
            ?membership org:member ?participant .
            ?membership org:role ?role .
            ?membership org:organization ?organization .
            OPTIONAL { ?membership mpmo:within ?project . }
}
```

participant	organization	role	project
Antonella_Carbonaro	Unibo	Customer	MyThirdAssignment
Roberto_Casadei	MyOwnOrganization	Project_manager	MyThirdAssignment
Roberto_Casadei	Unibo	Student	



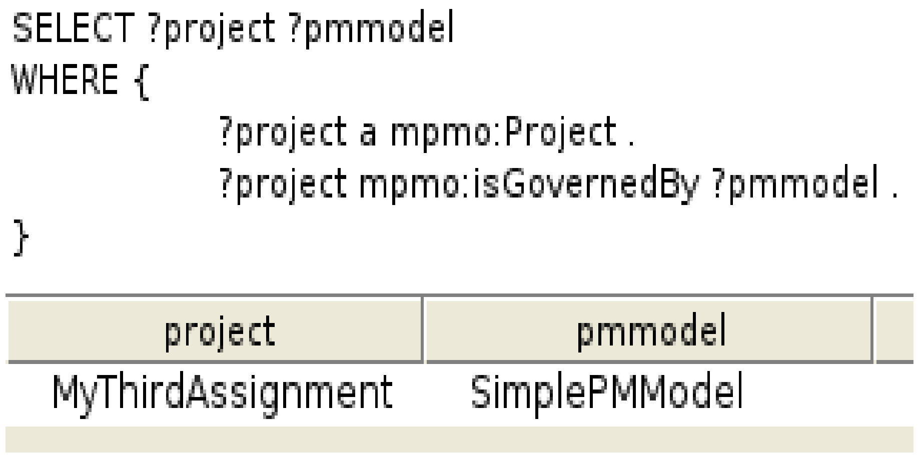
```
SELECT ?concept
WHERE {
            ?concept a mpmo:PMConcept .
}
ORDER BY ASC(?concept)
```

concept
Activity
Agent
Lifecycle
Membership
Organization
PMEffort
Process
Project
Resource
Role
Team



```
SELECT ?project ?process
WHERE {
            ?project mpmo:hasActivity ?process .
            ?process a mpmo:Process
}
```

project	process
MyThirdAssignment	Development
MyThirdAssignment	Deployment
MyThirdAssignment	Analysis



```
SELECT ?project ?pmmodel
WHERE {
            ?project a mpmo:Project .
            ?project mpmo:isGovernedBy ?pmmodel .
}
```

project	pmmodel
MyThirdAssignment	SimplePMModel

6a) Maintainance: usage monitoring and evolution

Is there **version compatibility** with the RDFS model released in the previous assignment?

- Do all constructs have the same meaning? NO.
- E.g., Stakeholder was a subclass of Person while now it is an instance of Role (and Role is disjoint with Agent, which is a superclass of Person)

Conclusion

Two main considerations

1) As for other engineering efforts, ontology development benefits from the adoption of a methodology

- In particular, be iterative and get feedback by use!

2) OWL allows for the definition of stronger ontologies with respect to those built upon RDF Schema

- The increased expressivity of the ontology language allowed us specify additional constraints and formalize the definitions