

# A Review on Ontology Design Methodology

**Tommaso Agnoloni**  
**Lorenzo Bacci**




**extracts from:**

## Computational Ontologies

Aldo Gangemi  
Valentina Presutti  
Semantic Technology Lab (ISTC-CNR), Roma  
{aldo.gangemi, valentina.presutti}@istc.cnr.it


### Credits:

Eva Blomqvist, Sean Bechhofer, Kingsley Idehen, Fabien Gandon,  
Harry Halpin, Jim Hendler, Tim Berners-Lee

**NeOn Methodology for Building Ontology  
Networks**

Asunción Gómez-Pérez  
Mari Carmen Suárez-Figueroa  
{agomez, mcsuarez}@fi.upm.es  
Ontology Engineering Group, Departamento de Inteligencia Artificial  
Facultad de Informática  
Universidad Politécnica de Madrid



September 29<sup>th</sup>, 2008  
EKAW 2008. Sicily, Italy

# What is Ontology Design?

- **Ontologies are artifacts**
  - Have a structure (linguistic, “taxonomical”, logical)
  - Their function is to “encode” a description of the world (actual, possible, counterfactual, impossible, desired, etc.) for some purpose
- **Ontologies must match both domain and task**
  - Allow the description of the entities (“domain”) whose attributes and relations are concerned by some purpose, e.g. *drugs as commodities that contain preparations of selected compounds having an expected application within medical treatments*
  - Serve a purpose (“task”), e.g. *finding piperocaine-based anesthetic drugs, integrating a drug database with a compound database, matching available resources to devised drug production plans, etc.*
- **Ontologies have a lifecycle**
  - Are created, evaluated, fixed, and exploited just like any artifact
  - Their lifecycle has some original characteristics regarding:
    - *Data*
    - *Project and workflow types*
    - *Argumentation structures*
    - *Design patterns*

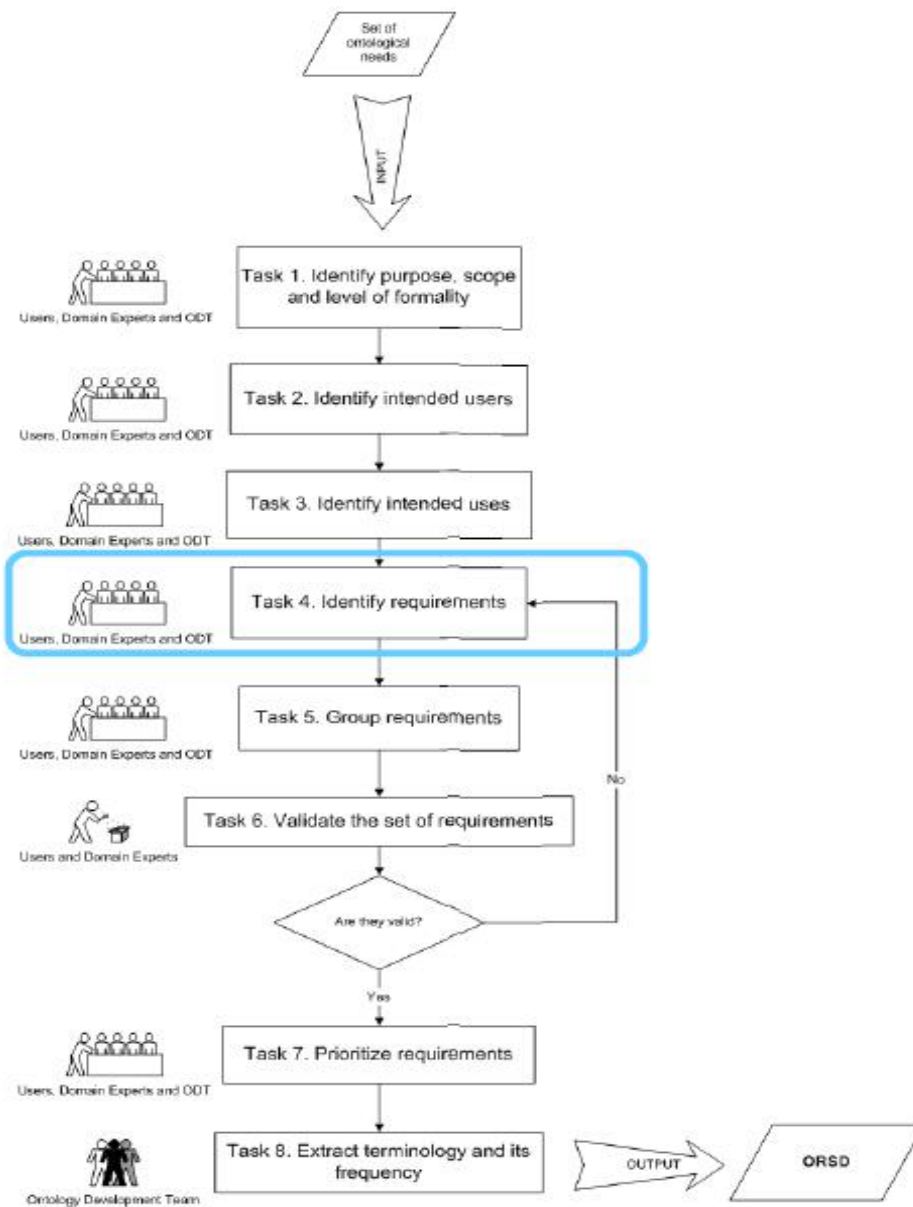
# Ontology Specification

- **Ontology (Requirements) Specification** is a collection of requirements that the ontology should fulfill, e.g. reasons to build the ontology, target group, intended uses, possibly reached through a consensus process.
- **Requirements** are those needs that the ontology to be built should represent/cover.
- **Competency Questions (CQs)** are questions that the ontology to be built should be able to answer.
  - CQs are a way to represent requirements.
  - CQs can be written in natural language (NL) and can be formalized in ontology query languages (e.g. SPARQL).

# Ontology Specification / example

- Story
- Decompose story in sentences
- Write instance-free sentences
- Transform into competency questions
- Sentence: *Charlie Parker is the alto sax player on Lover Man, Dial, 1946*
  - *Charlie Parker (person)*
  - *the alto sax player (player role)*
  - *on Lover Man (tune)*
  - *Dial (publisher)*
  - *1946 (recording year)*
- CQs
  - *what persons do play a musical instrument?*
  - *on what tune?*
  - *for what publisher?*
  - *in what recording year?*

# NeOn Methodology



*Input:* a set of ontological needs

- **Objective:** identifying the **ontology requirements**

- **Techniques:** writing the requirements in Natural Language in the form of **competency questions (CQs)**

- **Tools:** mind map tools, excel, and collaborative tools

- **Output:** a list of competency questions written in Natural Language and a set of answers for the CQs

	A	B
1	N	Competency Questions
2	CQ1	What is the Job Seeker Name?
3	CQ2	What is the Job Seeker nationality?
4	CQ3	When is the Job Seeker birthdate?
5	CQ4	What is the Job Seeker contact information?
6	CQ5	What is the Job Seeker current job?
7	CQ6	What is the Job Seeker desired job?
8	CQ7	What are the Job Seeker desired working conditions?
9	CQ8	What kind of contract does the Job Seeker want?
10	CQ9	How much salary does the Job Seeker want to earn?
11	CQ10	What is the Job Seeker education level?
12	CQ11	What is the Job Seeker work experience?
13	CQ12	What is the Job Seeker knowledge?
14	CQ13	What is the Job Seeker expertise?
15	CQ14	What are the Job Seeker skills?
16	CQ15	What publications does the Job Seeker have?

# Resources for the Semantic Web

- **Metadata**

- Resources are marked-up with descriptions of their content. No good unless everyone speaks the same language

- **Terminologies**

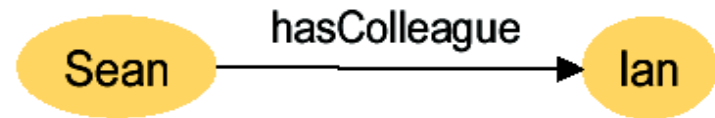
- provide shared and common vocabularies of a domain, so search engines, agents, authors and users can communicate. No good unless everyone means the same thing

- **Ontologies**

- provide a shared and common understanding of a domain that can be communicated across people and applications, and will play a major role in supporting information exchange and discovery

# RDF Data Model

- Provides a simple data model based on triples
- Statements are <subject, predicate, object> triples:
  - <Sean,hasColleague,Ian>
- Can be represented as a graph:
- Statements describe properties of resources
- A resource is any object that can be pointed to by a URI:
  - The generic set of all names/addresses that are short strings that refer to resources
  - a document, a picture, a paragraph on the Web, <http://www.cs.man.ac.uk/index.html>, a book in the library, a real person (?), isbn://0141184280
- Properties themselves are also resources (URIs)



# What does RDF give us?

- A mechanism for annotating data and resources.
- Single (simple) data model.
- Syntactic consistency between names (URIs).
- Low level integration of data.



# Ontology Design Patterns (ODP)

OWL gives us logical language constructs, but does not give us any guidelines on how to use them in order to solve our tasks. *E.g. modeling something as a class or an object property is mostly arbitrary*

... OWL is not enough for building a good ontology and we cannot ask all web users either to learn logic, or to study ontology design

# Ontology Design Patterns (ODP)

Reusable solutions are described as “Ontology Design Patterns”, which help reducing arbitrariness without asking for sophisticated skills ...

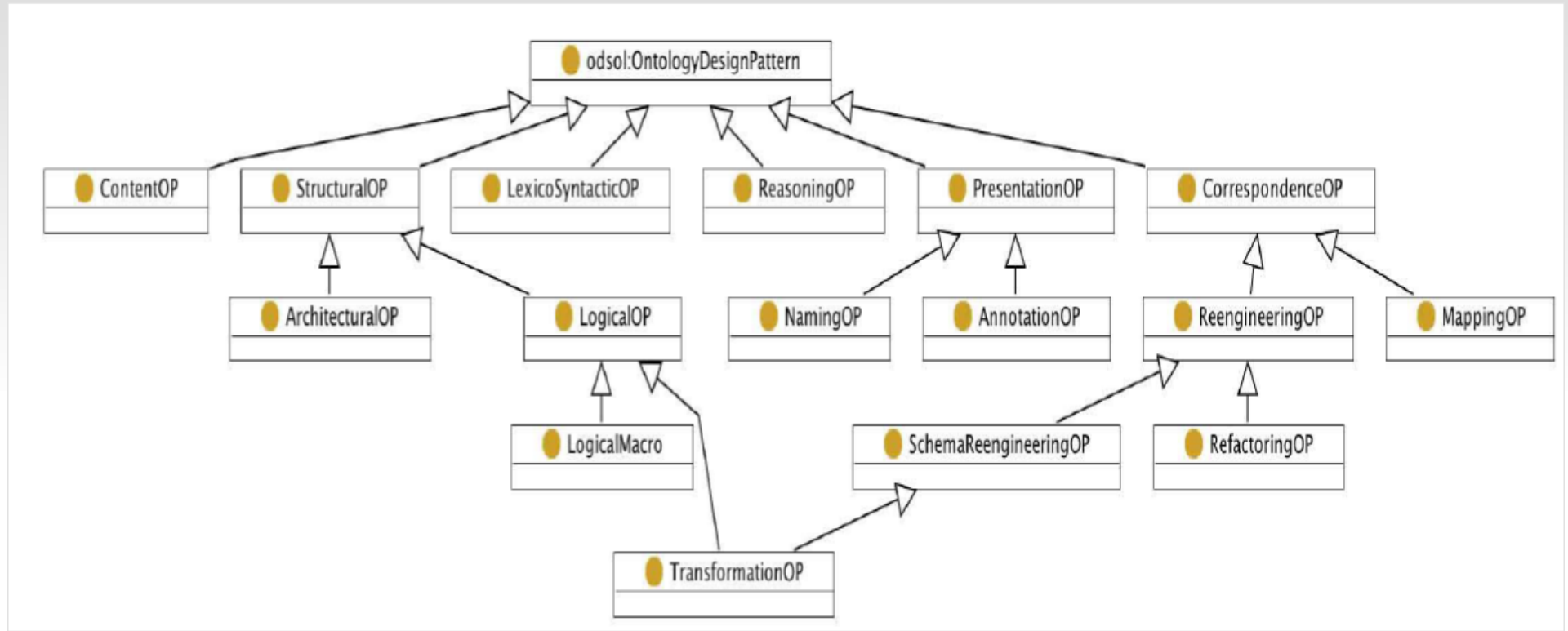
DEFINITION:

An Ontology Design Pattern is a reusable modeling solution to solve a recurrent ontology design problem

## OPs and patterns in other disciplines

- Our concept of “pattern” is associable with the wider “good/best practice” of software engineering.
- It includes a wider range of solution types. For example:
  - naming conventions in software engineering are considered good practices, they are not design patterns.
  - In ontology engineering “naming” is an important design activity (it can have a strong impact on the usage of the ontology e.g., for selection, mapping, etc.).
- We distinguish the different types of OPs by grouping them into six families.
- Each family addresses different kinds of problems, and can be represented with different levels of formality.

# Types of Ontology Design Patterns (OPs)



- We also distinguish between ontological resources that are not OPs and Ontology Design Anti-Patterns (AntiOPs)

# Common misconceptions

- Disjointness of primitives
- Interpreting domain and range
- Closed and Open Worlds

# Disjointness

- By default, primitive classes are not disjoint.
- Unless we explicitly say so, the description (Animal and Vegetable) is not inconsistent.
- Similarly, with individuals -- the so-called Unique Name Assumption (often present in DL languages) does not hold, and individuals are not considered to be distinct unless explicitly asserted to be so.

## Domain and Range

- OWL allows us to specify the domain and range of properties.
- Note that this is not interpreted as a constraint as you might expect.
- Rather, the domain and range assertions allow us to make inferences about individuals.
- Consider the following:
  - `ObjectProperty(employs domain(Company) range(Person))`  
`Individual(IBM value(employs Jim))`
- If we haven't said anything else about IBM or Jim, this is not an error. However, we can now infer that IBM is a Company and Jim is a Person.

# Close and Open World assumptions

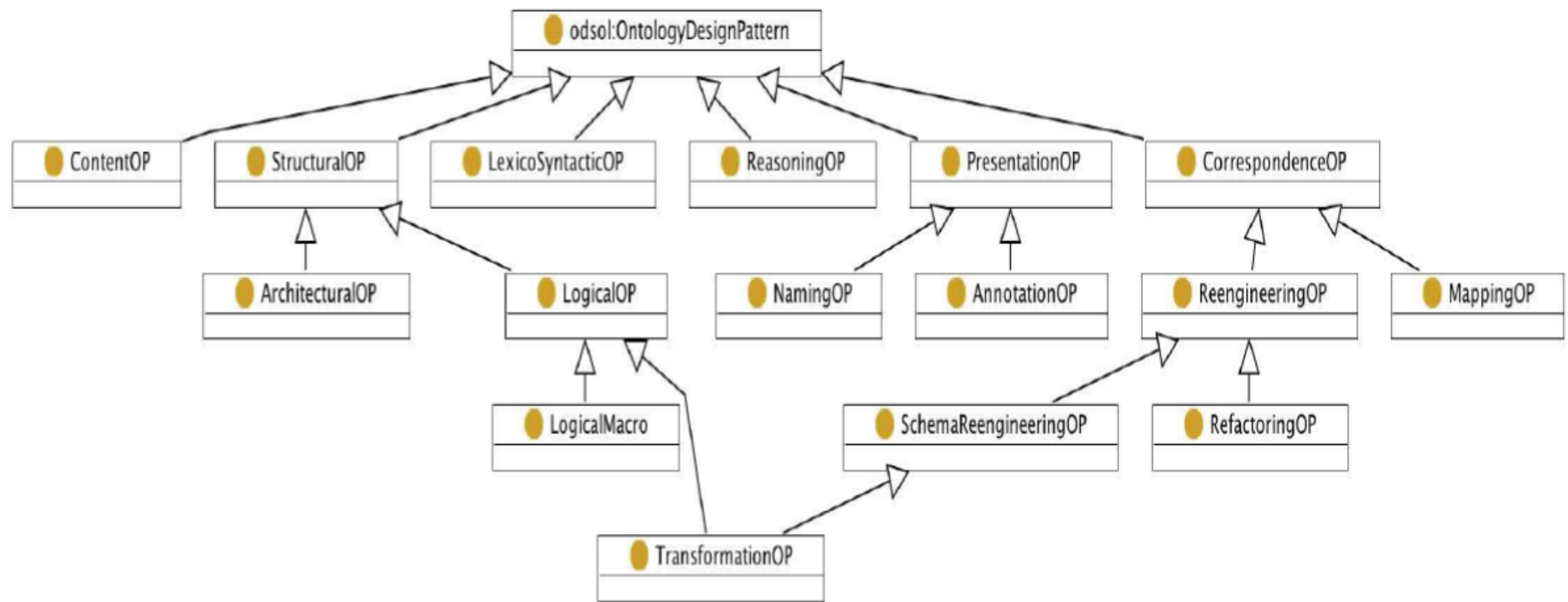
- The standard semantics of OWL makes an Open World Assumption (OWA).
  - We cannot assume that all information is known about all the individuals in a domain.
  - Negation as contradiction
    - Anything might be true unless it can be proven false
- Closed World Assumption (CWA)
  - Named individuals are the only individuals in the domain
  - Negation as failure.
    - If we can't deduce that  $x$  is an  $A$ , then we know it must be a  $(\neg A)$ .



## Example 1

- cat/dog/lion

# Types of Ontology Design Patterns (OPs)



# Presentation OPs

## *Definition*

- *Presentation OPs deal with usability and readability of ontologies from a user perspective.*
- *They are meant as good practices that support the reuse of patterns by facilitating their evaluation and selection.*
- *Two types:*
  - *Naming Ops*
  - *Annotation OPs*

# Naming OPs

## *Definition*

- *Naming OPs are conventions on how to create names for namespaces, files, and ontology elements in general (classes, properties, etc.).*
- *Naming OPs are good practices that boost ontology readability and understanding by humans, by supporting homogeneity in naming procedures.*

# Examples of Naming OPs 1/2

- Namespace declared for ontologies.
- It is recommended to use the base URI of the organization that publishes the ontology
  - e.g. <http://www.w3.org> for the W3C, <http://www.fao.org> for the FAO, <http://www.ittig.cnr.it> ....
- followed by a reference directory for the ontologies
  - e.g. <http://www.ittig.cnr.it/ontologies/>
- It is also important to choose an approach for encoding versioning, either on the name, or on the reference directory

# Examples of Naming OPs 2/2

- Class names
- They should not contain plurals, unless explicitly required by the context
  - Names like Areas is considered bad practice, if e.g. an instance of the class Areas is a single area, not a collection of areas
- It is also recommended to use readable names instead of e.g. alphanumerical codes
  - Non-readable name can be used (even if not recommended) if associated to proper annotations (see Annotation OPs)
- It is useful to include the name of the parent class as a suffix of the class name
  - e.g. MarineArea rdfs:subClassOf Area
- Class names conventionally start with a capital letter
  - e.g. Area instead of area

# Annotation OPs

- Annotation OPs provide annotation properties or annotation property schemas that are meant to improve the understandability of ontologies and their elements
- Annotation properties:
  - <http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl>
  - annotation of OWL implementation of CPs

# Examples of Annotation OPs

- RDF Schema labels and comments (crucial for manual selection and evaluation)
- Each class and property should be annotated with meaningful labels
  - i.e., by means of the annotation property `rdfs:label`, with also translations in different languages.
- Each ontology and ontology element should be annotated with the rationale they are based on
  - i.e., by means of the annotation property `rdfs:comment`



# Content OPs (CPs)

- CPs encode conceptual, rather than logical design patterns.
  - Logical OPs solve design problems independently of a particular conceptualization
  - CPs are patterns for solving design problems for the domain classes and properties that populate an ontology, therefore they address content problems
  - they are content-dependent
- Modeling problems solved by CPs have two components: domain and requirements.
  - A same domain can have many requirements (e.g. different scenarios in a clinical information context)
  - A same requirement can be found in different domains (e.g. different domains with a same “expert finding” scenario)
  - CPs are strictly related to small use cases i.e., each of them is built out of a domain task
  - A typical way of capturing requirements is by means of *competency questions*
  - A *competency question* is a typical query that an expert might want to submit to a knowledge base of its target domain, for a certain task.

## Formal characteristics of OWL CPs

- (Small) ontology morphing
- Downward subsumption of at least one element
- Mostly graphs of classes and properties that are self-connected through axioms (subClassOf, equivalentClass, domain, range, disjointFrom)
- Usually there is an underlying n-ary relation (sometimes polymorphic)

# Characteristics of CPs

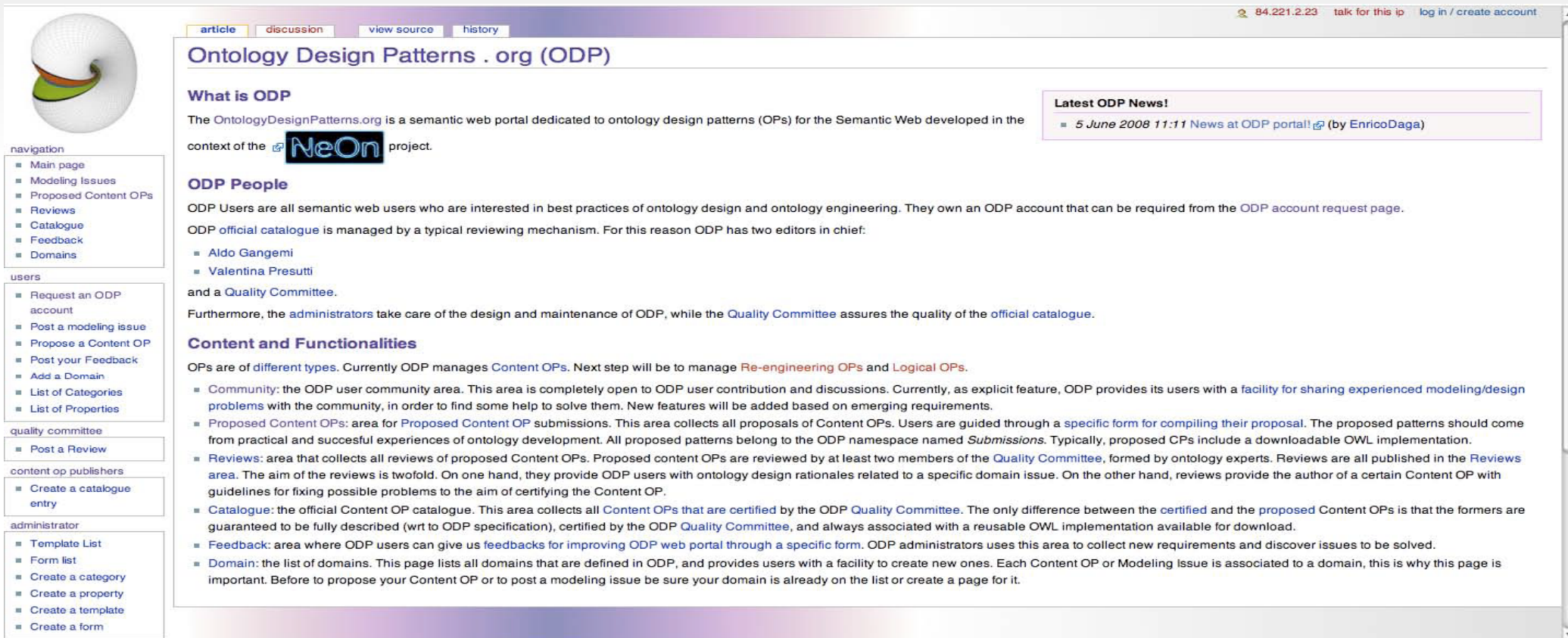
- Requirement-covering components
  - They are defined in terms of the requirements (or cqs) they satisfy
- Small, autonomous components.
  - A CP is a small, autonomous ontology and ensures a certain set of inferences to be enabled on its corresponding knowledge base.
  - Smallness and autonomy of CPs facilitate ontology designers: composing CPs enables them to govern the complexity of the whole ontology.
  - CPs require a critical size, so that their diagrammatical visualizations are aesthetically acceptable and easily memorizable.
  - CP visualization must be intuitive and compact, and should catch relevant, “core” notions of a domain.

# Characteristics of CPs

- Reasoning-relevant components
  - They allow some form of inference (*minimal axiomatization*, e.g. not an isolated class)
- Linguistically relevant components.
  - Many CPs nicely match linguistic patterns called frames.
  - A frame can be described as a lexically founded ontology design pattern.
    - Frames typically encode argument structures for verbs, e.g. the frame Desiring associates elements (or “semantic roles”) such as Experiencer, Event, FocalParticipant, LocationOfEvent, etc.
- Best practice components.
  - A CP should be used to describe a “best practice” of modelling.
  - Best practices are intended as local, thus derived from experts.
  - The quality of CPs is currently based on the personal experience and taste of the proposers, or on the provenance of the knowledge resource where the pattern comes from.

# Catalogue

- A catalogue of Cps
  - <http://www.ontologydesignpatterns.org> (odp-web)
  - catalogue entry




The screenshot displays the homepage of the Ontology Design Patterns (ODP) website. At the top, there is a navigation bar with tabs for 'article', 'discussion', 'view source', and 'history'. The main header reads 'Ontology Design Patterns . org (ODP)'. Below this, a section titled 'What is ODP' explains that the portal is dedicated to ontology design patterns (OPs) for the Semantic Web, developed in the context of the NeOn project. To the right, a 'Latest ODP News!' box shows a news item from June 5, 2008, about news at the ODP portal by Enrico Daga. The left sidebar contains several menu sections: 'navigation' (Main page, Modeling Issues, Proposed Content OPs, Reviews, Catalogue, Feedback, Domains), 'users' (Request an ODP account, Post a modeling issue, Propose a Content OP, Post your Feedback, Add a Domain, List of Categories, List of Properties), 'quality committee' (Post a Review), 'content op publishers' (Create a catalogue entry), and 'administrator' (Template List, Form list, Create a category, Create a property, Create a template, Create a form). The main content area continues with 'ODP People', stating that ODP users are semantic web users interested in best practices, and lists Aldo Gangemi and Valentina Presutti as editors in chief. It also mentions a Quality Committee and administrators. The 'Content and Functionalities' section describes the types of OPs managed (Content, Re-engineering, Logical) and lists several functional areas: Community, Proposed Content OPs, Reviews, Catalogue, and Feedback, each with a brief description of their purpose and how they are managed.

84.221.2.23 talk for this ip log in / create account


article discussion view source history

## Ontology Design Patterns . org (ODP)

### What is ODP

The OntologyDesignPatterns.org is a semantic web portal dedicated to ontology design patterns (OPs) for the Semantic Web developed in the context of the  project.

**Latest ODP News!**

- 5 June 2008 11:11 News at ODP portal!  (by EnricoDaga)

### ODP People

ODP Users are all semantic web users who are interested in best practices of ontology design and ontology engineering. They own an ODP account that can be required from the [ODP account request page](#). ODP [official catalogue](#) is managed by a typical reviewing mechanism. For this reason ODP has two editors in chief:

- Aldo Gangemi
- Valentina Presutti

and a [Quality Committee](#).

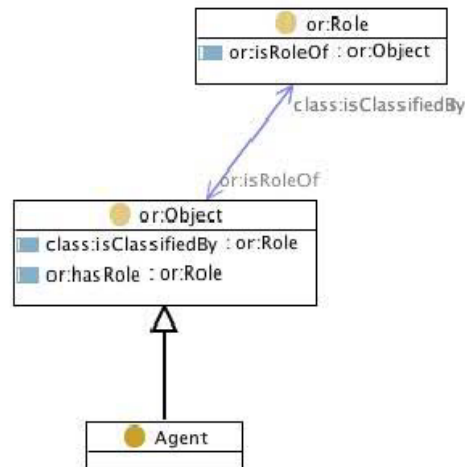
Furthermore, the [administrators](#) take care of the design and maintenance of ODP, while the [Quality Committee](#) assures the quality of the [official catalogue](#).

### Content and Functionalities

OPs are of different types. Currently ODP manages [Content OPs](#). Next step will be to manage [Re-engineering OPs](#) and [Logical OPs](#).

- **Community**: the ODP user community area. This area is completely open to ODP user contribution and discussions. Currently, as explicit feature, ODP provides its users with a [facility for sharing experienced modeling/design problems](#) with the community, in order to find some help to solve them. New features will be added based on emerging requirements.
- **Proposed Content OPs**: area for [Proposed Content OP](#) submissions. This area collects all proposals of Content OPs. Users are guided through a [specific form for compiling their proposal](#). The proposed patterns should come from practical and successful experiences of ontology development. All proposed patterns belong to the ODP namespace named *Submissions*. Typically, proposed CPs include a downloadable OWL implementation.
- **Reviews**: area that collects all reviews of proposed Content OPs. Proposed content OPs are reviewed by at least two members of the [Quality Committee](#), formed by ontology experts. Reviews are all published in the [Reviews area](#). The aim of the reviews is twofold. On one hand, they provide ODP users with ontology design rationales related to a specific domain issue. On the other hand, reviews provide the author of a certain Content OP with guidelines for fixing possible problems to the aim of certifying the Content OP.
- **Catalogue**: the official Content OP catalogue. This area collects all [Content OPs that are certified](#) by the ODP [Quality Committee](#). The only difference between the [certified](#) and the [proposed](#) Content OPs is that the formers are guaranteed to be fully described (wrt to ODP specification), certified by the ODP [Quality Committee](#), and always associated with a reusable OWL implementation available for download.
- **Feedback**: area where ODP users can give us [feedbacks for improving ODP web portal through a specific form](#). ODP administrators uses this area to collect new requirements and discover issues to be solved.
- **Domain**: the list of domains. This page lists all domains that are defined in ODP, and provides users with a facility to create new ones. Each Content OP or Modeling Issue is associated to a domain, this is why this page is important. Before to propose your Content OP or to post a modeling issue be sure your domain is already on the list or create a page for it.

# An example of CP: Agent Role



## Elements

The **AgentRole** Content OP locally defines the following ontology elements:

### Agent (owl:Class)

Any agentive **Object**, either physical, or social.

 [Agent page](#)

## Reviews about AgentRole

There are no reviews.

Go back to the [List of Content OP proposals](#)

The **time indexed person role** CP allows to represent temporariness of roles played by persons. It can be generalized for including objects or, alternatively the **n-ary classification** CP can be specialized in order to obtain the same expressivity.

The elements of this Content OP are added with the elements of its components and/or the elements of the Content OPs it is a specialization of.

## AgentRole

**Submitted by** [Valentina Presutti](#)

**Name** agent role

**Also Known As**

**Intent** To represent agents and the roles they play.

**Domains** [Management](#), [Organization](#), [Scheduling](#)

**Competency Questions** which agent does play this role?, what is the role that played by that agent?

**Reusable OWL** <http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl>

**Building Block**

**Consequences** This CP allows designers to make assertions on roles played by agents without involving the agents that play that roles, and vice versa. It does not allow to express temporariness of roles.

**Scenarios** She greeted us all in her various roles of mother, friend, and daughter.

**Known Uses**

**Web**

**References**

**Other**

**References**

**Examples (OWL files)** <http://www.ontologydesignpatterns.org/cp/examples/agentrole/ex1.owl>

**Extracted From** <http://www.loa-cnr.it/ontologies/DUL.owl>

**Reengineered**

**From**

**Has**

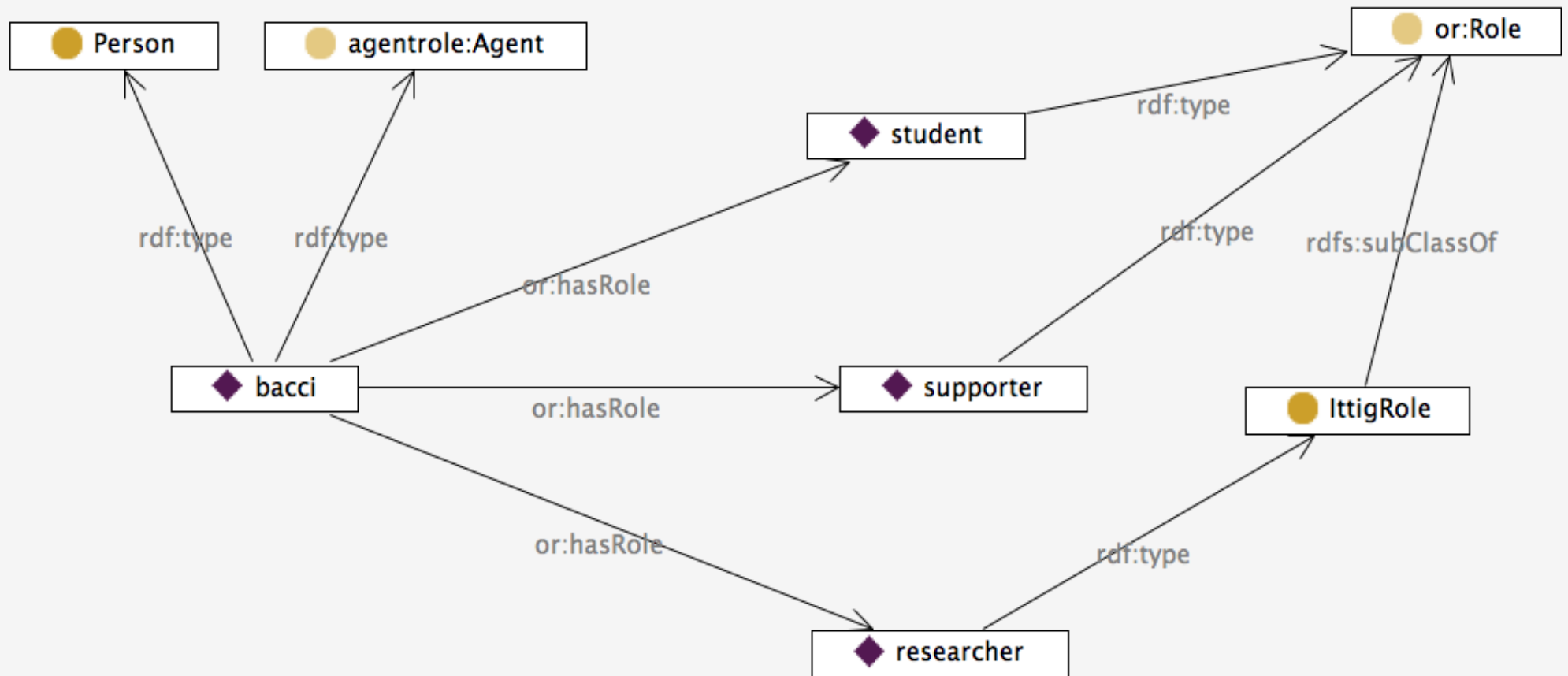
**Components**

**Specialization Of** [Submissions:Objectrole](#)

**Of**

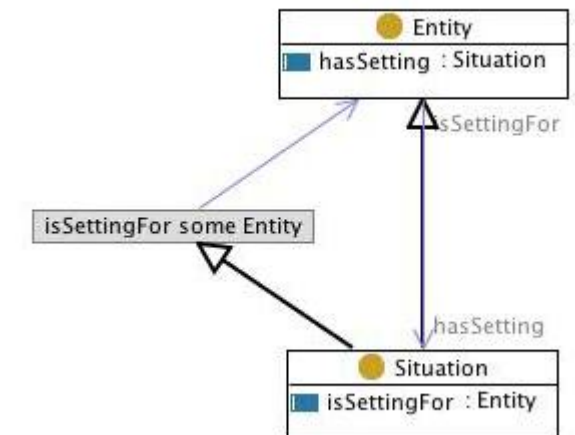
**Related CPs**

## An example of CP: Agent Role Instantiation





# An example of CP: Situation



## Elements

The **Situation** Content OP locally defines the following ontology elements:

### Entity (owl:Class)

Anything: real, possible, or imaginary, which some modeller wants to talk about for some purpose.

[Entity page](#)

### Situation (owl:Class)

A combination of circumstances involving a set of entities. It can be seen as a relational context, reifying a relation among the entities involved. In fact, it provides an explicit vocabulary to the [n-ary relation](#) [Logical OP](#)

[Situation page](#)

### has setting (owl:ObjectProperty)

a relation between entities and situations, e.g. this morning I've prepared my coffee with a new fantastic Arabica (i.e.: (an amount of) a new fantastic Arabica *hasSetting* the preparation of my coffee this morning). *is setting for* is its inverse.

[hasSetting page](#)

### is setting for (owl:ObjectProperty)

Inverse property of *has setting*

[isSettingFor page](#)

## Situation

Submitted by [Valentina Presutti](#)

Name Situation

Also Known As

Intent To represent facts, circumstances, observed contexts.

Domains General

Competency Questions What entities are in the setting of a certain situation?

Reusable OWL Building Block <http://www.ontologydesignpatterns.org/cp/owl/situation.owl>

Consequences This CP allows the designer to model both a certain situation, and the entities that are involved. It provides designers with a vocabulary for representing n-ary relations.

Scenarios I prepared a coffee with my heater, 300 ml of water, and an Arabica coffee mix.

Known Uses

Web References

Other References

Examples (OWL files) <http://www.ontologydesignpatterns.org/cp/examples/situation/coffee.owl>

Extracted From <http://www.loa-cnr.it/ontologies/DUL.owl>

Reengineered From

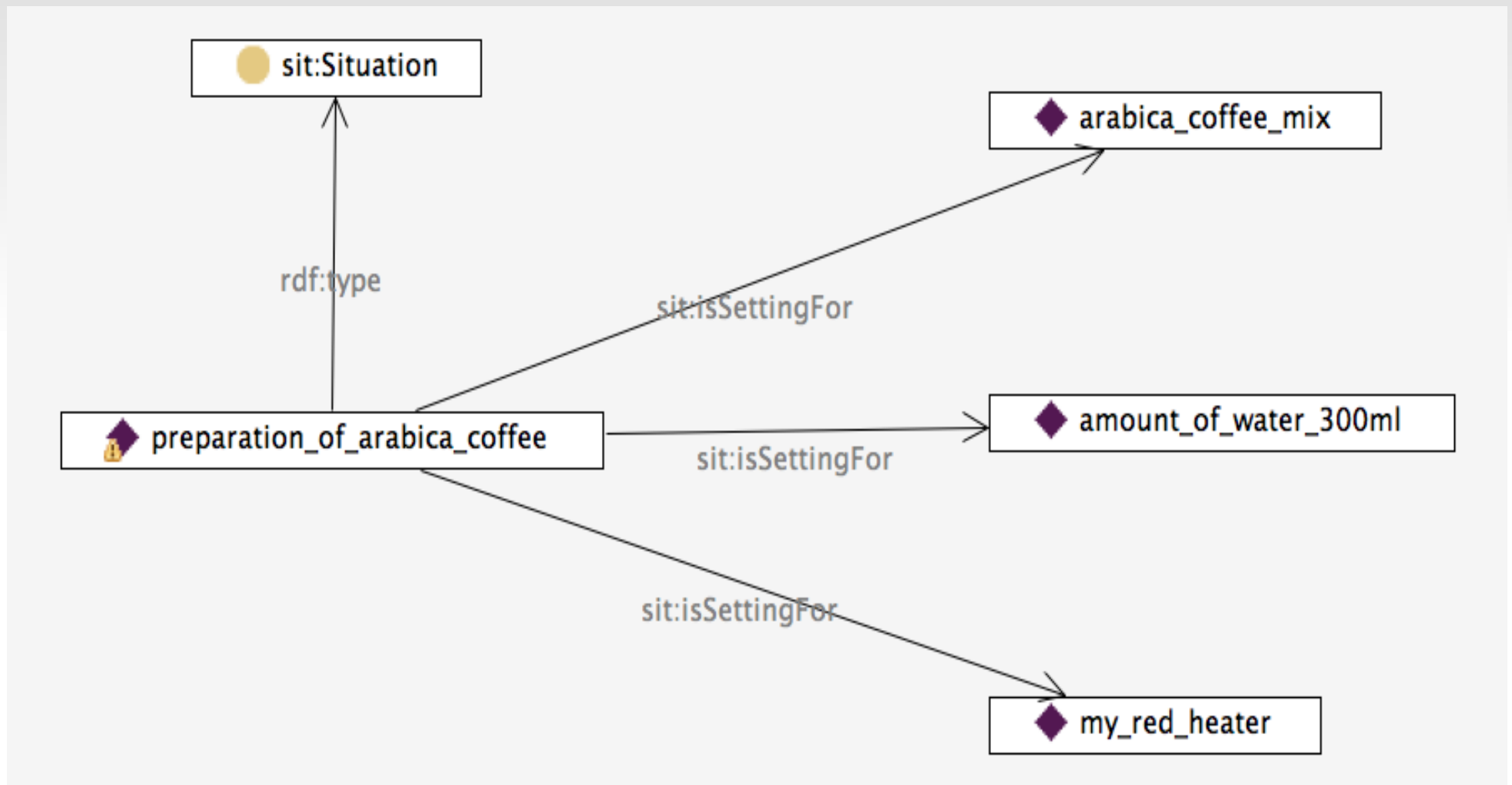
Has Components

Specialization Of

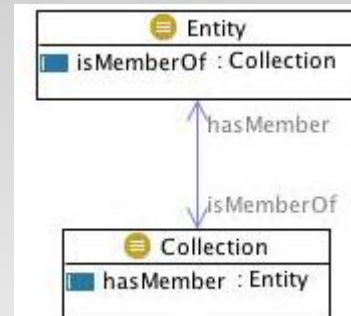
Related CPs [Submissions:Description](#)



## An example of CP: Situation Instantiation



# An example of CP: CollectionEntity



## Elements

The **CollectionEntity** Content OP locally defines the following ontology elements:

### **Collection** (owl:Class)

Any container for entities that share one or more common properties. E.g. *stone objects*, *the nurses*, *the Louvre Aegyptian collection*. A collection is not a logical class: a collection is a first-order entity, while a class is a second-order one.

 [Collection page](#)

### **Entity** (owl:Class)

Anything: real, possible, or imaginary, which some modeller wants to talk about for some purpose.

 [Entity page](#)

### **hasMember** (owl:ObjectProperty)

A relation between collections and entities, e.g. 'my collection of saxophones includes an old Adolphe Sax original alto' (i.e. my collection has member an Adolphe Sax alto). The object property **isMemberOf** is its inverse.




 [hasMember page](#)

### **isMemberOf** (owl:ObjectProperty)

The inverse of **hasMember**.

 [isMemberOf page](#)

## CollectionEntity

Submitted by	ValentinaPresutti
Name	collection entity
Also Known As	collections, membership
Intent	To represent collections, and their entities, i.e. to represent membership.
Domains	<a href="#">Parts and Collections</a>
Competency Questions	Which collection this entity is member of?, Which are the members of this collection?
Reusable OWL Building Block	<a href="http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl">http://www.ontologydesignpatterns.org/cp/owl/collectionentity.owl</a> 
Consequences	It is possible to put sets in the domain of discourse through the class <a href="#">Collection</a> , which reifies them.
Scenarios	The Louvre Aegyptian collection.
Known Uses	
Web References	
Other References	
Examples (OWL files)	<a href="http://www.ontologydesignpatterns.org/cp/examples/collectionentity/ISTC.owl">http://www.ontologydesignpatterns.org/cp/examples/collectionentity/ISTC.owl</a> 
Extracted From	<a href="http://www.loa-cnr.it/ontologies/DUL.owl">http://www.loa-cnr.it/ontologies/DUL.owl</a> 
Reengineered From	
Has Components	
Specialization Of	
Related CPs	

# An example of CP: Time Interval

TimeInterval
hasIntervalDate : date
hasIntervalEndDate : date[0..1]
hasIntervalStartDate : date[0..1]

## Elements

The **TimeInterval** Content OP locally defines the following ontology elements:

### Time Interval (owl:Class)

Any region in a dimensional space that represents time.

[TimeInterval page](#)

### has interval date (owl:DatatypeProperty)

A datatype property that encodes values from xsd:date for a time interval; a same time interval can have more than one xsd:date value: begin date, end date, date at which the interval holds, as well as dates expressed in different formats: xsd:gYear, xsd:dateTime, etc.

[hasIntervalDate page](#)

### has interval start date (owl:DatatypeProperty)

The start date of a **time interval**.

[hasIntervalStartDate page](#)

### has interval end date (owl:DatatypeProperty)

The end date of a **time interval**.

[hasIntervalEndDate page](#)

## TimeInterval

Submitted by [Valentina Presutti](#)

Name time interval

### Also Known As

Intent To represent time intervals.

Domains [Time](#)

Competency Questions What is the end time of this interval?, What is the starting time of this interval?, What is the date of this time interval?

Reusable OWL <http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl>

### Building Block

Consequences The dates of the time interval are not part of the domain of discourse, they are datatype values. If there is the need of reasoning about dates this Content OP should be used in composition with the **region** Content OP.

Scenarios The time interval "January 2008" starts at 2008-01-01 and ends at and ends at 2008-01-31.

### Known Uses

### Web

### References

### Other

### References

Examples (OWL <http://www.ontologydesignpatterns.org/cp/examples/timeinterval/january2008.owl> files)

### Extracted From

### Reengineered

### From

### Has

### Components

### Specialization

### Of

### Related CPs

## Example: ittig project

- In TopBraid

(Re)use situations:  
matching CPs covering against local problems

# Covering

- The *covering* property expresses the fact that a CP satisfies a set CQ of *competency questions* ( $cq_1, \dots, cq_n$ ).

$$cov(CP, CQ)$$

- A  $cq_i$  can be transformed to a query  $q_i$  to be submitted to a knowledge base.
- A CP covers CQ if it is as expressive as it is needed to store the necessary knowledge for answering  $q_1, \dots, q_n$ .

# Representing local problems

- Local problems can be expressed in different ways:
  - use cases, scenarios, user requirements, local competency questions (cqs), etc.
- All can be transformed to local “cqs”.
  - Red Hot Chili Peppers recorded the Stadium Arcadium album during 2005
  - When did Red Hot Chili Peppers record the Stadium Arcadium album?
  - Which albums did Red Hot Chili Peppers record during 2005?
  - ....
- Local “cqs” are not usually at the same level of generality as the cqs of Cps
  - e.g., they may contain reference to instance element e.g. Stadium Arcadium
  - we need to abstract them
  - When did a certain band record a certain album?
  - Which albums did a certain band record during a certain time period?
  - ...

## What we mean by *matching cqs to CPs*

- What do we mean by matching a cq to Cps?
  - To compare the local cqs to the cqs covered by a CP in order to evaluate the CP suitability for solving the local problems.
  - Ongoing work on automatic support for CP selection starting from local cqs
    - parsing of requirements and extraction of cqs
    - formalization of cqs
    - NLP support to match cqs terminology to CP lexicalizations
  - ontology matching



## Summary of reuse situations and examples

- Precise or redundant matching
- Broader or narrower matching
- Partial matching

# Import

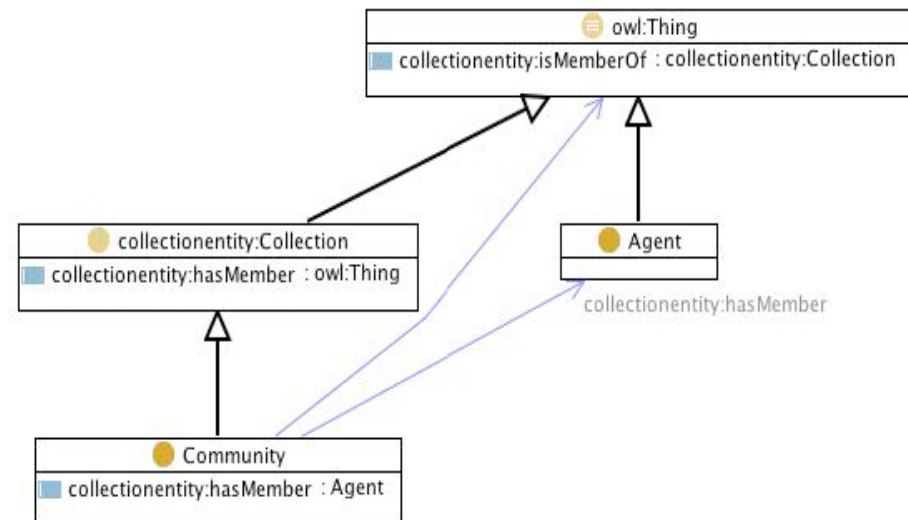
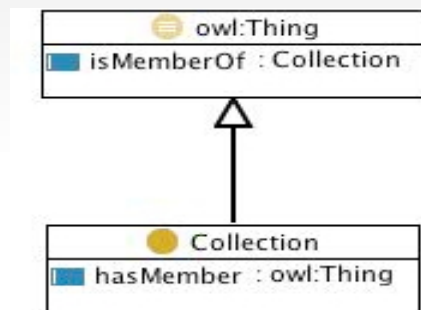
- Import is the basic mechanism for ontology reuse.
- It is also the only one directly supported in the OWL vocabulary
  - i.e., *owl:import*.
- Import is applicable to ontologies, hence also to Cps.
- If an ontology O2 imports an ontology O1, all the ontology elements and OWL axioms from O1 are included in O2 .
- The imported ontology elements and axioms cannot be modified
  - i.e., the ontology elements and axioms are read-only entities for O2.
- By importing a CP, an ontology ensures the set of inferences allowed by the CP in its corresponding knowledge base.

## Broader/narrower matching

- Broader matching:
  - The cqs covered by a CP are more general than the local ones.
  - The CP has firstly to be **imported**, then it has to be **specialized** in order to cover the local scenarios.
- Narrower matching:
  - The cqs covered by a CP are more specific than the local ones.
  - The CP has firstly to be **imported**, then it has to be **generalized** in order to cover the local scenarios.
- Two usage operations are identified:
  - specialization
  - generalization

# Specialization

- A content pattern CP2 specializes CP1 if at least one ontology element of CP2 is subsumed by an ontology element of CP1
  - i.e., either by *rdfs:subClassOf* or *rdfs:subPropertyOf*



## Broader matching example

- Consider the following scenario:
  - a person plays a certain role.
- it can be expressed by the competency question included in the following set:
  - CQ1={who did play a certain role?}
- From the previous example we know that
  - $\text{cov}(\text{agent role}, \text{Req})$
- where CQ is more general than CQ1
- *We can import agent role (prefix ar:) and define the class Person in the following way:*
  - *Person rdfs:subClassOf ar:Agent*

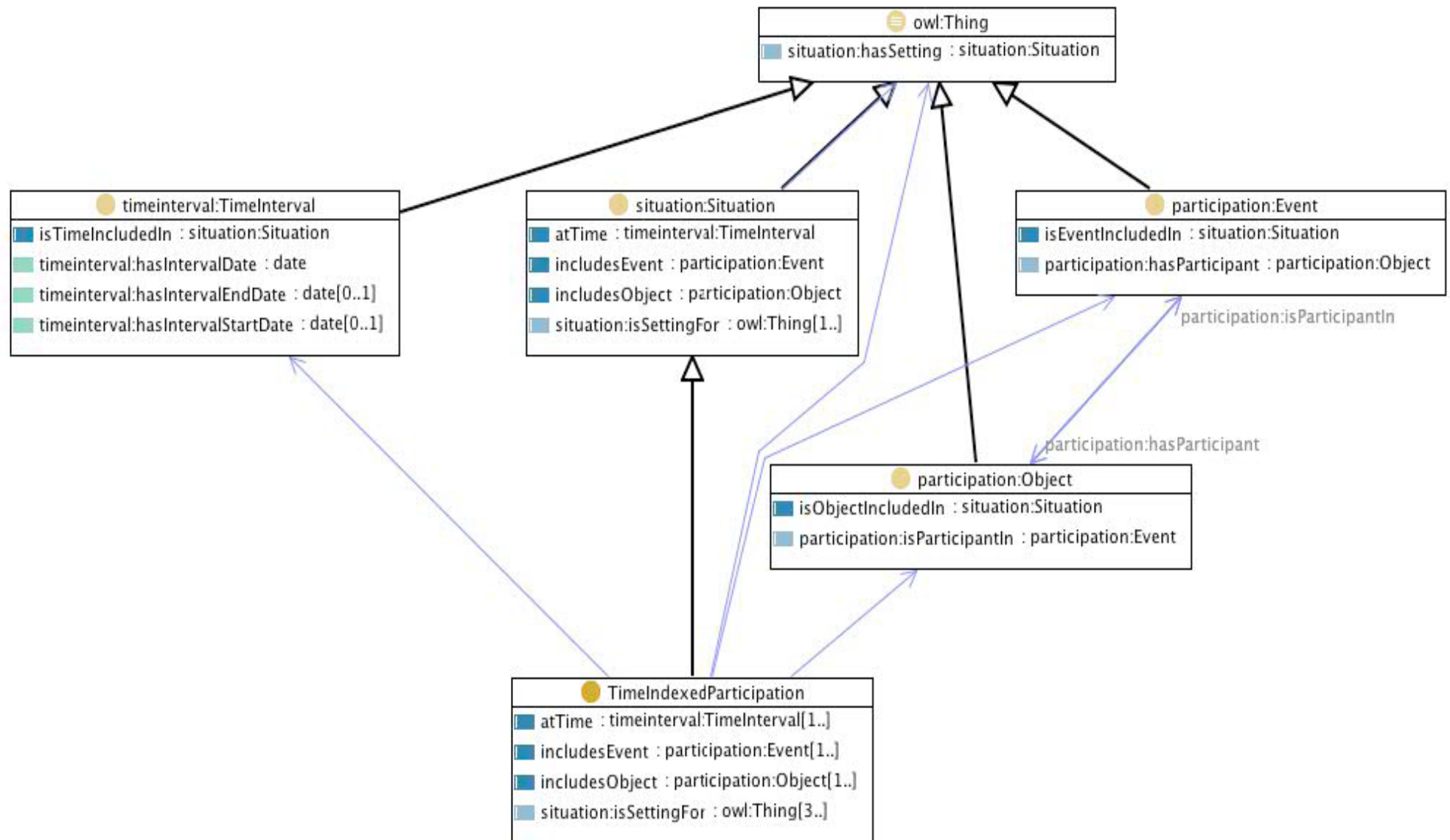
## Partial matching

- The CP does not cover all aspects of the local cqs
- The local use case has to be partitioned into smaller pieces.
- One of these pieces will be covered by the selected CP.
- For the other pieces, other CPs have to be selected.
- All selected CPs have to be imported and **composed**.
- One additional usage operation is identified:
  - composition

# Composition

- The composition operation relates two CPs and results into a new ontology
- The resulting ontology is composed of the union of the ontology elements and axioms from the two CPs, plus the axioms (e.g. disjointness, equivalence, etc.) that are added in order to link the CPs
- The composition of CP1 and CP2 consists of creating a semantic association between CP1 and CP2 by adding at least one new axiom, which involves ontology elements from both CP1 and CP2
- Typically, also new elements (“expansion”) are added when composing

# Sample composition





## Partial matching example

- For example, consider the following competency questions:
  - cq1 : who did play a specific role in a certain period?
  - cq2 : which role does a certain person have at a certain time?
- From previous examples we know that
  - **agent role** covers partially cq1 and cq2, as it allows to represent agents and the role they play
  - **time interval** covers partially cq1 and cq2, as it allows to represent time intervals
- The ontology resulting from the composition of these two CPs covers both cq1 and cq2

# Expansion

- *Expansion* consists of adding new ontology elements and axioms to a CP.
- The resulting ontology is composed of the ontology elements and axioms of the CP, plus the added ontology elements and axioms.

## Where do CPs come from?

- Content ontology design patterns (CPs) come from the experience of ontology engineers in modeling foundational, core, or domain ontologies
- There are four ways of creating CPs, which can be summarized as follows:
  - Reengineering from patterns expressed in other data models
    - Data model patterns, Lexical Frames, Workflow patterns, Knowledge discovery patterns, etc.
  - Specialization/Generalization/Composition of other CPs
  - Extraction from reference ontologies (by *cloning*)
  - Creation by combining extraction, specialization, generalization, composition, and expansion

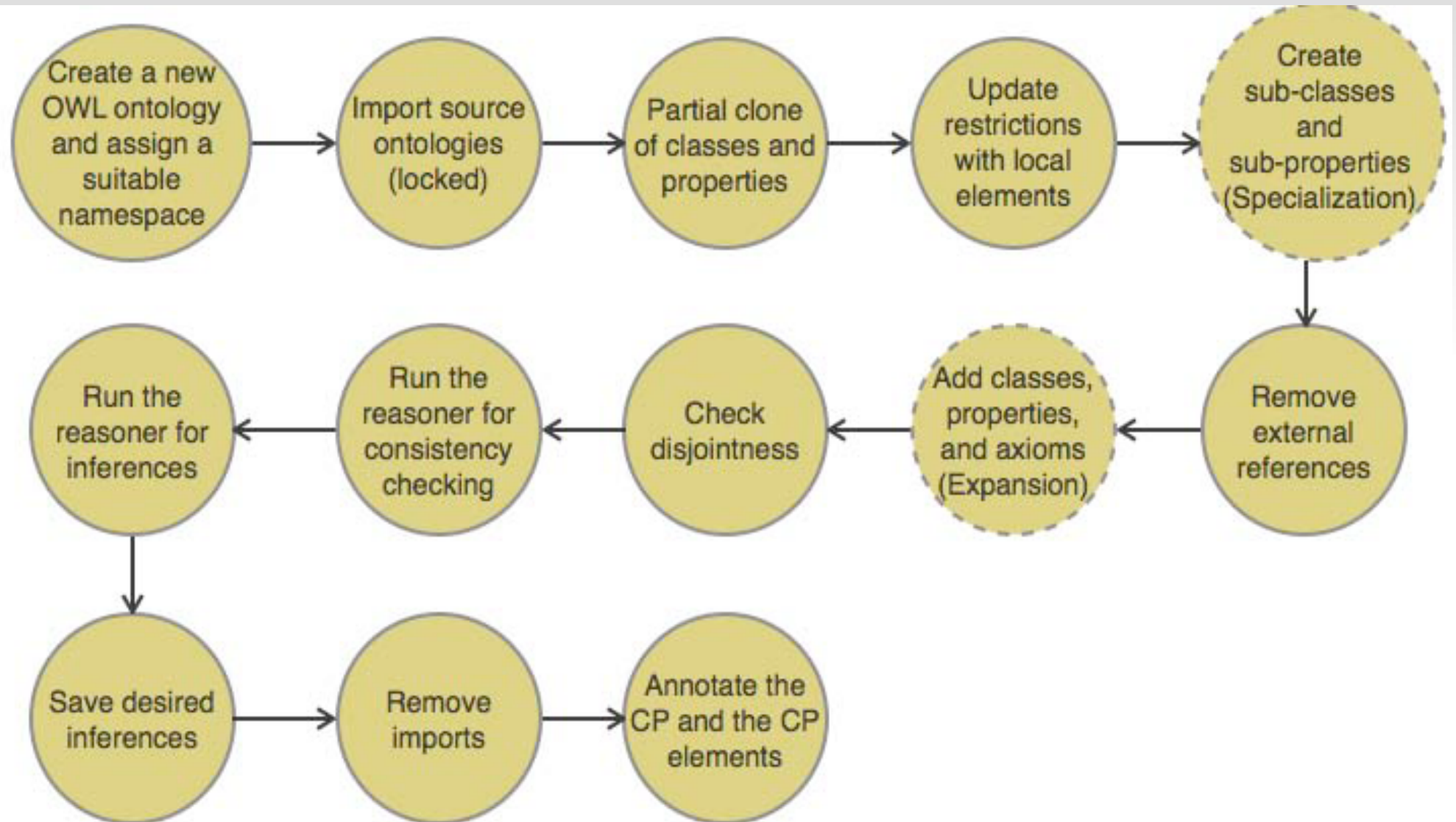
# Clone

- The extraction process relies on the *clone* operation
- The clone operation consists of duplicating an ontology element, which is used as a prototype.

# Types of clone operation

- Shallow clone
  - consists of creating a new ontology element oe2 by duplicating an existing ontology element oe1 . OWL restrictions of and axioms defined for oe1 and oe2 will be exactly the same
- Deep clone:
  - consists of creating a new ontology element oe2 by duplicating an existing ontology element oe1 , and by deep-cloning a new ontology element for each one that is referred in oe1 's axiomatization, recursively
- Partial clone:
  - consists of deep-cloning an ontology element, but by keeping only a subset of its axioms, and of partial-cloning the kept elements, recursively
- Some ontology design tools support the shallow clone operation
  - e.g., TopBraid Composer
- Deep clone and partial clone are not yet supported by any existing tool.

# The extraction process



## CP definition (finally!)

### *Definition*

- CPs are distinguished networked ontologies and have their own namespace
- They cover a specific set CQ of competency questions (requirements), which represent the problem they provide a solution for
- A CP emerges from existing conceptual models and can be **extracted from** a reference ontology (based on the clone operation), can be **reengineered from** other conceptual models (e.g. data models), can be **created by composition** of other CPs, by **expansion** of a CP, and either by **specialization** or **generalization** of another CP

# Pattern-based ontology design method:

- Main principles
  - divide & conquer
    - understand the task and express it by means of *competency questions*
  - *re-use “good” solutions i.e., ontology design patterns*
  - *evaluate the result against the task*
    - *Transform cq in SPARQL queries*



# Sample iteration

- Sentence: *Charlie Parker is the alto sax player on Lover Man, Dial, 1946*
  - *Charlie Parker (person)*
  - *the alto sax player (player role)*
  - *on Lover Man (tune)*
  - *Dial (publisher)*
  - *1946 (recording year)*
- **CQs**
  - *what persons do play a musical instrument?*
  - *on what tune?*
  - *for what publisher?*
  - *in what recording year?*
- **Queries**
  - *SELECT ?x ?y WHERE { ?x ?r ?y . ?x a :Person . ?y a :PlayerRole }*
  - *SELECT ?x ?z WHERE { ?x ?r ?y . ?x a :Person . ?x ?s ?z . ?z a :Tune }*
  - *SELECT ?z ?w WHERE { ?z ?t ?w . ?z a :Tune . ?w a :Publisher }*
  - *SELECT ?z ?k WHERE { ?z :recordingYear ?k . ?z a :Tune . ?k a xsd:gYear }*

## cont.d

- Retrieve/Match cqs to CPs, or possibly propose new ones
  - agentrole.owl, timeindexedpersonrole.owl, timeinterval.owl, ...
- Specialize/Compose/Expand CPs to local cq terminology
  - person-playerrole, playing-instrument-on-a-tune, playing-on-a-tune-in-recordingyear
- Populate Abox
  - Person(CharlieParker), PlayerRole(AltoSaxPlayer), Tune(LoverMan), Session(LoverManWithParkerOnDial), ..
- Run unit test/Iterate until fixed

```
SELECT ?x ?y ?z ?w ?k
```

```
WHERE {
```

```
    ?x ?r ?y .
```

```
    ?x a :Person .
```

```
    ?y a :PlayerRole .
```

```
    ?x ?s ?z .
```

```
    ?z a :Tune .
```

```
    ?z ?t ?w .
```

```
    ?w a :Publisher .
```

```
    ?z :recordingYear ?k .
```

```
    ?k a xsd:gYear }
```

- **RESULTS:** ?x=CharlieParker ?y=AltoSaxPlayer ?z=LoverMan ?w=Dial ?k=1946

# iteration with Content OPs

- Requirements are divided into small stories
- Get your story (local problem)
- divide & conquer
  - read carefully the story and divide them into simple sentences  $s_1, \dots, s_n$
- **FOR EACH SENTENCE  $s_i$** 
  - *transform  $s_i$  to an instance-free sentence (“abstraction”)*
    - *an instance can be either an individual or a property value (fact)*
  - *transform the instance-free sentence to local competency questions (cqs)*
  - *translate local cqs to queries to be submitted to the knowledge base, and collect them in a unit test [12]*
  - *match the CP coverage to the local cqs*
    - *identify the CPs you need, and associate each CP with the local cqs it covers*
    - *if any local competency question remains uncovered, define separate small ontologies that cover them, and import them into the ontology. Treat these as CPs*
  - *identify ontology elements to be specialized, and specialize them*
  - *identify axioms and ontology elements to involve in the composition of chosen CPs, and compose them*
  - *expand the ontology in order to cover uncovered competency question*
  - *populate the ontology ABox with the instances from the story*
  - *test using the collected queries and fix until all tests succeed*
- **END FOR**

# Ontology evaluation

- Domain: entity types, expertise patterns
  - is the ontology appropriate to context?
- Task: competency questions
  - is the ontology appropriate to support relevant queries?
- Resources: tools and personnel
  - is the ontology (structure, function, annotations) manageable and costeffective?
- Direct measuring of graphs and annotations
- Black-box/glass-box measuring of admissibility wrt conceptualization
- Indirect measuring via user feedback, and correlation
- Principles, diagnosis and trade-offs

# Bibliography / Links

- Valentina Presutti and Aldo Gangemi. *Content Ontology Design Patterns as Practical Building Blocks for Web Ontologies*. In Proceedings of the 27th International Conference on Conceptual Modeling (ER 2008)
- Aldo Gangemi, Carola Catenacci, Massimiliano Ciaramita, Jos Lehmann. *Modelling Ontology Evaluation and Validation*. Y. Sure (ed.), Proceedings of the Third European Semantic Web Conference, Springer, 2006.
- [http://ontologydesignpatterns.org/index.php/Training:NeOn\\_2008\\_Tutorial\\_on\\_Computational\\_Ontologies](http://ontologydesignpatterns.org/index.php/Training:NeOn_2008_Tutorial_on_Computational_Ontologies)
- <http://www.neon-project.org/>
- <http://www.topquadrant.com/topbraid/composer/index.html>