Ontology Design

Aldo Gangemi
Semantic Technology Lab
ISTC-CNR, Rome, Italy
aldo.gangemi@cnr.it

Thanks to: Valentina Presutti and the members of the STLab
Outline

• The world of ontology design
• OWL, design, and logical layers
• Ontology design patterns
• Sample design issues and unit tests
• Summary
An ontology designer’s world

- Requirements (*I want to attend my ideal talk*)
- Logical constructs (*subClassOf, restriction, ...*)
- Existing ontologies (*FOAF, BibTex, SWC, DOLCE, ...*)
- Informal knowledge resources (*CiteSeer, ACM topic catalog*)
- Conventions and practices (*naming/URI making, disjoint covering, reification patterns, transitive partOf, role-task, ...*)
- Tools: editors, reasoners, translators, etc. (*Protégé, NeOn Toolkit, TCB, FaCT++, Pellet, SMW, Jena, AllegroGraph, Virtuoso, ...*)
A well-designed ontology ...

• Obeys to “capital questions”: true anywhere?
  – What are we talking about?
  – Why do we want to talk about it?
  – Where to find reusable knowledge?
  – [also: Do we have the resources to maintain it?]
• Whats, whys and wheres constitute the Problem Space of an ontology project
• Ontology designers need to find solutions from a Solution Space
• Matching problems to solutions is not trivial
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, e.g. the world of Semantic Web conferences

• 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. research topics as entities that are dealt with by a project, worked on by academic staff, and can be topic of documents, events, etc.
  – Serve a purpose/task/competency question, e.g. finding persons that work on a same topic, matching project topics to staff competencies, time left, available funds, 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
Design in C-ODO

Also tools that support:
- pattern-based design
- evaluation and selection
- reengineering
- reasoning and querying
- evolution and mapping
What we can do with OWL

• ... (maybe) we can check the consistency, classify, and query all this knowledge
• this is great, but ...
• ... when I locally reuse parts of such a big bunch of knowledge, inferences sometimes produce strange results:
  – a web page same as an email address (e.g. http://.../Aldo owl:sameAs mailto://aldo@...)
  – a person same as a wikipedia article (e.g. Aldo owl:sameAs http://en.wikipedia.org/Aldo)
  – Italy is a continent (e.g. (Italy rdf:type (Country) rdfs:subClassOf Continent))
  – ...
• ... and problems are hardly fixable on a large scale
• Logical consistency is not the main problem
  – e.g. owl:sameAs can be wrongly used and still we have consistency
• Why OWL is not enough?
When to use owl:Individual, Class, ObjectProperty, DatatypeProperty?

• 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 an individual, a class, or an object property can be quite arbitrary
  – cf. Semantic Web Interest Group post May 27th, 2008 by Zille Huma:
  – "I have been wondering for sometime now that why isn't it a popular trend to store standard activities of a domain in the ontology and not only the concepts, e.g., for the tourism domain, ontologies normally contain concepts like Tourist, Resort, etc. but I have not so far come across an ontology that also contains the standard activities like searchResort, bookHotel, etc. Why is it so? What support is provided in the ontology languages to model the standard activities of the domain as well?"
  
• (1) a functionality for searching resorts is implemented in our web service
  – owl:Individual(searchResort) rdf:type(Functionality)

• (2) searching resorts is a type of functionality required for this kind of services
  – owl:Class(searchResort) rdfs:subClassOf(Functionality)

• (3) who has been searching for what resorts in our web service?
  – owl:ObjectProperty(searchResort) rdfs:range(Resort)

• (4) how many users have been using our resort searching functionality?
  – owl:DatatypeProperty(searchResort) rdfs:range(xsd:boolean)
Introduction to the Semantic Web Tutorial

Logical layers, types of entities, and contexts

- **Ontology**
  - **Meta-level Theory**
    - (epistemically)
    - **Appendicectomy for Durban's school can be performed by ...**
  - **Meta-level Theory**
    - (semantically)
    - **Appendicectomy is a class**
  - **Meta-level Theory**
    - (syntactically)
    - **Appendicectomy is a compound word**
  - **First-order Theory \(\approx\) TBox**
    - (incl. classes, relations)
    - **An appendicectomy is a surgical removal of the vermiform appendix**
  - **Knowledge Base \(\approx\) ABox**
    - (incl. individuals, facts)
    - **John had an appendicectomy**

**Community Knowledge**

**Formal Entities**

**Information**

**Meanings**

**Facts, Situations**
Pattern-based design

• Ontology design is presented here as the activity of searching, selecting, and composing different patterns
  – Logical, Reasoning, Architectural, Naming, Reengineering, Content
  – Common framework to understand modelling choices (the "solution space") wrt task- and domain-oriented requirements (the "problem space")
  – http://www.ontologydesignpatterns.org
Kinds of ontology design patterns
Logical patterns (LPs). Definition

• Logical constructs or composition of them
• LPs are content-independent structures expressed only by means of a logical vocabulary (plus possible primitives, e.g. “owl:Thing”)
• They can be applied more than once in the same ontology in order to solve similar modeling problems
• Logical patterns presented here are specific to OWL (DL)
Some LPs: Subsumption Macros

**subsumption by class**: bibtex:University instances are also bibtex:Organization instances.

**subsumption by restriction**: bibtex:University instances can only have bibtex:Department instances as Parts (!)

**equivalence by intersection**: European universities are universities that are located in Europe.
Some LPs: **N-ary relation**

- How to represent a relation with n arguments

  ![Diagram](Image)

  - Person
  - University, Location, Course ...
  - TimeSpan

- Cf. W3C SWBPD, logical reification, DLR, UML association class
- Issue: identification constraint
Content Patterns (CPs): Definition

• Instances of LPs or of compositions of LPs.
• Domain-dependent
  – Expressed with a domain specific (non-logical) vocabulary
• Solve domain modelling problems (expressible as tasks or “competency questions”)
• Affect the specific part of the ontology dealing with the related domain modelling problem
• Examples:
  – PartOf, Participation, Plan, Medical Guideline, Sales Order, Research Topic, Legal Contract, Inflammation, Situation, TimeInterval, etc.
The ODP portal

- A catalogue of CPs
  - http://www.ontologydesignpatterns.org (odp-web)
  - catalogue entry
- Annotation properties:
  - http://www.ontologydesignpatterns.org/schemas/cpannotationschema.owl
  - annotation of OWL implementation of CPs
Example 1: Agent Role

The AgentRole Content OP locally defines the following ontology elements:

Agent (owl:Class)

Any agentive Object, either physical, or social.

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.
Agent Role Instantiation

- Scenario: Aldo Gangemi is a senior researcher. He is also father and saxophonist.
Example 2: Time Interval

<table>
<thead>
<tr>
<th>TimeInterval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted by: Valentina Prosubi</td>
</tr>
<tr>
<td>Name: Time interval</td>
</tr>
<tr>
<td>Also Known As:</td>
</tr>
<tr>
<td>Intent: To represent time intervals.</td>
</tr>
<tr>
<td>Domains: Time</td>
</tr>
<tr>
<td>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?</td>
</tr>
<tr>
<td>Reusable OWL: <a href="http://wwwontologydesignpatternsorg/op/owl/timeintervalowl">http://wwwontologydesignpatternsorg/op/owl/timeintervalowl</a></td>
</tr>
<tr>
<td>Building Block:</td>
</tr>
<tr>
<td>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.</td>
</tr>
<tr>
<td>Known Uses: Web</td>
</tr>
<tr>
<td>References:</td>
</tr>
<tr>
<td>Other References:</td>
</tr>
</tbody>
</table>

Elements

The TimeInterval Content OP locally defines the following ontology elements:

- **TimeInterval (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 date Time, 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
Example 3: PartOf

This also uses transitivity reasoning pattern

Example 4: Time-indexed Participation

This also uses N-ary logical pattern
Example 5: Role-based Participation
Other applied CPs
Specializing patterns

• Same structure down the taxonomy hierarchy
• A CP $p_2$ specializes another $p_1$ when at least one of the classes or properties from $p_2$ is a sub-class or a sub-property of some class or property from $p_1$, while the remainder of the CP is identical.

• Participation (of an object in an event)
  – Taking part in a public enterprise activities
    • Funding a Semantic Web project

• Co-participation
  – Having a social relationship
    • Being bunkmates

• Renaming elements of an imported patterns is a bad practice
  – Specializing is the way of using CPs
Composing patterns

• Linking sensible classes on the background of a common (or integrated) reference ontology

• A CP $p_2$ extends $p_1$ when $p_2$ contains $p_1$, while adding some other class, property, or axiom

• A CP $p_3$ integrates $p_1$ and $p_2$ when $p_3$ contains both $p_1$ and $p_2$

• A CP $p_3$ merges $p_1$ and $p_2$ when $p_3$ contains both $p_1$ and $p_2$, and there exist explicit links between at least two classes or properties from both $p_1$ and $p_2$

• $BiochemicalTreatment \rightarrow (Role \Leftrightarrow Task \ \° \ \ Description \Leftrightarrow Situation \ ° \ Substance \Leftrightarrow Agent \ ° \ Time-indexedParticipation)$
A quick test: the SWC ontology

• Patterns used
  – Logical patterns
    • N-ary: as in *Product*
  – Content patterns
    • *Topic* pattern: obeys some tasks, generic coverage
  – Architectural patterns: *Alignment without import* to schemas used in applications: FOAF, SWRC, iCAL, WordNet1.6
  – Naming patterns
The “topic” module as extracted from the SWRC ontology
Design evaluation

• Coverage: *topics, staff, projects, dealt with by, worked on by, being a topic of*
• Task: *reasoning on semantic web entities*
• Does the *topic* pattern satisfy coverage and task requirements?
Best practice check

- **Check that names are intuitive**
  - **Antipattern**: using a generic name for a subclass of class that have a specific name:
    - `Artefact subClassOf wn:Document`
Counterintuitive naming
Finding what documents have a same topic

- **Impossible**: hasTopic not an inverse of isTopicOf (!),
- **Workaround**: use SPARQL query
- **Also**: Document class detached from the pattern
- Minor problem for task, but implies design "sparseness"
- **Also**: topics related to papers are instances of DBpedia:Topic, not from the list of individuals from swrc:ResearchTopic
- **Fix**: equivalence axiom between swrc:ResearchTopic and DBpedia:topic
Task-based unit test 2

– Checking that only events can be sub-events (“atEvent”) of other events (universal restriction)
  • Impossible: Event is not disjoint from e.g. Document
  • Consequence: e.g. a document that is said “atEvent” of an event, will be an event as well
Task-based unit test 3

- **Finding all parts of the proceedings**
  - **Impossible**: swc:hasPart and swc:isPartOf are not Transitive (and not Inverses)
  - **Consequence**: e.g. a paper that is part of a section of the proceedings will not be part of the proceedings; a laboratory that is part of a department of a university will not be part of the university; that department will not be asserted to have the laboratory as part
  - **Also**: no relation between transitive part for events (swc:subEvent), and the generic hasPart
  - **Fix**: apply partOf patterns (e.g. SWBPD, ODP patterns), with **Transitive Reduction** pattern: transitive version of a property should be the more generic
Sample eXtreme Design iteration

• Sentence: *Charlie Parker is the alto sax player on Lover Man, Dial, 1946*
  - Charlie Parker (person)
  - the alto sax player (player role)
  - Lover Man (tune)
  - Dial (publisher)
  - 1946 (recording year)

• Competency Questions
  - what persons do play a musical instrument?
  - on what tune?
  - for what publisher?
  - in what recording year?

• Queries
  - 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 }

*What’s the role of NLP in ontology design? ;)*

*Alternative abstractions do exist!*
• 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
  – 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 }

 cf. Test-Driven Development in XP
Contribute to the collaborative design effort!

- http://www.ontologydesignpatterns.org
- http://www.neon-project.org
- http://www.w3.org/2001/sw/BestPractices/
Summary

• Ontology design and ontology evaluation
• Problem space vs. Solution space
• Ontology design patterns
  – CP are ontology building blocks that allow design by re-engineering (including OL), specialization and composition
  – With XD, they come with embedded task-based ontology evaluation
Some references


