Best practices in ontology design

Tutor: Aldo Gangemi
Assistant tutor: Valentina Presutti
Lecture @ SSSW 2009
Cercedilla, Spain
Tutor info

- Name: Aldo Gangemi
- Institute: ISTC-CNR, Rome, Italy
- Research Group: Semantic Technology Laboratory (STLab)
- Roles: Senior Researcher, Head of STLab
- Research interest: ontology design, KE+NLP, ...
Outline

• Ontology design
• Knowledge resources and OWL
• Pattern-based ontology design
• Pattern types
Computational ontologies

• Ontologies as (software) components, expressed and managed in standard W3C languages like RDF, OWL, RIF, SPARQL, Fresnel, etc.
• Ontology design is the core aspect
• Quality is associated with good design
• STLab people research from 2004-5: “A formal framework for ontology evaluation and selection” [5]
Quality

• Three quality dimensions: Structural-Content-Sustainability
  – Content is the primary dimension
• Content compliance spans Coverage-Task-SelfExplanation
  – Task is the immediately measurable aspect
  – Quality is not maximal and abstract, but bound to context
  – Partial orders of problems and reusable solutions (locality)
  – Good practices (history)
• Empirical methods for evaluation (measurability)
What is ontology design? 1/3

• Computational Ontologies are artifacts
  – Have a structure (linguistic, logical, etc.)
  – Their function is to “encode” a description of the world (actual, possible, counterfactual, impossible, desired, etc.) for some purpose
What is ontology design? 2/3

• Ontologies must match both domain and task
  – Allow the description of the entities (“domain”) whose attributes and relations are concerned because of some purpose
    • e.g. social events and agents as entities that are considered in a legal case, research topics as entities that are dealt with by a project, worked on by academic staff, and can be topic of documents, etc.
  – Serve a purpose (“task”), e.g. finding entities that are considered in a same legal case, finding people that work on a same topic, matching project topics to staff competencies, time left, available funds, etc.
What is ontology design? 3/3

• Ontologies have a lifecycle
  – They 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 solutions (incl. patterns), Interaction
Design aspects in *Kali-mashup*
Two kinds of ontologies

• Coverage-oriented ontologies
  – They cover the terminology/metadata/textual corpora/folksonomies ... that fit a specific domain [big reengineering problem - exploited for annotation, retrieval, etc.]

• Task-oriented ontologies
  – They are able to give a structure to a knowledge base that can be used to answer competency questions [big design and reuse problem - exploited for automated reasoning and querying]

• Currently
  – A mass of heterogeneous data and ontologies, either expressed or portable to RDF (DB lifting, rdf-ized sources, etc.)
  – with generally low quality in some quality dimension/aspect
Ontology-related data in the web suq ("knowledge resources")

- Informal through formal
- Reengineering-oriented
  - Text corpora
  - Lexica (dictionaries, wordnets, terminologies, nomenclatures)
  - Knowledge organization systems (thesauri, classification schemes)
  - Folksonomies (tag sets, directories, topic trees, subject indexes)
  - Frames, semantic networks
  - Microformats, infoboxes, HTML templates
  - DB schemas and records (RDBS, XML, proprietary or not)
- Reuse-oriented
  - LOD datasets
  - RDFa, eRDF
  - DC, FOAF, SIOC, SKOS, etc. (RDFS vocabularies)
  - OWL ontologies
Standard languages help

• Transform all in RDF, or even OWL
  – Cf. *Triplify* initiative
  – Datasets extracted from heterogeneous sources, and triplified
  – Semantics depends on intended task of data and relations used for linking

• Then search/visualize RDF data, or make integrating applications
Knowledge search over the semantic web

   - http://morpheus.cs.umbc.edu/aks/ontology/ks#Desire
   - http://morpheus.cs.umbc.edu/aks/ontology/ks#Goal
   - http://morpheus.cs.umbc.edu/aks/ontology/ks#NonAchievableDesire
   - http://morpheus.cs.umbc.edu/aks/ontology/ks#NonConflictingDesire

   - http://www.nuina.org/ontology/ks#Desire
   - http://www.nuina.org/ontology/ks#Goal
   - http://www.nuina.org/ontology/ks#NonAchievableDesire
   - http://www.nuina.org/ontology/ks#NonConflictingDesire

3. http://mogatu.umbc.edu/ont/2004/01/BDI.owl#BDI
   - http://mogatu.umbc.edu/ont/2004/01/BDI.owl#NonAchievableDesire
   - http://mogatu.umbc.edu/ont/2004/01/BDI.owl#NonConflictingDesire
   - http://mogatu.umbc.edu/ont/2004/01/BDI.owl#Desire

   - http://pervasive.semanticweb.org/ont/2004/06/bdi#Desire
   - http://pervasive.semanticweb.org/ont/2004/06/bdi#Desire

   - http://pervasive.semanticweb.org/ont/dev/bdi#Desire
   - http://pervasive.semanticweb.org/ont/dev/bdi#Desire


Knowledge search over the LOD cloud

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constitution</td>
<td>comment</td>
<td>En konstitusjon er et sett med grunnleggende prinsipper for et konstitusjon er ofte kodifisert i stiftelsesdokument.</td>
</tr>
<tr>
<td>Constitution</td>
<td>comment</td>
<td>La costituzione di un’organizzazione struttura, attività, carattere e derivata dal latino constituto, che legge di particolare importanza dell’imperatore ed è tutt’ora usi indicare decisioni rilevanti per costituzione apostolica che fiss il periodo di sede vacante e per conciliare, del nuovo vescovo di comune del termine rimanda, si fondamentale di uno stato. Que risico di significati, sia descrittiv</td>
</tr>
<tr>
<td>Constitution</td>
<td></td>
<td>En forfatning eller grundlovs lag</td>
</tr>
</tbody>
</table>

[Image of Data Explorer interface with triple statements and links]
Now we have all those data expressed in a language that allows semantic interoperability ...
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, owl:Class, owl:ObjectProperty, owl: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
New problems arising on the Web...

- 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) “searching resorts is a type of functionality required for this kind of services”
  - owl:Class(searchResort) rdfs:subClassOf(Functionality)

- (2) “a functionality for searching resorts is implemented in our web service”
  - owl:Individual(searchResort) rdf:type(Functionality)

- (3) “who has been searching for what resorts in our web service?”
  - owl:ObjectProperty(searchResort) rdfs:domain(Customer) rdfs:range(Resort)

- (4) “how many users have been using our resort searching functionality?”
  - owl:DatatypeProperty(searchResort) rdfs:domain(Customer) rdfs:range(xsd:boolean)
Solutions?

• ... 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.

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

• ... provided that tools are built for any user 😊
An ontology designer’s world

- Requirements (e.g. “I want to attend my ideal talk”)
- Logical constructs (rdfs:subClassOf, owl:Restriction, ...)
- Existing ontologies (FOAF, BibTex, SWC, DOLCE, ...)
- Informal knowledge resources (CiteSeer, ACM topic catalog)
- Conventions and practices (e.g. naming, URI making, XML2OWL, SKOS, disjoint covering, reification methods, transitive partOf, role-task, ...)
- Tools: editors, reasoners, translators, etc. (Protégé, NeOn Toolkit, TBC, FaCT++, Pellet, OWLAPI, Jena, AllegroGraph, Virtuoso, ...)
A well-designed ontology ...

- Obeys to “capital questions”:
  - What are we talking about?
  - Why do we want to talk about it?
  - Where to find reusable knowledge?
  - 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
Ontology Design Pattern

An ontology design pattern is a reusable successful solution to a recurrent modeling problem.
Pattern-based design
aka eXtreme Design (XD)

• Pattern-based ontology design is the activity of searching, selecting, and composing different patterns
  – Logical, Reasoning, Architectural, Naming, Correspondence, Content
  – Common framework to understand modeling choices (the “solution space”) wrt task- and domain-oriented requirements (the “problem space”)
  – [http://www.ontologydesignpatterns.org](http://www.ontologydesignpatterns.org) is the place that has started collecting ODPs
eXtreme ontology Design (XD)

• Inspired by eXtreme Programming basic rules
  – e.g., pair programming, test-oriented, continued integration, etc.
  – but XP focuses on refactoring rather that on “design”
• 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
• An example will be presented and implemented in the afternoon
Types of Ontology Design Patterns (ODPs)

- We also distinguish between ontological resources that are not ODPs and Ontology Design Anti-Patterns (AntiODP)
Architectural ODPs

Definition

Architectural ODPs affect the overall shape of the ontology: their aim is to constrain ‘how the ontology should look like’
Patterns in the FSDAS Ontology Network
PaCerns in the CNR Ontology Network
Patterns in the CODO Ontology Network
What criteria for architecture?

• Manageability, work distribution
  – how large the sections? how to partition?
  – weak evidence of good results when doing partitioning ex post

• Requirements ("competency questions")
  – cf. fisheries example, modelling issues for fsdas

• Broad topics
  – when no requirements available, or with large , lightweight ontologies
Logical ODPs

• cf. Sean’s tutorial
Content ODPs
• CPs encode conceptual, rather than logical design patterns.
  – Logical ODPs 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
Content ODPs (CPs) 2/3

- CPs are instantiations of Logical ODPs (or of compositions of Logical ODPs), featuring a non-empty signature
  - Hence, they have an explicit non-logical vocabulary for a specific domain of interest, i.e. they are content-dependent
Content ODPs (CPs) 3/3

• 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)

• A typical way of capturing requirements is by means of competency questions
Catalogues of CPs 1/2

• Content ODPs are collected and described in catalogues and comply to a common presentation template
• The ontologydesignpatterns.org initiative maintain a repository of CPs and a semantic wiki for their description, discussion, evaluation, certification, etc.
Catalogues of CPs 2/2

Ontology Design Patterns . org (ODP)

OntologyDesignPatterns.org is a Semantic Web portal dedicated to ontology design patterns (OPs). The portal started under the NeOn project that still partly supports its development (http://www.neon-project.org/).

Latest ODP News:
- 2 April 2008 10:10:52 The Loreto of Ontology Design Patterns (by VioletaDjuricjanovic)
- 21 October 2008 12:12:59 EvalWF has been released (by EnricoDaga)
- 5 June 2008 11:11:54 News at ODP portal (by EnricoDaga)

There is much more... Check out the about ODP page for more information on the portal content and structure.

If you have no idea what we are talking about, visit the "What is a pattern?" page.
Presentation template of CPs on ontologydesignpatterns.org
Pragmatic characteristics of CPs

- Domain-dependent
  - Expressed with a domain-specific (non-logical) vocabulary
- Requirement-covering
  - Solve domain modeling problems (expressible as use-cases, tasks or “competency questions”), at a typical maximum size (cf. blink)
- Reasoning-relevant components
  - Allow some form of inference (minimal axiomatization, e.g. not an isolated class)
- Cognitively-relevant components
  - Catch relevant core notions of a domain and the related expertise -- blink knowledge
- Linguistically-relevant components
  - Are lexically grounded, e.g. they match linguistic frames, or at least a domain terminology
- Examples:
  - PartOf, Participation, Sequence, Plan, Legal Norm, Legal Fact, Sales Order, Research Topic, Legal Contract, Inflammation, Medical Guideline, Gene Ontology Top, Situation, Time Interval, Tagging, Aquatic Resources ...
### Generic ontology requirements

<table>
<thead>
<tr>
<th>Generic Competency Questions</th>
<th>Specific Modelling Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Who does what, when and where?</td>
<td>Production reports, schedules</td>
</tr>
<tr>
<td>Which objects <strong>take part in</strong> a certain event?</td>
<td>Resource allocation, biochemical pathways</td>
</tr>
<tr>
<td>What are the <strong>parts</strong> of something?</td>
<td>Component schemas, warehouse management</td>
</tr>
<tr>
<td>What’s an object <strong>made of?</strong></td>
<td>Drug and food composition, e.g. for safety (comp.)</td>
</tr>
<tr>
<td>What’s the <strong>place</strong> of something?</td>
<td>Geographic systems, resource allocation</td>
</tr>
<tr>
<td>What’s the <strong>time frame</strong> of something?</td>
<td>Dynamic knowledge bases</td>
</tr>
<tr>
<td>What technique, method, practice is being used?</td>
<td>Instructions, enterprise know-how database</td>
</tr>
<tr>
<td>Which tasks should be <strong>executed</strong> in order to achieve a certain goal?</td>
<td>Planning, workflow management</td>
</tr>
<tr>
<td>Does this behaviour <strong>conform</strong> to a certain rule?</td>
<td>Control systems, legal reasoning services</td>
</tr>
<tr>
<td>What’s the <strong>function</strong> of that artifact?</td>
<td>System description</td>
</tr>
<tr>
<td>How is that object <strong>built?</strong></td>
<td>Control systems, quality check</td>
</tr>
<tr>
<td>What’s the <strong>design</strong> of that artifact?</td>
<td>Project assistants, catalogues</td>
</tr>
<tr>
<td>How did that phenomenon happen?</td>
<td>Diagnostic systems, physical models</td>
</tr>
<tr>
<td>What’s your <strong>role</strong> in that transaction?</td>
<td>Activity diagrams, planning, organizational models</td>
</tr>
<tr>
<td>What that information is about? How is it realized?</td>
<td>Information and content modelling, computational models, subject directories</td>
</tr>
</tbody>
</table>

**What argumentation model are you adopting for negotiating an agreement?**  
Cooperation systems

**What the degree of confidence that you give to this axiom?**  
Ontology engineering tools
Imagine we have to model the following

- Arnold Schwarzenegger is Shylock in the performance of the play "Merchant of Venice", that is given at the theatre “Globe” during July 2009
Analyze the sentence, detect the modeling issues, and match to the CPs

• Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
• A person plays a character
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009.
- A person plays a character.
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- A person plays a character
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- A person plays a character

- To represents objects and the roles they play.
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- A person plays a character
- To represents objects and the roles they play.
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- The play of some drama
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- To distinguish information objects from their concrete realizations.

- The play of some drama
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- To distinguish information objects from their concrete realizations.

The play of some drama
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009
- A time period
Analyze the sentence, detect the modeling issues, and match to the CPs

• Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009

• A time period

• To represent time intervals, their start/end dates, and any dates falling into the period
Analyze the sentence, detect the modeling issues, and match to the CPs

• Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009

• A time period

• To represent time intervals, their start/end dates, and any dates falling into the period
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater "Roma" during September and October 2009
- A person plays a character in a play of a drama, given at a theater during a time period
- How can we relate them together?
Analyze the sentence, detect the modeling issues, and match to the CPs

• Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009

• A person plays a character in a play of a drama, given at a theater during a time period

• A situation, a set of circumstances in a defined setting
Analyze the sentence, detect the modeling issues, and match to the CPs

- Arnold Schwarzenegger is Shylock in the play of "Merchant of Venice", that is given at the theater “Roma” during September and October 2009

- A person plays a character in a play of a drama, given at a theater during a time period

- A situation, a set of circumstances in a defined setting
Reengineering ODPs

- Reengineering ODPs contain transformation rules applied in order to create a new ontology (target model) starting from elements of a source model.
- The target model is an ontology, while the source model can be either an ontology, or any other knowledge resource.
  - e.g., a thesaurus concept, a data model pattern, a UML model, a linguistic structure, etc.
Relation to other ODPs

- Correspondence patterns can make use of other ODPs:
  - Name patterns
  - Logical patterns
  - Content ODPs

- Alignment patterns might result from linking two equivalent ODPs

- Correspondence patterns can be used together with content patterns for parts A and B

Ex: An ontology transformation pattern (refactoring pattern) making use of two Content patterns and an alignment between them to transform from one structure to the other. (See application at the end of the lecture)
Particularities of correspondence ODPs

This alignment corresponds to a Granularity Mismatch
Example: FOAF-VCard
Other alignment patterns

- Some alignment patterns are interpretation-preserving (IP), i.e. the semantics arising from the composition of the two resources stays in OWL-DL, while other patterns do not bear that characteristic (NIP), and therefore induce a non-standard semantics:
  - **Individual is equivalent to class** (NIP):
    - (m1) asfad:Aquatic_organisms owl:equivalentClass FI_commodities:FI_commodity_classification
    - (m2) asfad:Stocks owl:equivalentClass stock:Stock
    - (m3) asfad:Water_bodies owl:equivalentClass water:FAO_fishing_area
  - **Individual is subclass of class** (NIP):
    - (m4) asfad:Ocean_space rdfs:subClassOf water:FAO_fishing_area
  - **Individual has type class** (IP):
    - (m5) asfad:Research_vessels rdf:type vessels:vessel_classification
Reasoning ODPs are applications of Logical ODPs oriented to obtain certain reasoning results, based on the behavior implemented in reasoning engines.
Examples of Reasoning ODPs

• Precise
  – Classification
  – Subsumption
  – Inheritance
  – Materialization
  – Rule firing
  – Constructive query
  – ...
• Approximate
  – Approximate classification
  – Similarity induction
  – Taxonomy induction
  – Relevance detection
  – Latent semantic indexing
  – Automatic alignment
  – ...

or some workflow of them, e.g.
1) Materialize property chains
2) Run classifier+subsumer
3) Run constructive query
4) Assert results
5) Run similarity engine
6) Assert results
Presentation ODP

Presentation ODPs deal with usability and readability of ontologies from a user perspective
Examples of Naming ODPs

- Namespace declared for ontologies.
- It is recommended to use the base URI of the organization that publishes the ontology
  - e.g. [http://www.fao.org](http://www.fao.org) for the FAO
- followed by a reference directory for the ontologies
  - e.g. [http://www.fao.org/ontologies/](http://www.fao.org/ontologies/)
- It is also important to choose an approach for encoding versioning, either on the name, or on the reference directory
Benefits of naming patterns for humans (I)

- Example: **Elementary triple**
  - IBM *product* ThinkPad
- What does this say?
  - „IBM has a product called ThinkPad“?
  - or, actually, „IBM is a product of ThinkPad“?
- What is the issue?
  - Trivial violation of naming pattern: plain noun (‘product’) used as object property name
Benefits of naming patterns for humans (II)

- There can be multiple adequate naming patterns
- IBM produces ThinkPad
  - The ‘semantic direction’ of the relation is made clear
  - ‘produces’ can be understood as ‘manufactures’ as processive relationship
- IBM has_product ThinkPad
  - The use of ‘has’ can be sign of a stative relationship, often to a dependent entity
  - “IBM has ThinkPad as its product” (even if it is no longer manufactured)
Other examples of naming patterns

• Class names 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 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
Examples of Annotation ODPs

• 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
References (1)


References (2)


10. W3C Ontology Engineering and Patterns Task Force (OEP), http://www.w3.org/2001/sw/BestPractices/OEP/


