Ontology Design

Aldo Gangemi
Laboratory for Applied Ontology
ISTC-CNR, Rome, Italy
aldo.gangemi@istc.cnr.it
Thanks to: Valentina Presutti and the members of LOA

Outline

• The world of ontology design
• Ontologies and language
• Ontology design components
• Ontology design patterns
• The SWC ontology
• Summary
An ontology designer’s world

- Requirements
- Logical constructs
- Existing ontologies
- Informal knowledge resources
- Conventions and practices
- Tools (editors, reasoners, translators, ...)

The cultural context of ontologies

- Logic
- Cognitive and social sciences
- Empirical sciences
- Linguistics, Semiotics
- Computer science, AI
- Web science
- Philosophy
- Ontology engineering
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?
- 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”), 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
The modular architecture of (collaborative) ontology design

Ontologies and language

• Ontologies describe some domain (for some purpose)
• But also natural language can do it
• Ok, but natural languages are appropriate for humans, not for machines
• What’s the difference?
  – Humans share tacit knowledge (“presuppositions”) that provides the context for interpreting natural language utterances and texts
  – Some tacit knowledge is general
    • “US Army auditor who attacked Halliburton deal is fired”
    • auditor is a role played by persons within organizations
    • persons can “attack” others by denouncing something (e.g. a deal)
    • persons can be “fired” from a position (role)
  – Some is local
    • “US Army auditor who attacked Halliburton deal is fired”
    • denounced the decision to give billions of dollars in Iraq reconstruction contracts to a subsidiary of Vice-President Dick Cheney’s old company Halliburton
    • “She told a congressional hearing that the decision was “the most blatant and improper abuse I have witnessed” in 20 years as a government contract supervisor”
Ontologies = controlled terminologies?

• Beware the mismatch between language and conceptualization!
• An ontology may not just be a controlled terminology
• We may have to capture the conceptual schema (or pattern) underlying the use of a certain terminology, in order to make it reusable for design, interoperability, meaning negotiation, etc.
• Should ontologies be considered reference conceptual schemas?
• Indeed, that was the original motivation for ontologies. Cf. Ontolingua library, 1992
• Nowadays, it’s pretty different
  – Thousands of ontologies, many different uses, the most successful are very simple (DublinCore, FOAF, WSGeo, ...), huge uptake on folksonomies
• Need for simple schemas, which are close to users’ way of thinking

Pattern-based matching

• 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")
  – They are being collected in the NeOn catalogue that will be available at the beginning of 2008
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: \textit{N-ary relation}

- How to represent a relation with \( n \) arguments

- Cf. W3C SWBPD, logical reification, DLR, UML association class

---

Content Patterns (CPs): Definition

- Instances of LPs or compositions of LPs.
- Domain-dependent
  - Expressed with a domain specific (non-logical) vocabulary
- Solve domain modelling problems
- Affect the specific part of the ontology dealing with the related domain modelling problem
- Examples:
  - PartOf, Participation, Plans, Medical Guidelines, Sales Order, Research Topic, Legal Contract, Inflammation, Identity on the web, etc.
Some CPs: PartOf

Time-indexed Participation
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
    - Giving a grant to 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 \circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circ\circle
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” content pattern 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, DOLCE-Ultralite patterns), with Transitive Reduction pattern: transitive property as the more generic
Appendix:
Other types of ontology design patterns

Naming Patterns (NPs): Definition

• Conventions about how to create names for namespaces, files and ontology elements in general (classes, properties, etc.)
• Good practices that boost ontology readability and understanding by humans, by supporting homogeneity in naming procedures
Examples of NPs

• Class and property names
  – A good practice for naming classes and properties is to give them a readable name either directly to the class, or to its label (this allows tools to visualize hierarchies and diagrams with readable names)
  – Usually class names start with a capital letter, and if a class name is composed of more than one terms, they are concatenated without special characters between such terms and with each term starting with capital letter
    • e.g., Person, Car, PersonalComputer
  – Classes representing elements that are single should not contain plurals. If the class name contains plurals, it should represent elements that are collections
    • e.g., Nurses, Shoes, etc.
  – Usually property names does not start with capital letter, and if a property name is composed of more than one terms, they are concatenated without special characters between such terms and with each term but the first starting with capital letter
    • e.g., hasFriend, hasChild, reads, etc.

Reasoning Patterns (RPs): Definition

• Application of LPs oriented to obtain certain inferencing results, based on the behavior implemented in a reasoning engine
• They are inference schemas, depending on the inference rules defined for a language
• Examples: Classification, Subsumption, Inheritance, Materialization, Query result construction
Classification and Subsumption RPs

- **Automatic classification**
  - Yes-Man(x) =df Man(x) ∧ ∃y(hasFiancee(x,y))
  - Man(John)
  - hasFiancee(John,Mary)
  - ∴ Yes-Man(John)

- **Automatic subsumption**
  - Yes-Man(x) =df Man(x) ∧ ∃y(hasFiancee(x,y))
  - ItalianMan(x) ⇒ Man(x)
  - hasFrenchFiancee(x,y) ⇒ hasFiancee(x,y)
  - ∴ ((ItalianMan(x) ∧ ∃y(hasFrenchFiancee(x,y)) ⇒ Yes-Man(x))

Inheritance and Materialization RPs

- **Inheritance**
  - Man(x) ⇒ Human(x)
  - Yes-Man(x) ⇒ Man(x)
  - ∴ (Yes-Man(x) ⇒ Human(x))

- **Materialization**
  - hasFiancee(x,y) ⇔ hasFiance(y,x)
  - hasFiancee(John,Mary)
  - ∴ hasFiance(Mary,John)
Architectural Patterns (APs): Definition

- Equivalent to LPs (or compositions of them) that are used exclusively in the design of an ontology
- An AP is a content-independent structure
- It is supposed to characterize the overall structure of an ontology
- An AP dictates how the ontology should look like

Examples of APs

- Taxonomy
  - A hierarchical structure of classes only related by subsumption relations.
- Lightweight ontology. Taxonomy + other features, e.g.:
  - A class can be related to other classes through the disjointWith relation.
  - Object and datatype properties can be defined and used to relate classes.
  - A specific domain and range can be associated with defined object and datatype properties.
- Modular architecture
  - Structuring an ontology as a configuration of components, each having its own identity based on some design criteria
  - When an ontology is committed to a huge domain of knowledge, a good practice is to decompose the domain into smaller subdomains which address simpler tasks
  - Each subdomain can be then encoded in an ontology module, in order to provide the whole ontology with a modular architecture.
Stratified MP

• To create a layering of modules, according to some criterion

<table>
<thead>
<tr>
<th>Foundational ontology (domain-independent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object, Process, Part, Time, Location, Representation, Plan, …</td>
</tr>
</tbody>
</table>

Inherits from

<table>
<thead>
<tr>
<th>Core ontology (specific domain-independent)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work of art, Painting technique, Author, Artistic period, Plastic art, Interpretation, …</td>
</tr>
</tbody>
</table>

Inherits from

<table>
<thead>
<tr>
<th>Domain ontology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sculpture, Restoration, Mythical being, Caryatid, Doric order, Armilla, Fresco, …</td>
</tr>
</tbody>
</table>

Re-engineering Patterns (RePs): Definition

• Transformation rules to be applied in order to map elements of a source model (i.e. knowledge resource) to elements of a target model.

• The target model is an ontology, while the source model can be either an ontology, a thesaurus, a DB schema, a UML model, etc.
Knowledge resource types

• Modeling Languages:
  – E/R, UML, XSD, Petri Nets, ebXML, BPEL4WS

• Conceptual models:
  – Database schemas, UML diagrams, XSD schemas, etc.

• Informal Data Structures
  – Spreadsheets, tables, etc.

• Lexical resources:
  – WordNet, FrameNet, Oxford Dictionary, etc.

• Concept Schemes
  – Thesauri, classifications, nomenclatures, etc.

• Web 2.0 resources:
  – Wikipedia, Flickr, de.li.ci.o.us, etc.

• Natural Language documents

Example of ReP: from thesauri to ontologies in SKOS

• KOS ⇒ skos:ConceptSchema

• Descriptor ⇒ skos:Concept

• Broader Term ⇒ skos:broader

• Related Term ⇒ skos:related
Summary

• Interdisciplinary character of ontology design
• Ontology design and ontology evaluation
• Problem space vs. Solution space
  – The issue of matching problems to solutions
• Ontology design patterns
  – Ontology building blocks
  – Allow design by re-engineering, specialization and composition
  – Support ontology evaluation

Contribute to the collaborative design effort!

• http://www.ontologydesignpatterns.org
• http://www.neon-project.org
• http://www.w3.org/2001/sw/BestPractices/
Some references

Gangemi, A., A. Catenacci, R. Johnson, and Viinikka J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA (1998).