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
• Ontologies and language
• Ontology design components
• 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, FaCT++, Pellet, SMW, Jena, AllegroGraph, Virtuoso, ...*)
The cultural context of ontologies
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?
  - [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”), 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

In the figure:
- **Input and Output**: The diagram shows the flow of the design process, starting with **ontological-related data** and ending with **ontological project execution**.
- **Design Actions**: These include **design solution**, **ontological project execution**, **collaborative procedure**, and **argumentation session**.
- **Tools and Resources**: Various tools such as Watson, Swoogle, Oyster, etc., are highlighted, along with collaborations like Cicero and Collaborative Protégé.

Additional notes:
- The diagram also mentions platforms like Linking Open Data, Biological ODPs, and ODP-Web.
- Tools that support pattern-based design, evaluation and selection, reengineering, reasoning and querying, evolution, and mapping are indicated.

**Collaborative Ontology Design Components**
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
Logical layers, types of entities, and contexts

- **Ontology**
  - Meta-level Theory (epistemically)
  - Meta-level Theory (semantically)
  - Meta-level Theory (syntactically)
  - First-order Theory $\approx$ TBox (incl. classes, relations)
  - Knowledge Base $\approx$ ABox (incl. individuals, facts)

- **Communities**
- **Formal entities**
- **Information**
- **Meanings**
- **Facts, situations**

Diagram:

- Appendicectomy for Durban's school can be performed by...
- Appendicectomy is a class
- Appendicectomy is a compound word
- An appendicectomy is a surgical removal of the vermiform appendix
- John had an appendicectomy

Notes:
- Appendicectomy is a class
- Appendicectomy is a compound word
- Appendicectomy is a surgical removal of the vermiform appendix
- John had an appendicectomy
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

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

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

The 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

The TimeInterval Content OP locally defines the following ontology elements:

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

**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

- **hasInterval 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:Year, xsd:date:Time, etc.
  - hasIntervalDate page

- **hasInterval start date** (owl:DatatypeProperty)
  The start date of a time interval.
  - hasIntervalStartDate page

- **hasInterval end date** (owl:DatatypeProperty)
  The end date of a time interval.
  - hasIntervalEndDate page

**Submitted by** ValentinaPresuti
**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 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 $p_2$
- $\text{BiochemicalTreatment} \rightarrow (\text{Role} \leftrightarrow \text{Task} \circ \text{Description} \leftrightarrow \text{Situation} \circ \text{Substance} \leftrightarrow \text{Agent} \circ \text{Time-indexedParticipation})$
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” 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

![Diagram showing the relationship between 'wordnet:Document', 'Artefact', and other terms like 'Programme', 'SlideSet', 'Paper', 'Poster', and 'Proceedings'.]
– 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
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 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 ?x ?y WHERE { ?x r :PlayerRole . ?y a :PlayerRole }
  - SELECT ?x ?z WHERE { ?x r :Tune . ?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:integer }

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
  - SELECT ?x ?y ?z ?w ?k
  - WHERE {
    - ?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 }
Appendix:
Other types of ontology design patterns
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 \text{Man}(x) \land \exists y(\text{hasFiancee}(x,y))
  – \text{Man}(\text{John})
  – \text{hasFiancee}(\text{John},\text{Mary})
  – \therefore \text{Yes-Man}(\text{John})

• Automatic subsumption
  – Yes-Man(x) =df \text{Man}(x) \land \exists y(\text{hasFiancee}(x,y))
  – \text{ItalianMan}(x) \Rightarrow \text{Man}(x)
  – \text{hasFrenchFiancee}(x,y) \Rightarrow \text{hasFiancee}(x,y)
  – \therefore ((\text{ItalianMan}(x) \land \exists y(\text{hasFrenchFiancee}(x,y))) \Rightarrow \text{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) ⇔ hasFiancé(y,x)
  – hasFiancee(John,Mary)
  – ∴ hasFiancé(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.
Re-engineering Patterns (RPs): 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.
Ontology-related data: 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.

- **Open tag systems**
  - Flickr, Wikipedia, MySpace, ...

- **Linked Open Data**
  - DBpedia, Microformats, RDFa, etc.

- **Text extractors**
  - Text2Onto, TermExtractor, SST, Frame Detector, ...
Searching and using ontologies, on-the-fly data reengineering

- Watson (RDF search engine)
- Sindice (RDF search engine)
- Yago (a metamodel for dbpedia and wordnet)
- Umbel (a topic ontology for Linked Open Data)
- LMM (a semiotic metamodel for Linked Open Data and lexical resources)
- Freebase (a metamodel and user interface to enriched Linked Open Data)
- OpenLink Data Explorer (a user interface over Linked Open Data)
- GRDDL, RDFa (RDFizers over Web pages and Microformats)
Example of RP: from thesauri to ontologies in SKOS

• KOS $\rightarrow$ skos:ConceptSchema
• Descriptor $\rightarrow$ skos:Concept
• Broader Term $\rightarrow$ skos:broader
• Related Term $\rightarrow$ 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


