Ontology Evaluation and Pattern-based Design

Aldo Gangemi
Semantic Technology Lab, ISTC-CNR, Rome, Italy
aldo.gangemi@cnr.it
@ontolog @ontologysummit
User justifications for exemplary ontologies (from ODP wiki)

- 32 sets of justifications supporting the inclusion of some ontologies in the “Exemplary Ontologies” page of ODP
  - Template designed by Mike Uschold for NeOn ODP
- Analyzed and classified according to the evaluation framework by Gangemi et al. (2006)
  - See also d’Aquín and Gangemi (2011) on “beautiful” ontologies
  - Classes: syntactic and formal **structure**, conceptual **coverage** and **task**, and pragmatic or **social sustainability**
- The most frequent justifications are from the two classes: **task**, and **social sustainability**
Structural justifications

• Reusing foundational ontologies
• Being designed in a principled way
• Being formally rigorous
• Implementing also non-taxonomic relations
• Following strictly an evaluation methodology
• Being modular, or embedded in a modular framework
Conceptual coverage justifications

• Providing important reusable distinctions
• Having a good domain coverage
• Implementing an international standard (e.g., ISO or WHO)
• Providing an organisation to unstructured or poorly structured domains
Conceptual task justifications

- Being oriented at an explicit task
- Having spelled out requirements from scenarios
- Being based on competency questions

Most frequent justifications
Social sustainability justifications

- Being the result of an evolution (many revisions)
- Having wide usage or acceptance
- Having commercial impact
- Being recommended by industry
- Implementing scientific knowledge

*Most frequent justifications*
Pragmatic sustainability justifications

• Having applications built on top of it
• Having successful personal experience in building apps with it
• Designed for efficient query answering
• Maintaining original expressivity of data, and improving or enriching it
• Able to get rid of clunky constructs or to overcome expressivity limitations
• Being well documented
• Solving other technical aspects
Notions of quality from Ontolog Summit discussion

• What ontology to use?
• How to measure/evaluate?
• Any requirements specified?
• Analogy to QA (requirements, design/production, check against requirements)
• Any (successful) revision story?
Notions of quality from Ontology Summit discussion -- my contribution

• What ontology to use?
  – Match requirements to repositories (search from Watson, ODP, recommendations, advanced search from XD NeOn Toolkit plugin)

• How to measure/evaluate?
  – Ex-post: decide your principles and metric (QOOD approach)
  – Ex-ante: follow eXtreme Design methods: CQs, patterns, unit tests

• Any requirement specified?
  – Competency questions-based approach

• Analogy to QA (requirements, design/production, check against requirements)
  – Either QOOD, or eXtremeDesign

• Any (successful) revision story?
  – Integration monitoring from eXtreme Design workflow
Quality measure classification

• STLab research from 2004-5: “A formal framework for ontology evaluation and selection”
• 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*
  – Good practices (*history*)
• Empirical methods for evaluation (measurability)
Examples of structural checking tools

- Graph measures
- Reasoners: HermiT, Pellet, etc.
- LINTs: Pellet, OPPL (custom tests)
  - agghiai-2:pellet-2.2.2 agghiai$ sh pellet.sh lint -v /Users/agghiai/Workspaces/AllPatterns/dul/DUL.owl
  - No RDF lints found.
  - No OWL 2 DL violations found for ontology <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>
  - OWL Lints found for ontology <http://www.ontologydesignpatterns.org/ont/dul/DUL.owl>:
    - [EquivalentAndSubclassAxiomPattern: A named concept appears in equivalent axiom(s) and on the left-hand side of a subclass axiom]
    - [ExistentialExplosionPattern (MaxTreeSize = 10000): Concepts/Individuals are involved in a large some/min/ exact value restrictions tree/loop - maximum recommended number of generated nodes is 10000]
  - and 5 more.

- User rank-based measures (e.g. Open Rating System)
- XD Analyzer
  - rule-based anti-pattern tests: typical sources of errors such as domain intersection in properties
Task-oriented ontology design

• Ontologies must match both domain and task
  – Allow the description of the entities ("domain") whose attributes and relations are concerned because of some purpose
    • 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 topics of documents
  – Serve a purpose ("task")
    • 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.
Evidence that units of expertise are larger than what we have from average linked data, or worse, ontology learning

- “Blinking” effects in reacting to events, in evaluating the actions and theories of the others, in understanding context, in interpreting news and ads, etc.
- Competency questions (Gruninger, 1994) try to convey these units as requirements
  - Which objects take part in a certain event?
  - Which tasks should be executed in order to achieve a certain goal?
  - What’s the function of that artifact?
  - Does this behaviour conform to a certain rule?
  - What norms are applicable to a certain case?
  - What norm is superordinated among these ones?
- Sometimes exception conditions should be added
- Task-based ontology evaluation can be performed with unit tests against ontologies trying to satisfy competency questions
Ontology Design Patterns

An ontology design pattern is a reusable successful solution to a recurrent modeling problem

---->
Gruninger on Competency Questions
Alexander on Design Patterns
Clark on Knowledge Patterns
W3C SWBPD WG
ODP movement, cf. ontologydesignpatterns.org
Pattern-based design
aka eXtreme Design (XD)

• Pattern-based ontology design is the activity of searching, selecting, composing different patterns, and performing testing and integration of pair-design sessions (see later)
  – Logical, Content (or Knowledge), Reasoning, Architectural, Naming, Correspondence, Reengineering
  – Common framework to understand modeling choices (the “solution space”) wrt task- and domain-oriented requirements (the “problem space”)
  – http://www.ontologydesignpatterns.org
Catalogues of ODP 1/2

- ODPs are collected and described in catalogues and comply to a common presentation template
- The ontologydesignpatterns.org initiative maintains a repository of ODPs and a semantic wiki for their description, discussion, evaluation, certification, etc.
Catalogues of ODP 2/2
Types of ODPs
Logical vs. Knowledge patterns

- A Logical ODP describes a formal expression that can be exemplified, morphed, instantiated, and expressed in order to solve a domain modelling problem.
- \( \text{owl:Class:}_\text{x} \text{ rdfs:subClassOf owl:Restriction:}_\text{y} \)
- Inflammation \( \text{ rdfs:subClassOf (localizedIn some BodyPart)} \)
- Colitis \( \text{ rdfs:subClassOf (localizedIn some Colon)} \)
- John’s_colitis \( \text{ isLocalizedIn John’s_colon} \)
- “John’s colon is inflammated”, “John has got colitis”, “Colitis is the inflammation of colon”

![Diagram showing the logical vs. knowledge patterns]

**Abstraction**
Sample Knowledge Patterns
aka Content Patterns
Situation

- A general vocabulary for n-ary relations
- *Situation* abstracts from reified n-ary relations, by defining a top-level relation for all binary projections of the n-ary relation
- A way to conceive a state of affairs, a set of things, a fact
- All time indexed (and place indexed) patterns are (in principle) specializations of *Situation*
The *Observation* knowledge pattern

Continuity between logical and knowledge patterns

- Signature introduction ->
- Signature morphism ->
- Adaptation ->
- Evaluation against competency questions
Aquatic Resource Observation
Time Interval

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: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**: ValentinaPresutti  
**Name**: time interval  
**Also Known As**: interval  
**Intent**: To represent time intervals.  
**Domains**: Time  
**Competency**: What is the end time of this interval?, What is the starting time of this interval?, What is the date of this time interval?  
**Questions**:  
**Reusable OWL**: [http://www.ontologydesignpatterns.org/cp/owl/timeinterval.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.  

**Known Uses**  
- Web  
- References  
- Other  
- References  

**files**  

Extracted From  
Reengineered From  
Has Components  
Specialization Of  
Related CPs
Parthood

This also uses transitivity reasoning logical pattern

Time-indexed Participation

This also uses N-ary logical pattern
Crime
Anti-patterns (1/2)

- Partonomies or subject classifications as subsumption hierarchies
  - *City subClassOf Country
  - City subClassOf (partOf some Country)
  - *City subClassOf Geography
  - City broader Geography (e.g. in SKOS)

- Linguistic disjunction as class disjointness
  - Dead or alive
    - *Dead or Alive
    - Dead disjointWith Alive

- Linguistic conjunction as class disjunction
  - Pen and paper
    - *Pen and Paper
    - Pen or Paper | Collection subClassOf (hasMember some Paper ; some Pen)
Anti-patterns (2/2)

- Causality as entailment
  - Kaupthing bank behavior caused Iceland crisis
    *KaupthingBankBehavior subClassOf IcelandCrisis
    *KaupthingBankBehavior isCauseOf IcelandCrisis

- Expressions as instances of the class representing their meaning
  - *dog(word) rdf:type Dog
  - dog(word) expresses Dog (with punning)

- Multiple domains or ranges of properties as intersection
  - *hasInflammation rdfs:domain Epithelium ; Endothelium
  - hasInflammation rdfs:domain (Epithelium or Endothelium)
eXtreme Design
Tool support: matching and selection of patterns are perceived to be the most difficult tasks
Method and tool support

• eXtreme Design (XD)
  – a method for developing ontologies with Content Patterns

• XD tool
  – a tool that supports XD method
  – released as both an Eclipse plugin and a NeOn Toolkit plugin
  – We will use it with the NeOn toolkit

http://neon-toolkit.org/wiki/Download/2.3.2
Method 1 – Unit tests

• Verifying that the ontology supports retrieval of information corresponding to the CQs
• Write one unit test corresponding to each CQ of the ontology (module)
  – Unit test? – SPARQL query
• What kind of errors?
  – Mainly missing elements
  – Violation of modeling best practices
Method 2 – Performing inferences

• Verifying that the ontology supports the reasoning tasks required

• Create one test case for each inference to be made (according to the requirements)
  – Test case? – Set of “input” facts that should produce the desired output of inference

• What kind of errors?
  – Unfulfilled reasoning requirements
  – Unexpected side-effects
  – Inconsistencies
Method 3 – Performing “stress tests”

• Verify that the ontology enforces the contextual statements and is robust against unexpected or erroneous data

• Create one test case for each contextual statement and “boundary value” to test
  – Test case? – Data that should produce some errors, e.g. inconsistency, in the ontology

• What kind of errors?
  – Unfulfilled or missing contextual statements, i.e. missing axioms
  – Implicit constraints that should be made explicit
  – Unexpected side-effects
Other ways of testing?

• Inspection
  – Checking coverage
    • terminology
    • axioms
  – Checking against best practices, e.g. ODPs

• “Peer review”

• …
Browse, search, and get Content ODPs

Specialize, compose, annotate ODPs and ontologies

Analyze your ontology against good practices and patterns
Experimental evidence (I)

- Content patterns improve the quality of ontologies
  - Experiments with master and PhD students
  - Quality measured in terms of
    - task-coverage
    - error-freedom
    - subjective perception of smooth and good design

- Published at KCAP09
Experimental evidence (II)

• eXtreme Design method further improves quality and also improves coverage of the proposed requirements
  – Experiment with 7 designer pairs (PhD students)
• Published at EKAW2010
Experimental evidence (III)

- ODP-based ontology learning improves results
- Ontologies are better in terms of cohesion, consistency, functional quality, etc.
- Experiment with OntoCase applied to Text2Onto ontology learning
- Published at ISWC2009
Training material on ODP

- http://ontologydesignpatterns.org/wiki/Training:Main
- Latest material on eXtreme Design from 2011 PhD course
References