Pattern-based ontology design

Aldo Gangemi
Valentina Presutti

Semantic Technology Lab
ISTC-CNR, Roma
aldo.gangemi@cnr.it
valentina.presutti@istc.cnr.it
Outline

• Designing Computational Ontologies
• Ontology Design Patterns
• ontologydesignpatterns.org initiative
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-SelfExplaination
  • 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
Ontology-related data

- Informal vs. formal
- Text corpora
- Folksonomies (tag sets, directories, topic trees, subject indexes, infoboxes)
- Lexica (dictionaries, wordnets, terminologies, nomenclatures)
- Knowledge organization systems (thesauri, classification schemes)
- Frames, semantic networks
- DB schemas
- Linked Open Data datasets
- (Computational) ontologies
A lot of data in the web “suq”

- Mash-ups
- Linked open data
- Wikipedia, DBpedia, Freebase, etc.
- Triplify, GRDDL, RDFa, SKOS, SIOC, etc.
- Corpora, terminologies, lexica, thesauri, “KOS”, frames, ontologies
Standard languages help

- Transform all in RDF, or even OWL
- Cf. Triplify initiative
- Dataset extracted from heterogeneous sources, and triplified
- Relations are added in direct, naïve ways: Linked Open Data
- Semantics depends on intended task of data and relations used for linking
- Then search/visualize RDF data, or make integrating applications
Search results for term “desire”, found about 8.53 thousand

Is This Desire? (RDF)
[2008-12-10 - 116 triples in 19.4 kb]
http://dbpedia.org/resource/Is_This_Desire%3F (Search) (Cached) (Ontologies)

desire (RDF)
[2008-11-11 - 7 triples in 1 kb]
http://wordnet.rikexplorer.com/id/synset-desire-verb-3 (Search) (Cached) (Ontologies)

is (RDF)
[2008-11-11 - 9 triples in 1.3 kb]
http://wordnet.rikexplorer.com/id/word-desire (Search) (Cached) (Ontologies)

desire (RDF)
[2008-11-11 - 8 triples in 1.3 kb]
http://wordnet.rikexplorer.com/id/wordsense-desire-verb-3 (Search) (Cached) (Ontologies)

is (RDF)
[2008-11-11 - 8 triples in 1.3 kb]
http://wordnet.rikexplorer.com/id/wordsense-desire-verb-2 (Search) (Cached) (Ontologies)

is (RDF)
[2008-11-11 - 7 triples in 1.1 kb]
http://wordnet.rikexplorer.com/id/wordsense-desire-noun-1 (Search) (Cached) (Ontologies)

is (RDF)
[2008-11-11 - 9 triples in 1.4 kb]
http://wordnet.rikexplorer.com/id/wordsense-desire-noun-2 (Search) (Cached) (Ontologies)
Integrated knowledge search: DBpedia
Integrated knowledge search: Freebase
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, 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
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) 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)
• ... 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 as Ontology Design Patterns, which help reducing arbitrariness without asking for sophisticated skills ...
• ... provided that tools are built for any user :)

Solutions?
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, SMW, 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?
- What, 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
Designing Computational Ontologies

Ontology Design Patterns

ontologydesignpatterns.org initiative
Ontology Design Pattern

• An ontology design pattern is a successful reusable 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, Reengineering, Content
  - Common framework to understand modeling choices (the “solution space”) wrt task- and domain-oriented requirements (the “problem space”)
  - http://www.ontologydesignpatterns.org
Types of Ontology Design Patterns (OPs)

- We also distinguish between ontological resources that are not OPs and Ontology Design Anti-Patterns (AntiOP)
Examples of Presentation OPs

- 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 Reasoning OPs

- Precise
  - Classification
  - Subsumption
  - Inheritance
  - Materialization
  - De-anonymizing
  - ...

- Approximate
  - Approximate classification
  - Similarity induction
  - Taxonomy induction
  - Relevance detection
  - Latent semantic indexing
  - Automatic alignment
  - ...

*or some workflow of them, cf. TBC*
Example of Schema Reengineering
OP: kos2skosABox

\[
\begin{align*}
\text{KOS} & \mapsto \text{skos:Concept} \tag{2.1} \\
\text{Descriptor} & \mapsto \text{skos:Concept} \tag{2.2} \\
\text{Broader Term} & \mapsto \text{skos:broader} \tag{2.3} \\
\text{Related Term} & \mapsto \text{skos:related} \tag{2.4}
\end{align*}
\]
Example of Mapping OPs

- Also called “correspondence patterns” in [16]
  - equivalent to, (not equivalent to)
    - foaf:Agent ≡ wn16:Agent-3
  - contained in, (not contained in)
    - foaf:Person ⊆ geo:SpatialThing
  - overlap with
    - foaf:Person ∩ dul:Person
  - disjoint with
    - (dul:PhysicalPerson ∩ dul:SocialPerson) = ∅
  - logically heterogeneous mapping
    - dul:PhysicalPerson (owl:Class) ≈ p1:PhysicalPersonRole (owl:Individual)
- We also consider an additional semantic relation, cloned from
- ontology element oe₁ in one ontology is the clone of an ontology element oe₂ in another ontology
Example of Logical Macro

- Logical macros provide a shortcut to model a recurrent intuitive logical expression

Example:
- the macro: $\forall R. C$ [7] colloquially means "every R must be a C"
- formally: $\exists R. T \sqcap \forall R. C$

in OWL:
- the combination of an owl:allValuesFrom restriction with an owl:someValuesFrom restriction.
Example of Transformation pattern: N-ary relation (1/2)
Example of Transformation pattern: N-ary relation (2/2)

But beware of identification constraints! [15]
Content Ontology Design Patterns

Some theory
Reusable ontologies?

- How many cases of reusability?
- What kind of ontologies are mostly reusable?
- How many ontologies have been actually specialized in more than one domain?
- How many studies in comparing the cost of reusing vs. developing from scratch?
- How many studies in evaluating/facilitating reusability?

- Let’s face it: reusing, when applied, is an art, not a communicable/manageable know-how
- Started with W3C SWBPD: OWL modelling best practices, semantic SE patterns, techniques to vocabulary porting and migration to the SW
- Then in EU NeOn project: ontology design for networked and contextualized ontologies: Watson, ODP Portal, Modularization plugin, Pattern-based design plugin
From the lessons learnt ...

- We envision small ontologies with explicit documentation of design rationales, and best reengineering practices
- components supported by specific functionalities
  - selection, matching, composition, etc.
- implemented in repositories, registries, catalogues, open discussion and evaluation forums, and in new-generation ontology design tools
  - ontologydesignpattern.org
  - ODP and Watson APIs
  - NeOn ODP Plugin
  - etc.
Content OPs (CPs) 1/2

- CPs encode conceptual, rather than logical design patterns.
- Logical OPs 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.
- CPs are instantiations of Logical OPs (or of compositions of Logical OPs), 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 OPs (CPs) 2/2

• CPs are instantiations of Logical OPs (or of compositions of Logical OPs), 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
• 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 [11]
A pattern is a theory template. It denotes a structure that is invariant under signature transformation (morphism). Pattern validity in an application is then left to a subjective decision.

E.g. the axiom:

\[ \forall c((\text{consumer}(c) \land \exists p(\text{producer}(p) \land \text{connects}(c,p))) \rightarrow \text{supplied}(c)) \]

via signature morphism becomes e.g. in an application:

\[ \forall c((\text{light}(c) \land \exists p(\text{battery}(p) \land \text{connects}(c,p))) \rightarrow \text{powered}(c)) \]

But if a pattern is just an untyped structure, there are no ways to distinguish a Logical OP vs. a CP
CPs vs. Logical OPs

- $\forall c((\text{consumer}(c) \land \exists p(\text{producer}(p) \land \text{connects}(c,p))) \rightarrow \text{supplied}(c))$

- **SubClassOf**
  - $((\text{intersectionOf}
  \begin{align*}
  \text{Consumer} \\
  (\text{restriction}(\text{connects}\ \text{someValuesFrom}(\text{Producer})))
  \end{align*}
  \text{Supplied})$

- $\forall c((\psi(c) \land \exists p(\psi(p) \land \rho(c,p))) \rightarrow \chi(c))$

- **SubClassOf**
  - $((\text{intersectionOf}
  \begin{align*}
  \text{owl:Class}:\psi \\
  (\text{restriction}(\text{owl:ObjectProperty}:\rho\ \text{someValuesFrom}(\text{owl:Class}:\psi)))
  \end{align*}
  \text{owl:Class}:\chi)$

- In OWL, this is a GCI (General Concept Inclusion) axiom. Not a typical LP

**CPs vs. Logical OPs**

- CP: specific (non-logical) vocabulary
- Logical OP: no specific vocabulary
Formal characteristics of OWL CPs

• Mostly graphs of classes and properties, self-connected through axioms (subClassOf, equivalentClass, domain, range, disjointFrom)
  • ObjectProperty(component domain(System))
• Usually applied through downward subsumption of at least one element
  • “being a part of something at some time”
  • “being a component of a system at some time”
  • “being a section in a law at some time”
• Or through composition
  • “being a section in a law at some time” $\otimes$ “being expressed in a legal text”
• Usually there is an underlying n-ary relation (sometimes polymorphic)
  • component(s,e,t) $\rightarrow$ System(s) $\land$ Entity(e) $\land$ Time(t)
  • $? \text{component(s,e,t,...)} \rightarrow$ System(s) $\land$ Entity(e) $\land$ Time(t) $\land$ Function(...) ...

NeOn
Pragmatic characteristics of CPs

- Domain-dependent
  - Expressed with a domain-specific (non-logical) vocabulary
- Requirement-covering
  - Solve domain modelling 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, Plan, Legal Norm, Legal Fact, Sales Order, Research Topic, Legal Contract, Inflammation, Medical Guideline, Gene Ontology Top, Situation, TimeInterval, etc.
# Generic ontology requirements (GCQ)

<table>
<thead>
<tr>
<th>Generic Competency Questions</th>
<th>Specific Modelling Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Who</strong> does <strong>what, when and where?</strong></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</strong> frame 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 <strong>tasks</strong> 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 <strong>happen</strong>?</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 <strong>is about</strong>? How is it <strong>realized</strong>?</td>
<td>Information and content modelling, computational models, subject directories</td>
</tr>
<tr>
<td>What <strong>argumentation model</strong> are you adopting for negotiating an <strong>agreement</strong>?</td>
<td>Cooperation systems</td>
</tr>
<tr>
<td>What’s the <strong>degree of confidence</strong> that you give to this axiom?</td>
<td>Ontology engineering tools</td>
</tr>
</tbody>
</table>
• A catalogue of CPs
• http://www.ontologydesignpatterns.org (odp-web)
• catalogue entry

• Annotation properties:
• http://www.ontologydesignpatterns.org/schemas/cpannotatonschema.owl
• annotation of OWL implementation of CPs
Example 1: AgentRole

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.

<table>
<thead>
<tr>
<th>AgentRole</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submitted by</td>
</tr>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Also Known As</td>
</tr>
<tr>
<td>Intent</td>
</tr>
<tr>
<td>Domains</td>
</tr>
<tr>
<td>Competency</td>
</tr>
<tr>
<td>Questions</td>
</tr>
<tr>
<td>Reusable OWL</td>
</tr>
</tbody>
</table>
Agent Role Instantiation

![Diagram showing relationships between roles and agents.]

- Aldo_Gangemi
  - or:hasRole Saxophonist
  - or:hasRole SeniorResearcher
  - or:hasRole Father
  - or:Agent

Saxophonist
SeniorResearcher
Father
ar:Agent

or:Role

rdf:type
Example 2: Time Interval

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

 hasIntervalDate (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:dateTime etc.

 hasIntervalDate page

 hasIntervalStartDate (owl:DatatypeProperty)

The start date of a time interval.

 hasIntervalStartDate page

 hasIntervalEndDate (owl:DatatypeProperty)

The end date of a time interval.

 hasIntervalEndDate page

TimeInterval

Submitted by: Valentina Prestiti
Name: time interval
Also Known As:

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: Date of this time interval?

Reusable OWL: http://www.ontologydesignpatterns.org/ocl/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

Examples (OWL) http://www.ontologydesignpatterns.org/ocl/examples/timeinterval/january2008.owl

Extracted From
Reengineered From
Has
Components
Specialization Of
Related CPs
Example 3: Part

This also uses transitivity reasoning pattern

Example 4: Time-indexed Participation

This also uses N-ary logical pattern
Example 5: Crime
Example 6: Aquatic Resource Observation
(Re)use situations:
matching CPs covering against local problems
Representing local problems (LCQ)

- Local problems can be expressed in different ways:
  - use cases, scenarios, user requirements, local competency questions (cqs), etc.
- Following [11] all can be transformed to local “cqs”.
  - Red Hot Chili Peppers recorded the Stadium Arcadium album during 2005
  - When did Red Hot Chili Peppers record the Stadium Arcadium album?
  - Which albums did Red Hot Chili Peppers record during 2005?
  - ...
- Local “cqs” are not usually at the same level of generality as the cqs of CPs
  - e.g., they may contain reference to instance element e.g. Stadium Arcadium
  - we need to abstract them
  - When did a certain band record a certain album?
  - Which albums did a certain band record during a certain time period?
  - ...

•

ST•Lab
Semantic Technology Laboratory

NeOn
What we mean by **matching cqs to CPs**

- What do we mean by matching a cq to CPs?
  - To compare the local cqs to the cqs covered by a CP in order to evaluate the CP suitability for solving the local problems
  - There is not yet automatic support for this task, hence it is performed as a human task
  - Ongoing work on automatic support for CP selection starting from local cqs
    - parsing of requirements and extraction of cqs
    - formalization of cqs
    - NLP support to match cqs terminology to CP lexicalizations
    - case-based reasoning [13]
    - ontology matching
    - ...
A content pattern CP₂ specializes CP₁ if at least one ontology element of CP₂ is subsumed by an ontology element of CP₁

i.e., either by `rdfs:subClassOf` or `rdfs:subPropertyOf`
The resulting ontology is composed of the union of the ontology elements and axioms from the two CPs, plus the axioms (e.g. disjointness, equivalence, etc.) that are added in order to link the CPs.
Where do CPs come from?

• Content ontology design patterns (CPs) come from the experience of ontology engineers in modeling foundational, core, or domain ontologies

• There are four ways of creating CPs, which can be summarized as follows:
  • Reengineering from patterns expressed in other data models
  • Data model patterns, Lexical Frames, Workflow patterns, Knowledge discovery patterns, etc.
  • Specialization/Generalization/Composition of other CPs
  • Extraction from reference ontologies (by cloning)
  • Creation by combining extraction, specialization, generalization, composition, and expansion
eXtreme ontology Design (XD)

- Inspired by eXtreme Programming basic rules
  - e.g., pair programming, test-oriented, continued integration, etc.
- 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
- As an example, we apply an XD iteration with CPs
Sentence: Charlie Parker is the alto sax player on Lover Man, Dial, 1946

- Charlie Parker (person)
- the alto sax player (player role)
- on Lover Man (tune)
- Dial (publisher)
- 1946 (recording year)

CQs
- what persons 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 }

Alternative abstractions do exist!
Sample XD iteration 2/3

- 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), ...
Sample XD iteration 3/3

- 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 }
Experiments: first results

- During a four-day course for PhD students
  - Most have never constructed an ontology, or only a small example
  - Mostly taxonomies or lightweight ontologies
  - Most subjects familiar with some modeling language (like ER or UML), but only a few have tried OWL
- Background questionnaires, ontology design exercises (end of every day), subjective feedback questionnaire after exercise
  - first two days no patterns, second two days with patterns
- Some preliminary results based on subjective feedback questionnaires only
  - Main difficulties: mapping from the problem to the patterns, pattern composition
  - Most found the patterns useful and many perceived that they introduced some solution they did not think of themselves
  - Most perceived the second exercise as the easiest to solve, and the fourth as the most successfully modeled
  - The last day we have also got pattern proposals
Ongoing and future work

✓ Bootstrapping and improving functionalities in the ODP portal
✓ ODP APIs
  • Building the NeOn Toolkit ODP plugin
  • Continue with experimentation
  • Use of CBR for pattern-based automatic ontology construction
  • Join the ODP community! http://www.ontologydesignpatterns.org
Outline

• Designing Computational Ontologies
• Ontology Design Patterns
• ontologydesignpatterns.org initiative
ontologysdesignpatterns.org (ODP)  
a semantic web portal

Evaluation WikiFlow  
a Semantic MediaWiki extension
ontologydesignpatterns.org (ODP)

- A semantic web portal about OPs (Logical, Content, Presentation ...)
- currently supports CPs
- best practices for ontology design and ontology engineering
- evaluation, training and repository of reusable OWL ontologies
ODP areas

- Community: share experience, collects modeling issues and domains
- Proposals: collects ProposedCP
  - Submissions by form and import facility
- Reviews: guidelines and rationales.
- Open reviews and Quality Committee reviews
- Catalogue: collects CertifiedCP (complete, reusable, well-done)
- Training: tutorials, exercises
- Feedbacks: from users' feedback to development tasks (Editorial Board)
ODP types of user

- Anonym User
  - Asks for an ODP account
  - Improve content
  - Give feedbacks about ODP
  - Answer feedbacks
  - Manage development tasks
  - Create accounts
  - Manage User rights

- ODP User
  - Navigate ODP (read-only)
  - Define new Domains
  - Submit Modeling issues
  - Contribute to discussion
  - Submit OP proposals
  - Make open reviews
  - Make QC reviews
  - Certify patterns
  - Assign review

- EB Member
  - <<requires>>
  - <<extends>>

- QC Member
  - <<requires>>

- Administrator
ODP content

- Semantic representation
  - Semantic MediaWiki (SMW) and Semantic Forms (SF) + exts
- Each pattern is described by:
  - diagram
  - annotations (user, name, intent, domain(s), competency questions, known uses, consequences, OWL file, related CPs and Ontologies
  - elements (list, description)
  - scenarios
  - reviews
Evaluation WikiFlow

- Extends MW, SMW and SF extension
- Evaluation tab

**Features:**
- **configuration**
- **functionality**
Evaluation WikiFlow: configuration

- Activation tab
  - categories to evaluate.
  - e.g. currently ODP activates it for the ProposedCP category.
- Review schema(s) customization
  - different review schemas can be defined
- Category/review association
  - categories to evaluate with review schemas
    - E.g. ProposedLP and ProposedCP have different review schemas
- User rights configuration
  - view, ask for, assign, make, certify
  - E.g. QualityCommity members make reviews, while every ODPUser can request reviews
Evaluation WikiFlow: functionality

- **ask for review**: +WaitingForReview
- **assign review**: +AssignedReview
- **make review**: -WaitingForReview, +AssignedReview.
- **certify**: +Certified, freezed; new lifecycle.
- **semantic report of evaluation history**
  - aim: to analyze rationales behind evaluation of design patterns

This pattern has been certified.
Related submission, with evaluation history, can be found [here](#).
Evaluation WikiFlow: software

- alpha version as open source software
- can be downloaded from the MediaWiki wiki site
Conclusion and future work

- Ontologydesignpatterns.org and Evaluation WikiFlow
  - A community-based web portal (training, discussion and repository)
  - A domain-independent extension for SMW and SF
- ODP ongoing and planned work includes
  - new types of ontology design patterns
    - e.g Logical, Reengineering
      - 1st f2f editorial board meeting on Feb 23rd
  - a search service based on Watson
    - http://watson.kmi.open.ac.uk
  - the ODP repository APIs
  - OWL/RDF export service
  - an open rating system for open reviews (based on NeOn ORS)
  - statistical monitoring of CP downloads to be used as a dimension of user-based evaluation of CPs and ODP usage


References (2)

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