Ontology Design Patterns

Eva Blomqvist
eva.blomqvist@liu.se
Department of Computer and Information Science (IDA)
Linköpings universitet
Sweden

Slides partly by Valentina Presutti, STLab, ISTC-CNR, Italy
Outline

- Motivation for ontology design patterns
- Logical ODPs
- Content ODPs
- Other ODPs and best practices
- Exercise
How does city relate to country?

Relation types:
- All Types
- Inheritance
- Disjointness
- Named Relations

Strategy:
- Use one ontology
- Use more ontologies
- Find first relation
- Find all relations
- Use inheritance depth

Examples:
- River vs. waterway
- Cocaine vs. narcotic
- Water vs. solid
- Branch vs. Tree
- Coal vs. Industry
- Fish vs. Lobster
- Cholesterol vs. OrganicChemical
- Apple vs. Meat

KMi

October 4, 2011
What we can do with OWL

- ... (maybe) we can check the consistency, classify, and query our knowledge base
- ... but, remember the Scarlet example
  - City subClassOf Country
- Logical consistency is not the main problem
  - e.g. owl:sameAs can be wrongly used and still we have consistency
- Why is OWL not enough?
  - 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
Solutions?

- OWL is not enough for building a good ontology, and we cannot ask all web users neither to learn logic, or to study ontology design.
- Reusable solutions are here through Ontology Design Patterns, which help reducing arbitrariness without asking for sophisticated skills ...
- ... provided that tools are built for any user ☺️
Ontology Design Patterns

An ontology design pattern is a reusable successful solution to a recurrent modeling problem.
Logical Ontology Design Patterns
Logical ODPs

- **Definition**
  - A Logical ODP is a formal expression, whose only parts are expressions from a logical vocabulary e.g., OWL, that solves a problem of expressivity

- Logical ODPs are independent from a specific domain of interest
  - i.e. they are content-independent
Logical ODPs

- A Logical ODP describes a formal expression that can be exemplified, morphed, and instantiated in order to solve a domain modeling issue.
- `owl:Class:_:x rdfs:subClassOf owl:Restriction:_:y`
- `Inflammation rdfs:subClassOf (localizedIn some BodyPart)`
- `Colitis rdfs:subClassOf (localizedIn some Colon)`
- `John’s_colitis localizedIn John’s_colon`
Logical macros and Transformation ODPs

- Logical macros provide a shortcut to model a recurrent intuitive logical expression
  - Example:
    - Formally: R some Thing and R only C
    - Things that R, R at least one thing that is only C
    - Carnivore animals eat only animals + Carnivore animals eat some (at least one) animals

- Transformation ODPs
  - Example: Transforming a more expressive construct into something that can be expressed in OWL
Transformation ODPs
Example: N-ary relation

- Chad Smith was the drum player of Red Hot Chili Peppers when they recorded their album Stadium Arcadium from September 2004 to December 2005.
- A person plays a certain role in a band during an album recording, taking place during a certain time interval

N-ary relation:
- PlaySituation(Person, MusicianRole, Band, Album, TimeInterval)

How can we express this in OWL with only binary relations?
Transformation ODPs
Example: N-ary relation

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden

October 4, 2011
Transformation ODPs
Example: N-ary relation

ClassArgument1

ClassArgument3

ClassArgument4

ClassArgument2

NaryRelation

relation1

relation3

relation4

relation2
Transformation ODPs
Example: Transitive Reduction

- I want to represent that a car is composed of several parts
  - part of – transitive property
- I also want to represent that each part can have “direct” components
  - e.g. the turbine is a component of the engine
- The turbine is a component of the engine, hence it is part of the car, but not its “direct” component
Transformation ODPs
Example: Direct components in a car
Transformation ODPs
Example: Transitive reduction

aProperty rdf:type owl:TransitiveProperty
anotherProperty rdfs:subPropertyOf aProperty

anotherProperty does not inherit transitivity,

but:

does not inherit transitivity,

but:

entity1 anotherProperty entity2
implies
entity1 aProperty entity2
Content ODPs
Content ODPs (CPs)

- 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

- 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
Catalogues of CPs

- Content ODPs are collected and described in catalogues and comply to a common presentation template.
- The [ontologydesignpatterns.org](http://ontologydesignpatterns.org) initiative maintains a repository of CPs and a semantic wiki for their description, discussion, evaluation, certification, etc.
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, Plan, Legal Norm, Legal Fact, Sales Order, Research Topic, Legal Contract, Inflammation, Medical Guideline, Gene Ontology Top, Situation, TimeInterval, etc.
Examples of CPs

owl:TransitiveProperty

owl:Thing

isPartOf

Band

isMemberOf

Person

narrower

Concept

owl:TransitiveProperty

broader

rdfs:domain

rdfs:range

rdfs:domain

rdfs:range

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden
October 4, 2011
Examples of CPs

Content Pattern
(generic TBox)

Content Pattern
(specific TBox)

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden
October 4, 2011
Sample Specialization

- A content pattern CP$_2$ specializes CP$_1$ if at least one ontology element of CP$_2$ is subsumed by an ontology element of CP$_1$
  - i.e., either by rdfs:subClassOf or rdfs:subPropertyOf
Composition

- The composition operation relates two CPs and results into a new ontology.

- 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.

- The composition of \( CP_1 \) and \( CP_2 \) consists of creating a semantic association between \( CP_1 \) and \( CP_2 \) by adding at least one new axiom, which involves ontology elements from both \( CP_1 \) and \( CP_2 \).

- Typically, also new elements (“expansion”) are added when composing.
General Content ODPs

- Roles of objects
- Classification
- Part-whole relationships
- Membership
- Information and its realization
- Sequences
- Topics
- Time
- Places
- Moving
- Plans
- Events
- Descriptions and Situations

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden
October 4, 2011
Example: Roles of objects

- Objects can play different roles in different situations
- Depending on the constraints given by the requirements, modeling of objects and their roles can be addressed differently
- Do we want to represent properties of roles?
- Do we want to classify objects based on their roles?
- Do we want to assert facts about roles?
Roles of objects

- A beer mug used as vase
- Books used as table’s legs
- A sax player (person)
- A song writer (person)
1st ODP: Roles as classes

Diagram showing the relationships between classes:

- Person
  - SongWriter
  - Mug
  - Vase
  - Books
  - TableLeg
  - SaxPlayer
1\textsuperscript{st} ODP: Roles as classes

- An object and its roles are related through the rdf:type property
- rdf:type relations can be either asserted or inferred through classification
- In order to automatically classify individuals in a certain class the ontology has to define appropriate axioms
1st ODP: Roles as classes

- **Consequences**
  - Low expressivity
  - Roles are described at TBox level
  - Class taxonomy is bigger - a class for each role
  - Class taxonomy is entangled - multi-typing
  - ABox is smaller – same individual, several (role) types
  - Automatic classification of individuals through rdfs:subClassOf inheritance – with proper axioms
  - Roles cannot be indexed in terms of space and time
  - Facts about roles cannot be expressed e.g. “Roles in UniBo can be student, professor, researcher”, “Valentina is teacher for KMDM course”

- **Queries:** ?x a SongWriter

- **General CQs**
  - What things have a certain (role) type?
2\textsuperscript{nd} ODP: Roles as individuals

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden

October 4, 2011
2\textsuperscript{nd} ODP: Roles as individuals

- An object and its roles are related through domain-specific relations
- Relations between an object and its roles have to be asserted
- Automatic inference of relations between an object and its roles can be obtained through property subsumption

\[\text{Person} \rightarrow \text{Role} \]
\[\text{Person} \rightarrow \text{song_writer} \]
\[\text{Person} \rightarrow \text{father} \]
**2nd ODP: Roles as individuals**

- **Consequences**
  - More expressive
  - Roles are described at ABox level
  - Class taxonomy is smaller – roles are individuals
  - Abox is bigger
  - Facts on roles can be asserted
  - Roles can be indexed in terms of time and space - through n-ary relations
  - N-ary relations are needed for relating an object to its role with respect to some other object e.g. Valentina is teacher for KMDM course
    - kmdm_teacher involvesPerson Valentina
    - kmdm_teacher involvesRole teacher
    - kmdm_teacher involvesCourse KMDM
    - Valentina hasRole teacher
  - Roles do not type objects, no automatic classification of objects
  - Queries: ?x hasRole ?y ; ?x a Role

- **General CQs**
  - What roles does an object have? What things have a certain role?

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden

October 4, 2011
3rd ODP: Roles as properties
3rd ODP: Roles as properties

- The semantics of “having a role” is embedded in the name of a property
- Objects are not explicitly related to their roles, they are related to other things through a property expressing an action they perform, a role they play
- Most common pattern in the web of data for modeling roles
3\textsuperscript{rd} ODP: Roles as properties

- Consequences
  - Smaller taxonomy of classes
  - Bigger taxonomy of properties – a property for each role
  - Simpler graph of data – one triple for “Valentina is teacher for KMDM course”
    - Valentina teaches KMDM
  - Roles cannot be indexed in terms of space and time
  - Semantics of roles is implicit (embedded in a property name)
  - Facts about roles cannot be expressed
  - Queries: ?x teaches ?y

- General CQs
  - Who did something?
ODPs for Roles of objects - Summary

- The three solutions differ in expressivity, simplicity, and CQs
  - Simplest is roles as properties
  - Most expressive is roles as individuals
  - Least expressive is roles as classes
- Each of them has pros and cons
- The choice depends on requirements
- What about combining them?
Combining roles as instances with roles as classes

- A class Role
- A class for each Role e.g. SaxPlayer
- A property restriction on classes representing roles, for automatic classification
...and add roles as properties

- Note the restriction on property writerOf
Indexing roles in terms of time and space

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden

October 4, 2011
Indexing roles in terms of time and space

Department of Computer and Information Science (IDA)
Linköpings universitet, Sweden

October 4, 2011
Content ODPs for roles of objects

- The general pattern is called “classification”
- Object-Role and Agent-Role
  - OWL pattern representing roles as individuals
  - http://ontologydesignpatterns.org/wiki/Submissions:AgentRole
- Time-indexed person role-pattern
  - http://ontologydesignpatterns.org/wiki/Submissions:Time_indexed_person_role
- Time-place-indexed object-role
  - N-ary relation representing an objects, the roles it plays at a certain date in a certain place
  - http://www.ontologydesignpatterns.org/cp/owl/dul/timeplaceindexedobjectrole.owl
Other ODPs?

- Lexico-syntactic ODPs
  - What patterns in text correspond to logical constructs, e.g. in OWL?

- Presentation ODPs
  - Templates for annotation and documentation
  - Naming conventions
    - Singular for class names
    - Readable property names
    - URI conventions

- Correspondence ODPs
  - Reengineering ODPs for transforming DBs, XML etc into OWL
  - Alignment patterns, i.e. correspondence between alternative modeling choices

ODPs vs. “best practices”?

- ODPs ideally represent “best practices”
- Some things that are called “best practices” in other areas are collected under the ODP umbrella
  - C.f. naming conventions
- What about automatically generated ODPs? Top-down vs. bottom-up pattern engineering…
  - EKPs
  - Linguistic Frames
Exercise – Exploring the ODP portal

- ODP selection
- Given a set of requirements find Content ODPs that could be used as components in a solution
