Semantic Web Technologies

Lecture 5: Ontology engineering methodologies

Maria Keet

email: keet -AT- inf.unibz.it
home: http://www.meteck.org
blog:
http://keet.wordpress.com/category/computer-science/72010-semwebtech/

KRDB Research Center
Free University of Bozen-Bolzano, Italy

30 November 2009
Outline

Methodologies overview

A collection of parameters
  Purposes of the ontologies
  Reusing ontologies
  Bottom-up development of ontologies
  Representation languages and reasoning services

Methods
  Guidance for modelling: OntoClean
  Debugging ontologies

Methodologies and tools
Outline

Methodologies overview

A collection of parameters
- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

Methods
- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools
The landscape

- **Difference between method and methodology**
- Difference between writing down what you did (to make it a ‘guideline’) vs. experimentally validating a methodology
- Isn’t ontology development just like conceptual data model development?
  - yes: e.g., interaction with the domain expert, data analysis
  - no: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge
- There are many methods for ontology development, but no up-to-date methodology
The landscape

- Difference between *method* and *methodology*
- Difference between writing down what you did (to make it a ‘guideline’) vs. experimentally validating a methodology
- Isn’t ontology development just like conceptual data model development?
  - yes: e.g., interaction with the domain expert, data analysis
  - no: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge
- There are many methods for ontology development, but no up-to-date methodology
The landscape

- Difference between *method* and *methodology*
- Difference between writing down what you did (to make it a ‘guideline’) vs. experimentally validating a methodology
- Isn’t ontology development just like conceptual data model development?
  - yes: e.g., interaction with the domain expert, data analysis
  - no: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge
- There are many methods for ontology development, but no up-to-date methodology
The landscape

- Difference between *method* and *methodology*
- Difference between writing down what you did (to make it a ‘guideline’) vs. experimentally validating a methodology
- Isn’t ontology development just like conceptual data model development?
  - yes: e.g., interaction with the domain expert, data analysis
  - no: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge
- There are many methods for ontology development, but no up-to-date methodology
The landscape

- Difference between *method* and *methodology*
- Difference between writing down what you did (to make it a ‘guideline’) vs. experimentally validating a methodology
- Isn’t ontology development just like conceptual data model development?
  - yes: e.g., interaction with the domain expert, data analysis
  - no: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge
- There are many methods for ontology development, but no up-to-date methodology
The landscape

• Difference between method and methodology

• Difference between writing down what you did (to make it a ‘guideline’) vs. experimentally validating a methodology

• Isn’t ontology development just like conceptual data model development?
  • yes: e.g., interaction with the domain expert, data analysis
  • no: e.g., logic, automated reasoning, using (parts of) other ontologies, different scopes/purposes, specific isolated application scenario vs. general knowledge

• There are many methods for ontology development, but no up-to-date methodology
Outline

Methodologies overview

A collection of parameters

- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

Methods

- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools
Advances and questions

- Multiple modelling issues in ontology development for the applied life sciences (e.g., part-of, uncertainty, prototypes, multilingual), methodological issues, highly specialised knowledge
- W3C’s incubator group on modelling uncertainty, mushrooming of bio-ontologies, ontology design patterns, W3C standard OWL, etc.
- Solving the early-adopter issues moves the goal-posts
  - Which ontologies are reusable for one’s own ontology?
  - What are the consequences choosing one ontology over the other?
  - The successor of OWL, OWL 2, has 5 languages: which one should be used for what and when?
Advances and questions

- Multiple modelling issues in ontology development for the applied life sciences (e.g., part-of, uncertainty, prototypes, multilingual), methodological issues, highly specialised knowledge

- W3C’s incubator group on modelling uncertainty, mushrooming of bio-ontologies, ontology design patterns, W3C standard OWL, etc.

- Solving the early-adopter issues moves the goal-posts
  - Which ontologies are reusable for one’s own ontology?
  - What are the consequences choosing one ontology over the other?
  - The successor of OWL, OWL 2, has 5 languages: which one should be used for what and when?
Advances and questions

- Multiple modelling issues in ontology development for the applied life sciences (e.g., part-of, uncertainty, prototypes, multilingual), methodological issues, highly specialised knowledge

- W3C’s incubator group on modelling uncertainty, mushrooming of bio-ontologies, ontology design patterns, W3C standard OWL, etc.

- Solving the early-adopter issues moves the goal-posts
  - Which ontologies are reusable for one’s own ontology?
  - What are the consequences choosing one ontology over the other?
  - The successor of OWL, OWL 2, has 5 languages: which one should be used for what and when?
Outline

Methodologies overview

A collection of parameters

Purposes of the ontologies
  Reusing ontologies
  Bottom-up development of ontologies
  Representation languages and reasoning services

Methods
  Guidance for modelling: OntoClean
  Debugging ontologies

Methodologies and tools
• Ontologies are application-independent, hence sole purpose of representing reality. But...

• Ontology engineers do take it into account

• A real caveat with choosing explicitly for a specific goal is that a few years after initial development of the ontology, it may get its own life and be used for other purposes than the original scope

• This, then, can require a re-engineering of the ontology (being done with, e.g., the GO and FMA)
• Ontologies are application-independent, hence sole purpose of representing reality. But...

• Ontology engineers do take it into account

• A real caveat with choosing explicitly for a specific goal is that a few years after initial development of the ontology, it may get its own life and be used for other purposes than the original scope

• This, then, can require a re-engineering of the ontology (being done with, e.g., the GO and FMA)
Possible purposes (1/2)

A. Ontology-based data access through linking data to ontologies

B. Data(base) integration, most notably the strand of applications initiated by the Gene Ontology Consortium and a successor, the OBO Foundry

C. Structured controlled vocabulary to link database records and navigate across databases on the Internet, also known as ‘linked data’;

D. Using it as part of scientific discourse and advancing research at a faster pace, including experimental ontologies in a scientific discipline and usage in computing and engineering to build prototype software;
Possible purposes (2/2)

E. As full-fledged discipline “Ontology (Science)”, where an ontology is a formal, logic-based, representation of a scientific theory, or: representation of reality;

F. Coordination and integration of Web Services;

G. Tutorial ontologies to learn modelling in the ontology development environment (e.g., the wine and pizza ontologies).
Outline

Methodologies overview

A collection of parameters

- Purposes of the ontologies
- Reusing ontologies
  - Bottom-up development of ontologies
  - Representation languages and reasoning services

Methods

- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools
Which ontologies to reuse and how? (1/2)

a. Foundational ontologies that provide generic top-level categorisations;
   ⇒ give a head-start by providing a basic structure, such as endurants being disjoint from perdurants, types of processes, attributes, and basic relations; e.g., GFO, DOLCE, BFO, RO; Marine Microbial Loops reusing DOLCE

b. ‘Reference ontologies’ that contain the main concepts of a subject domain;
   ⇒ Restricted in scope of the content, such as an ontology of measurements, and ‘top-level’ ontologies for a domain, such as BioTop, OBI
Which ontologies to reuse and how? (2/2)

c. Domain ontologies that have a (partial) overlap with the new ontology;
⇒ e.g., Gramene extending GO, reuse of the FMA
c. Domain ontologies that have a (partial) overlap with the new ontology;
⇒ e.g., Gramene extending GO, reuse of the FMA

image from http://www.imbi.uni-freiburg.de/ontology/biotop/
d. Legacy representations of information systems and ontology-like artefacts: conceptual data models of database and application software (sometimes called ‘application ontologies’), terminologies, and thesauri; ⇒ ‘ontologise’ a conceptual data model and possibly extend the contents; e.g. the conceptual data model for the bacteriocins
Examples

MicroOrganism

Disease

causes a

Genetic determinant

contains

Bacteriocin

contains

Food

ingredient in

added to

encoded on

Categorisation of food and processing (e.g., AGROVOC, AOS, Food Ontology Project, HuFO)

Biochemical compounds (e.g., GO, KEGG, CheBI, bacteriocin classification)

Genes w.r.t. prokaryotes, chromosomal DNA, plasmids, transposons (e.g., GO)

Disease categorisation (e.g., SNOMED, ICD10, infectious disease ontology)

Taxonomic information (e.g., FAO species, NCBI)
Outline

Methodologies overview

A collection of parameters

- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

Methods

- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools
Extracting in a semi-automatic way the subject domain semantics

I. Extraction of types from data in database and object-oriented software applications, including database reverse engineering and clustering;

II. Abstractions from models in textbooks and diagram-based software;

III. Text mining of documents, including scientific articles and other Digital Libraries, to find candidate terms for concepts and relations;

VI. Wisdom of the crowds and usage of those tagging techniques;

V. Other (semi-)structured data, such as excel sheets and company product catalogs.
Outline

Methodologies overview

A collection of parameters
- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

Methods
- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools
Preliminary considerations

- Depending on the purpose(s)—and, in practice, available resources, such as time, money, domain experts, and available baseline material—one ends up with either
  
  (a) a large but simple ontology, i.e., mostly just a taxonomy without, or very few, properties (relations) linked to the concepts, where ‘large’ is, roughly, > 10000 concepts, so that a simple representation language suffices;
  
  (b) a ‘medium size’ and elaborate ontology, which includes rich usage of properties, defined concepts, and, roughly, requiring OWL-DL; or
  
  (c) a small and very complex ontology, where ‘small’ is, roughly, < 250 concepts, and requiring at least OWL 2 DL

- Certain choices for reusing ontologies or legacy material, or goal, may lock one a language

- ⇒ Separate dimension that interferes with the previous parameters: the choice for a representation language
Preliminary considerations

- Depending on the purpose(s)—and, in practice, available resources, such as time, money, domain experts, and available baseline material—one ends up with either
  - (a) a large but simple ontology, i.e., mostly just a taxonomy without, or very few, properties (relations) linked to the concepts, where ‘large’ is, roughly, \( > 10000 \) concepts, so that a simple representation language suffices;
  - (b) a ‘medium size’ and elaborate ontology, which includes rich usage of properties, defined concepts, and, roughly, requiring OWL-DL; or
  - (c) a small and very complex ontology, where ‘small’ is, roughly, \( < 250 \) concepts, and requiring at least OWL 2 DL

- Certain choices for reusing ontologies or legacy material, or goal, may lock one a language

- ⇒ Separate dimension that interferes with the previous parameters: the choice for a representation language
Languages

- “OWL”: OWL-Lite, OWL-DL, OWL full
- “OWL 2” with 4 languages to tailor the choice of ontology language to fit best with the usage scope in the context of a scalable and multi-purpose SW:
  - OWL 2 DL is most expressive and based on the DL language SROIQ
  - OWL 2 EL fragment to achieve better performance with larger ontologies (e.g., for use with SNOMED-CT)
  - OWL 2 QL fragment to achieve better performance with ontologies linked to large amounts of data in secondary storage (databases); e.g. DIG-QuOnto
  - OWL 2 RL has special features to handle rules

- Differences between expressiveness of the ontology languages and their trade-offs
Reasoning services

- The current main reasoning services fall into three categories:
  - i. The ‘standard’ reasoning services for ontology usage: satisfiability and consistency checking, taxonomic classification, instance classification, and querying functionalities including epistemic and (unions of) conjunctive queries;
  - ii. Additional ‘non-standard’ reasoning services to facilitate ontology development: explanation/justification, glass-box reasoning, pinpointing errors;
  - iii. Further requirements for reasoning services identified by users, such as hypothesis testing, reasoning over role hierarchies, and discovering type-level relations from ABox instance data.
On trade-offs and choices

- OWL 2 DL, but not OWL 2 QL, has: role concatenation, qualified number restrictions, enumerated classes, covering constraint over concepts, and reflexivity, irreflexivity, and transitivity on simple roles.

- With OWL 2 QL, but not OWL 2 DL: UCQ and one can obtain similar performance as with relational databases.

- Use parameters in a software-supported selection procedure:
  - select the desired purpose and reasoning services to find the appropriate language
  - decide on purpose of usage of the ontology and one’s language, and obtain which reasoning services are available

- Purpose A or B goes well together with OWL 2 QL and query functionalities

- For purposes D and E, OWL 2 DL and the non-standard reasoning services will be more useful
On trade-offs and choices

- OWL 2 DL, but not OWL 2 QL, has: role concatenation, qualified number restrictions, enumerated classes, covering constraint over concepts, and reflexivity, irreflexivity, and transitivity on simple roles.

- With OWL 2 QL, but not OWL 2 DL: UCQ and one can obtain similar performance as with relational databases

- Use parameters in a software-supported selection procedure:
  - select the desired purpose and reasoning services to find the appropriate language
  - decide on purpose of usage of the ontology and one's language, and obtain which reasoning services are available

- Purpose A or B goes well together with OWL 2 QL and query functionalities

- For purposes D and E, OWL 2 DL and the non-standard reasoning services will be more useful
On trade-offs and choices

- OWL 2 DL, but not OWL 2 QL, has: role concatenation, qualified number restrictions, enumerated classes, covering constraint over concepts, and reflexivity, irreflexivity, and transitivity on simple roles.
- With OWL 2 QL, but not OWL 2 DL: UCQ and one can obtain similar performance as with relational databases.
- Use parameters in a software-supported selection procedure:
  - select the desired purpose and reasoning services to find the appropriate language
  - decide on purpose of usage of the ontology and one's language, and obtain which reasoning services are available
- Purpose A or B goes well together with OWL 2 QL and query functionalities
- For purposes D and E, OWL 2 DL and the non-standard reasoning services will be more useful
Recap

- Four influential factors to enhance the efficiency and effectiveness of developing ontologies:
  - seven types of purpose(s) of the ontology;
  - what and how to reuse existing ontologies and ontology-like artefacts;
  - five different types of approaches for bottom-up ontology development from other legacy sources;
  - the interaction with choice of representation language and reasoning services

- Future works pertain to setting up a software-mediated guidance system; hence, to structure and make accessible more easily the ‘soft’ knowledge about ontology development, which then could feed into design methodologies such as methontology
Recap

- Four influential factors to enhance the efficiency and effectiveness of developing ontologies:
  - seven types of purpose(s) of the ontology;
  - what and how to reuse existing ontologies and ontology-like artefacts;
  - five different types of approaches for bottom-up ontology development from other legacy sources;
  - the interaction with choice of representation language and reasoning services.

- Future works pertain to setting up a software-mediated guidance system; hence, to structure and make accessible more easily the ‘soft’ knowledge about ontology development, which then could feed into design methodologies such as methontology.
Outline

Methodologies overview

A collection of parameters
  Purposes of the ontologies
  Reusing ontologies
  Bottom-up development of ontologies
  Representation languages and reasoning services

Methods
  Guidance for modelling: OntoClean
  Debugging ontologies

Methodologies and tools
Outline

Methodologies overview

A collection of parameters
- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

Methods
- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools
OntoClean overview

- Problem: messy taxonomies on what subsumes what
- How to put them in the right order?
  - OntoClean provides guidelines for this (refer to Guarino & Welty, 2004 for an extended example)
  - Based on philosophical principles, such as identity and rigidity (see Guarino & Welty’s EKAW’00 and ECAI’00 papers for more information on the basics)
OntoClean overview

- Problem: messy taxonomies on what subsumes what
- How to put them in the right order?
- OntoClean provides guidelines for this (refer to Guarino & Welty, 2004 for an extended example)
- Based on philosophical principles, such as identity and rigidity (see Guarino & Welty’s EKAW’00 and ECAI’00 papers for more information on the basics)
Basics

- A property of an entity is *essential* to that entity if it must be true of it in every possible world, i.e. if it necessarily holds for that entity.
- Special form of essentiality is *rigidity*

**Definition (+R)**

A *rigid* property $\phi$ is a property that is essential to *all* its instances, i.e., $\forall x \phi(x) \rightarrow \Box \phi(x)$.

**Definition (-R)**

A *non-rigid* property $\phi$ is a property that is not essential to *some* of its instances, i.e., $\exists x \phi(x) \land \neg \Box \phi(x)$.
Basics

**Definition (∼R)**

An *anti-rigid* property $\phi$ is a property that is not essential to *all* its instances, i.e., $\forall x \phi(x) \rightarrow \neg \square \phi(x)$.

**Definition (¬R)**

A *semi-rigid* property $\phi$ is a property that is non-rigid but not anti-rigid.

- Anti-rigid properties cannot subsume rigid properties
Basics

- **Identity**: being able to recognize individual entities in the world as being the same (or different)

- **Unity**: being able to recognize all the parts that form an individual entity; e.g., ocean carries unity (+U), legal agent carries no unity (-U), and amount of water carries anti-unity (“not necessarily wholes”, ∼U)

- **Identity criteria** are the criteria we use to answer questions like, “is that my dog?”

- Identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions)
Basics

- **Identity**: being able to recognize individual entities in the world as being the same (or different)
- **Unity**: being able to recognize all the parts that form an individual entity; e.g., ocean carries unity (+U), legal agent carries no unity (-U), and amount of water carries anti-unity (“not necessarily wholes”, ∼U)
- **Identity criteria** are the criteria we use to answer questions like, “is that my dog?”
- Identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions)
Basics

• *Identity*: being able to recognize individual entities in the world as being the same (or different)

• *Unity*: being able to recognize all the parts that form an individual entity; e.g., ocean carries unity (+U), legal agent carries no unity (-U), and amount of water carries anti-unity (“not necessarily wholes”, ∼U)

• *Identity criteria* are the criteria we use to answer questions like, “is that my dog?”

• Identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions)
Basics

- **Identity**: being able to recognize individual entities in the world as being the same (or different)
- **Unity**: being able to recognize all the parts that form an individual entity; e.g., ocean carries unity (+U), legal agent carries no unity (-U), and amount of water carries anti-unity ("not necessarily wholes", ∼U)
- **Identity criteria** are the criteria we use to answer questions like, “is that my dog?”
- Identity criteria are conditions used to determine equality (sufficient conditions) and that are entailed by equality (necessary conditions)
**Basics**

**Definition**

A non-rigid property carries an IC $\Gamma$ iff it is subsumed by a rigid property carrying $\Gamma$.

**Definition**

A property $\phi$ supplies an IC $\Gamma$ iff i) it is rigid; ii) it carries $\Gamma$; and iii) $\Gamma$ is not carried by all the properties subsuming $\phi$. This means that, if $\phi$ inherits different (but compatible) ICs from multiple properties, it still counts as supplying an IC.

- Any property carrying an IC: $+I$ ($-I$ otherwise).
- Any property supplying an IC: $+O$ ($-O$ otherwise); “O” is a mnemonic for “own identity”
- $+O$ implies $+I$ and $+R$
Formal ontological property classifications

<table>
<thead>
<tr>
<th>+O</th>
<th>+I</th>
<th>+R</th>
<th>+D</th>
<th>-D</th>
<th>Type</th>
<th>Sortal</th>
</tr>
</thead>
<tbody>
<tr>
<td>-O</td>
<td>+I</td>
<td>+R</td>
<td>+D</td>
<td>-D</td>
<td>Quasi-Type</td>
<td></td>
</tr>
<tr>
<td>-O</td>
<td>+I</td>
<td>~R</td>
<td>+D</td>
<td>-D</td>
<td>Material role</td>
<td></td>
</tr>
<tr>
<td>-O</td>
<td>+I</td>
<td>~R</td>
<td>-D</td>
<td></td>
<td>Phased sortal</td>
<td></td>
</tr>
<tr>
<td>-O</td>
<td>+I</td>
<td>¬R</td>
<td>+D</td>
<td>-D</td>
<td>Mixin</td>
<td></td>
</tr>
<tr>
<td>-O</td>
<td>-I</td>
<td>+R</td>
<td>+D</td>
<td>-D</td>
<td>Category</td>
<td>Non-Sortal</td>
</tr>
<tr>
<td>-O</td>
<td>-I</td>
<td>~R</td>
<td>+D</td>
<td></td>
<td>Formal role</td>
<td></td>
</tr>
<tr>
<td>-O</td>
<td>-I</td>
<td>¬R</td>
<td>-D</td>
<td></td>
<td>Attribution</td>
<td></td>
</tr>
</tbody>
</table>
Formal ontological property classifications

- **Property**
  - **Sortal**
    - Non-rigid Mixin Phased sortal Caterpillar, Chrysalis, Butterfly (for Papilionoidae)
  - Rigid Type Cat, Chair Quasi-type Herbivore
    - **Non-rigid**
      - **Role**
        - **Anti-rigid**
          - **Category** Endurant, Abstract entity
          - **Attribution** Blue, Spherical
          - **Formal role** Recipient
          - **Material role** Student, Food
          - **Phased sortal** Caterpillar, Chrysalis, Butterfly (for Papilionoidae)
          - **Mixin**
            - **Type** Cat, Chair
            - **Quasi-type** Herbivore

- **Non-sortal**
  - **Role**
    - **Non-rigid**
      - **Anti-rigid**
Basics

- Given two properties, \( p \) and \( q \), when \( q \) subsumes \( p \) the following constraints hold:
  1. If \( q \) is anti-rigid, then \( p \) must be anti-rigid
  2. If \( q \) carries an identity criterion, then \( p \) must carry the same criterion
  3. If \( q \) carries a unity criterion, then \( p \) must carry the same criterion
  4. If \( q \) has anti-unity, then \( p \) must also have anti-unity
Example: before
Example: after
Outline

Methodologies overview

A collection of parameters
  - Purposes of the ontologies
  - Reusing ontologies
  - Bottom-up development of ontologies
  - Representation languages and reasoning services

Methods
  - Guidance for modelling: OntoClean
  - Debugging ontologies

Methodologies and tools
Overview

• Domain experts are expert in their subject domain, which is not logic
  • Modellers often do not understand the subject domain well
  • The more expressive the language, the easier it is to make errors or bump into unintended entailments
  • Simple languages can represent more than it initially may seem (by some more elaborate encoding), which clutters the ontology and affects comprehension
  • In short: people make errors (w.r.t. their intentions) in the modelling task, and automated reasoners can help fix that
Overview

- Domain experts are expert in their subject domain, which is not logic
- Modellers often do not understand the subject domain well
  - The more expressive the language, the easier it is to make errors or bump into unintended entailments
  - Simple languages can represent more than it initially may seem (by some more elaborate encoding), which clutters the ontology and affects comprehension
- In short: people make errors (w.r.t. their intentions) in the modelling task, and automated reasoners can help fix that
Overview

- Domain experts are expert in their subject domain, which is not logic
- Modellers often do not understand the subject domain well
- The more expressive the language, the easier it is to make errors or bump into unintended entailments
- Simple languages can represent more than it initially may seem (by some more elaborate encoding), which clutters the ontology and affects comprehension
- In short: people make errors (w.r.t. their intentions) in the modelling task, and automated reasoners can help fix that
Overview

- Domain experts are expert in their subject domain, which is not logic
- Modellers often do not understand the subject domain well
- The more expressive the language, the easier it is to make errors or bump into unintended entailments
- Simple languages can represent more than it initially may seem (by some more elaborate encoding), which clutters the ontology and affects comprehension
- In short: people make errors (w.r.t. their intentions) in the modelling task, and automated reasoners can help fix that
Overview

- Domain experts are expert in their subject domain, which is not logic
- Modellers often do not understand the subject domain well
- The more expressive the language, the easier it is to make errors or bump into unintended entailments
- Simple languages can represent more than it initially may seem (by some more elaborate encoding), which clutters the ontology and affects comprehension
- In short: people make errors (w.r.t. their intentions) in the modelling task, and automated reasoners can help fix that
Overview

• Using automated reasoners for ‘debugging’ ontologies, requires one to know about reasoning services
• Using standard reasoning services (recollect slide 22)
• New reasoning services tailored to pinpointing the errors and explaining the entailments (slide 22)
• Details in KR & onto course next semester (and the two papers in the ‘recommended readings’ of this lecture), here a general overview
Common errors

- Unsatisfiable classes
  - In the tools: the unsatisfiable classes end up as direct subclass of owl:Nothing
  - Sometimes one little error generates a whole cascade of unsatisfiable classes

- Satisfiability checking can cause rearrangement of the class tree and any inferred relationships to be associated with a class definition: ‘desirable’ vs. ‘undesirable’ inferred subsumptions

- Inconsistent ontologies: all classes taken together unsatisfiable
Common errors

- Unsatisfiable classes
  - In the tools: the unsatisfiable classes end up as direct subclass of owl:Nothing
  - Sometimes one little error generates a whole cascade of unsatisfiable classes

- Satisfiability checking can cause rearrangement of the class tree and any inferred relationships to be associated with a class definition: ‘desirable’ vs. ‘undesirable’ inferred subsumptions

- Inconsistent ontologies: all classes taken together unsatisfiable
Common errors

- Unsatisfiable classes
  - In the tools: the unsatisfiable classes end up as direct subclass of owl:Nothing
  - Sometimes one little error generates a whole cascade of unsatisfiable classes

- Satisfiability checking can cause rearrangement of the class tree and any inferred relationships to be associated with a class definition: ‘desirable’ vs. ‘undesireable’ inferred subsumptions

- Inconsistent ontologies: all classes taken together unsatisfiable
Common errors

- Basic set of clashes for concepts (w.r.t. tableaux algorithms) are:
  - Atomic: An individual belongs to a class and its complement
  - Cardinality: An individual has a max cardinality restriction but is related to more distinct individuals
  - Datatype: A literal value violates the (global or local) range restrictions on a datatype property

- Basic set of clashes for KBs (ontology + instances) are:
  - Inconsistency of Assertions about Individuals, e.g., an individual is asserted to belong to disjoint classes or has a cardinality restriction but related to more individuals
  - Individuals Related to Unsatisfiable Classes
  - Defects in Class Axioms Involving Nominals (owl:oneOf, if present in the language)
Common errors

- Basic set of clashes for concepts (w.r.t. tableaux algorithms) are:
  - Atomic: An individual belongs to a class and its complement
  - Cardinality: An individual has a max cardinality restriction but is related to more distinct individuals
  - Datatype: A literal value violates the (global or local) range restrictions on a datatype property

- Basic set of clashes for KBs (ontology + instances) are:
  - Inconsistency of Assertions about Individuals, e.g., an individual is asserted to belong to disjoint classes or has a cardinality restriction but related to more individuals
  - Individuals Related to Unsatisfiable Classes
  - Defects in Class Axioms Involving Nominals (owl:oneOf, if present in the language)
## Outline

### Methodologies overview

### A collection of parameters
- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

### Methods
- Guidance for modelling: OntoClean
- Debugging ontologies

### Methodologies and tools
Where are we?

- Parameters that affect ontology development, such as purpose, base material, language
- Methods, such as reverse engineering, text mining to start, OntoClean to improve
- Tools to model, to reason, to debug, to integrate, to link to data
- Methodologies that are coarse-grained: they do not (yet) contain all the permutations at each step, i.e. *what* and *how* to do each step, given the recent developments;
- e.g. step x is “knowledge acquisition”, but what are its component-steps?
Where are we?

- Parameters that affect ontology development, such as purpose, base material, language
- Methods, such as reverse engineering text mining to start, OntoClean to improve
- Tools to model, to reason, to debug, to integrate, to link to data
- Methodologies that are coarse-grained: they do not (yet) contain all the permutations at each step, i.e. what and how to do each step, given the recent developments;
- e.g. step x is “knowledge acquisition”, but what are it component-steps?
Example methodology: **METHONTOLOGY**

- **Basic methodology:**
  - specification: why, what are its intended uses, who are the prospective users
  - conceptualization, with intermediate representations
  - formalization (transforms the domain-expert understandable ‘conceptual model’ into a formal or semi-computable model)
  - implementation (represent it in an ontology language)
  - maintenance (corrections, updates, etc)

- **Additional tasks (as identified by METHONTOLOGY):**
  - Management activities (schedule, control, and quality assurance)
  - Support activities (knowledge acquisition, integration, evaluation, documentation, and configuration management)

- Applied to chemical, legal domain, and others
- More comprehensive assessment of extant methodologies in Corcho et al, 2003
**Example methodology: Methontology**

- **Basic methodology:**
  - specification: why, what are its intended uses, who are the prospective users
  - conceptualization, with intermediate representations
  - formalization (transforms the domain-expert understandable ‘conceptual model’ into a formal or semi-computable model)
  - implementation (represent it in an ontology language)
  - maintenance (corrections, updates, etc)

- **Additional tasks (as identified by Methontology)**
  - Management activities (schedule, control, and quality assurance)
  - Support activities (knowledge acquisition, integration, evaluation, documentation, and configuration management)

- Applied to chemical, legal domain, and others
- More comprehensive assessment of extant methodologies in Corcho et al, 2003
Example methodology: **Methontology**

- **Basic methodology:**
  - specification: why, what are its intended uses, who are the prospective users
  - conceptualization, with intermediate representations
  - formalization (transforms the domain-expert understandable ‘conceptual model’ into a formal or semi-computable model)
  - implementation (represent it in an ontology language)
  - maintenance (corrections, updates, etc)

- **Additional tasks (as identified by Methontology)**
  - Management activities (schedule, control, and quality assurance)
  - Support activities (knowledge acquisition, integration, evaluation, documentation, and configuration management)

- Applied to chemical, legal domain, and others
- More comprehensive assessment of extant methodologies in Corcho et al., 2003
MOdelling wiKI

- MoKi is based on a **SemanticWiki**, which is used for collaborative and cooperative ontology development.
- It enables actors with different expertise to develop an “enterprise model” not only using structural (formal) descriptions but also adopting more informal and semi-formal descriptions of knowledge\(^1\)
  - access to the enterprise model at different levels of formality: informal, semi-formal and formal
  - more info at http://moki.fbk.eu

\(^1\)enterprise model: “a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise”
MOdelling wiKI

- MoKi is based on a **SemanticWiki**, which is used for collaborative and cooperative ontology development.
- It enables actors with different expertise to develop an “enterprise model” not only using structural (formal) descriptions but also adopting more informal and semi-formal descriptions of knowledge\(^1\).
- Access to the enterprise model **at different levels of formality**: informal, semi-formal and formal.

---

\(^1\)enterprise model: “a computational representation of the structure, activities, processes, information, resources, people, behavior, goals, and constraints of a business, government, or other enterprise”
Extending the methodologies

- **Methodontology**, MoKi, and others (e.g., On-To-Knowledge, KACTUS approach) are for developing one *single* ontology
- Changing landscape in ontology development towards building “ontology networks”
- Characteristics: dynamics, context, collaborative, distributed
- E.g. the emerging NeOn methodology (more info at http://www.neon-project.org/web-content/images/Publications/neon_2008_d5.4.1.pdf)
Extending the methodologies

- **Methontology**, MoKi, and others (e.g., On-To-Knowledge, KACTUS approach) are for developing one single ontology
- Changing landscape in ontology development towards building “ontology networks”
- Characteristics: dynamics, context, collaborative, distributed
- E.g. the emerging NeOn methodology (more info at http://www.neon-project.org/web-content/images/Publications/neon_2008_d5.4.1.pdf)
Extending the methodologies: NeOn

- NeOn’s “Glossary of Activities” identifies and defines 55 activities when ontology networks are collaboratively built.
- Among others: ontology localization, -alignment, -formalization, -diagnosis, -enrichment etc.
- Divided into a matrix with “required” and “if applicable”
- Embedded into a comprehensive methodology (under development)
Extending the methodologies: NeOn

- NeOn’s “Glossary of Activities” identifies and defines 55 activities when ontology networks are collaboratively built.
- Among others: ontology localization, -alignment, -formalization, -diagnosis, -enrichment etc.
- Divided into a matrix with “required” and “if applicable”
- Embedded into a comprehensive methodology (under development)
Scenarios for Building Ontology Networks

Knowledge Resources

- Non Ontological Resources
  - Glossaries
  - Dictionaries
  - Lexicons
  - Classification Schemes
  - Taxonomies
  - Thesauri

- Ontological Resources
  - Ontology Design Patterns
  - Ontology Repositories and Registries
    - Flogic RDF(S)
    - OWL

- Ontological Resource Reuse
- Ontology Design Pattern Reuse
- Ontological Resource Reengineering
- Ontology Aligning
- Ontology Merging

Ontology Support Activities: Knowledge Acquisition (Elicitation), Documentation, Configuration Management, Evaluation (V&V), Assessment
Tools

• Thus far, no tool gives you everything
  • WebODE to support METHONTOLOGY with a software application
  • Protégé with its plugins. a.o.: ontology visualisation, querying, OBDA, etc.
  • NeOn toolkit aims to be a “open source multi-platform ontology engineering environment, which aims to provide comprehensive support for all activities in the ontology engineering life-cycle”; 45 plugins
  • RacerPro, RacerPorter. a.o.: sophisticated querying
  • KAON, SWOOP, etc.
  • Specialised tools for specific task, such as ontology integration and evaluation (e.g. Protégé-PROMPT, ODEClean)
  • RDF-based ones, such as Sesame

• Longer list and links to more lists of tools at the end of the lecture’s blog post
Tools

• Thus far, no tool gives you everything
  • WebODE to support Methontology with a software application
  • Protégé with its plugins. a.o.: ontology visualisation, querying, OBDA, etc.
  • NeOn toolkit aims to be a “open source multi-platform ontology engineering environment, which aims to provide comprehensive support for all activities in the ontology engineering life-cycle”; 45 plugins
  • RacerPro, RacerPorter. a.o.: sophisticated querying
  • KAON, SWOOP, etc.
  • Specialised tools for specific task, such as ontology integration and evaluation (e.g. Protégé-PROMPT, ODEClean)
  • RDF-based ones, such as Sesame

• Longer list and links to more lists of tools at the end of the lecture’s blog post
Tools

- Thus far, no tool gives you everything
  - WebODE to support *Methontology* with a software application
  - Protégé with its plugins. a.o.: ontology visualisation, querying, OBDA, etc.
  - NeOn toolkit aims to be a “open source multi-platform ontology engineering environment, which aims to provide comprehensive support for all activities in the ontology engineering life-cycle”; 45 plugins
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - KAON, SWOOP, etc.
  - Specialised tools for specific task, such as ontology integration and evaluation (e.g. Protégé-PROMPT, ODEClean)
  - RDF-based ones, such as Sesame

- Longer list and links to more lists of tools at the end of the lecture’s blog post
Tools

- Thus far, no tool gives you everything
  - WebODE to support Methontology with a software application
  - Protégé with its plugins. a.o.: ontology visualisation, querying, OBDA, etc.
  - NeOn toolkit aims to be a “open source multi-platform ontology engineering environment, which aims to provide comprehensive support for all activities in the ontology engineering life-cycle”; 45 plugins
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - KAON, SWOOP, etc.
  - Specialised tools for specific task, such as ontology integration and evaluation (e.g. Protégé-PROMPT, ODEClean)
  - RDF-based ones, such as Sesame

- Longer list and links to more lists of tools at the end of the lecture’s blog post
Tools

- Thus far, no tool gives you everything
  - WebODE to support Methontology with a software application
  - Protégé with its plugins. a.o.: ontology visualisation, querying, OBDA, etc.
  - NeOn toolkit aims to be a “open source multi-platform ontology engineering environment, which aims to provide comprehensive support for all activities in the ontology engineering life-cycle”; 45 plugins
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - KAON, SWOOP, etc.
  - Specialised tools for specific task, such as ontology integration and evaluation (e.g. Protégé-PROMPT, ODEClean)
  - RDF-based ones, such as Sesame

- Longer list and links to more lists of tools at the end of the lecture’s blog post
Summary

Methodologies overview

A collection of parameters
- Purposes of the ontologies
- Reusing ontologies
- Bottom-up development of ontologies
- Representation languages and reasoning services

Methods
- Guidance for modelling: OntoClean
- Debugging ontologies

Methodologies and tools