COMP718: Ontologies and Knowledge Bases

Lecture 8: Methods and Methodologies

Maria Keet

email: keet@ukzn.ac.za

home: http://www.meteck.org

School of Mathematics, Statistics, and Computer Science
University of KwaZulu-Natal, South Africa

27 March 2012

Outline

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 vs. general knowledge, application scenarios vs. general knowledge
- There are plenty methods and guidelines for ontology development—use them.

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, application scenarios vs. general knowledge
- There are plenty methods and guidelines for ontology development—use them.

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 plenty methods and guidelines for ontology development—use them.

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 plenty methods and guidelines for ontology development—use them.

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 plenty methods and guidelines for ontology development—use them.

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 plenty methods and guidelines for ontology development—use them.

Outline

The changing landscape

- Multiple modelling issues in ontology development (e.g., part-of, uncertainty, prototypes, multilingualism), 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 has moved the goal-posts, and uncovered new issues; e.g.:
  - Which ontologies are reusable for one's own ontology, in whole or in part?
  - What are the consequences of choosing one ontology over the other?
  - OWL 2 has 5 languages: which one should be used for what and when?

Parameters

- Which parameters affect ontology development?
  Where?
  How?
- We consider:
  - Purposes
  - Reusing ontologies
  - Bottom-up development
  - Languages
  - Reasoning services

Parameters

- Which parameters affect ontology development?
  Where?
  How?
- We consider:
  - Purposes
  - Reusing ontologies
  - Bottom-up development
  - Languages
  - Reasoning services

Purposes

- Querying data by means of an ontology (OBDA) through linking databases to an ontology
- Database integration
- Structured controlled vocabulary to link data(base) records and navigate across databases on the Internet ('linked data')
- Using it as part of scientific discourse and advancing research at a faster pace, (including experimental ontologies)
- Coordination among and integration of Web Services

Purposes

- Ontology in an ontology-driven information system destined for run-time usage, e.g., in scientific workflows, MASs, ontology-mediated data clustering, and user interaction in e-learning
- Ontologies for NLP, e.g. annotating and querying Digital Libraries and scientific literature, QA systems, and materials for e-learning
- As full-fledged discipline "Ontology (Science)", where an ontology is a formal, logic-based, representation of a scientific theory
- Tutorial ontologies, e.g., the wine and pizza ontologies

Reusing ontologies

- Foundational ontologies
- Reference ontologies
- Domain ontologies that have an overlap with the new ontology;
- For each of them, resource usage considerations, such as
  - Availability of the resource (open, copyright)
  - If the source is being maintained or abandoned one-off effort;
  - Community effort, research group, and if it has already some adoption or usage;
  - Subject to standardization policies or stable releases;
  - If the ontology is available in the desired or required ontology language.

Reusing ontologies

- Foundational ontologies
- Reference ontologies
- Domain ontologies that have an overlap with the new ontology;
- For each of them, resource usage considerations, such as
  - Availability of the resource (open, copyright)
  - If the source is being maintained or abandoned one-off effort;
  - Community effort, research group, and if it has already some adoption or usage;
  - Subject to standardization policies or stable releases;
  - If the ontology is available in the desired or required ontology language.

Example

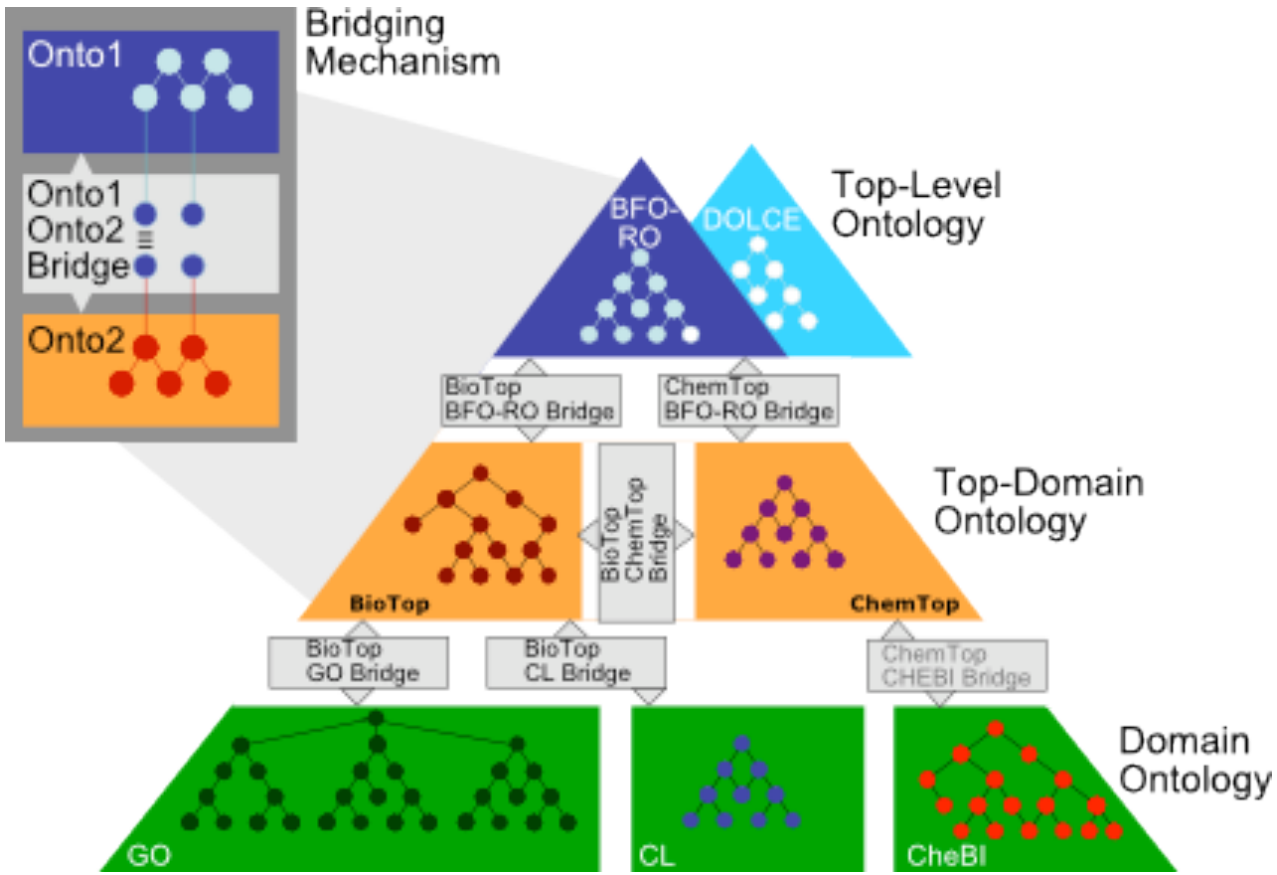


image from http://www.imbi.uni-freiburg.de/ontology/biotop/

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Bottom-up development

- Reuse of other knowledge-based representations:
  - conceptual data models (UML diagrams, ER, and ORM)
- Database (and OO) reverse engineering, and least common subsumer and clustering to infer new concepts;
- Abstractions from or formalisations of models in textbooks and diagram-based software;
- Thesauri and other structured vocabularies;
- Other (semi-)structured data, such as spreadsheets and company product catalogs;
- Text mining of documents to find candidate terms for concepts and relations;
- Terminologies, lexicons, and glossaries;
- Wisdom of the crowds tagging, tagging games, and folksonomies;

Languages – preliminary considerations

- Depending on the purpose(s) (and available resources), 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 large 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
- $\Rightarrow$ Separate dimension that interferes with the previous parameters: the choice for a representation language

Languages – preliminary considerations

- Depending on the purpose(s) (and available resources), 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 large 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
- $\Rightarrow$ Separate dimension that interferes with the previous parameters: the choice for a representation language

Languages

- Older KR languages (frames, obo, conceptual graphs, etc.)
- Web Ontology 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 $\mathcal{SROIQ}$
    - 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
- Extensions (probabilistic, fuzzy, temporal, etc.)
- Differences between expressiveness of the ontology languages and their trade-offs

Languages

- Older KR languages (frames, obo, conceptual graphs, etc.)
- Web Ontology 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 $\mathcal{SROIQ}$
    - 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
- Extensions (probabilistic, fuzzy, temporal, etc.)
- Differences between expressiveness of the ontology languages and their trade-offs

Languages

- Older KR languages (frames, obo, conceptual graphs, etc.)
- Web Ontology 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 $\mathcal{SROIQ}$
    - 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
- Extensions (probabilistic, fuzzy, temporal, etc.)
- Differences between expressiveness of the ontology languages and their trade-offs

Reasoning services

- Description logics-based reasoning services
  - The standard reasoning services for ontology usage: satisfiability and consistency checking, taxonomic classification, instance classification;
  - 'Non-standard' reasoning services to facilitate ontology development: explanation/justification, glass-box reasoning, pin-pointing errors, least-common subsumer;
  - Querying functionalities, such as epistemic and (unions of) conjunctive queries;
- Ontological reasoning services (OntoClean, RBox reasoning service)
- Other technologies (e.g., Bayesian networks)

	SKOS	OWL Language: 2 QL	2 EL	2 DL	DL	Extensions	Ontology reuse: foundational	reference	domain
Purpose ⇓									
1. Query data	−	+	−	−	−	+	−	−	±
2. Database integration	+	+	+	−	−	±	±	±	+
3. Integration / record navigation	+	+	+	−	−	−	−	±	+
4. Part of scientific discourse	−	−	−	+	+	+	+	+	+
5. Web services orchestration	−	−	+	±	+	−	±	+	+
6. ODIS	±	+	+	−	±	±	±	+	+
7. ontoNLP	+	+	+	−	±	−	±	+	+
8. Science	−	−	−	+	±	+	+	+	−
9. Tutorial ontology	−	−	−	+	+	±	−	−	+
Reasoning services ⇓									
1. Standard	−	±	±	+	+	+			
2. Non-standard	−	±	±	+	+	−			
3. Querying	−	+	+	−	−	±			
4. Ontological	+	+	+	+	+	+			
Bottom-up ⇓									
1. Other KR/CM	−	±	±	+	+	−			
2. DB reverse	−	±	±	+	+	−			
3. Textbook models	−	−	±	+	+	+			
4. Thesauri	+	±	+	−	−	−			
5. Other semi-structured	±	±	+	−	−	−			
6. Text mining	+	±	+	−	−	−			
7. Terminologies	+	±	+	−	−	−			
8. Tagging	+	±	+	−	−	−			
Ontology reuse ⇓									
1. Foundational	−	−	−	+	±	−			
2. Reference	−	±	±	+	+	−			
3. Domain	±	±	+	+	+	−			

Table 1 Basic cross-matching between realistic combinations of parameters. The more complex dependencies, such as the interaction between purpose, language, and reasoning service, can be obtained from traversing the table (*purpose* ↔

Outline

OntoClean overview

- Problem: messy taxonomies on what subsumes what
- How to put them in the right order?
- OntoClean provides guidelines for this (see 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 (see 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 ϕ 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 ϕ is a property that is not essential to *some* of its instances, i.e., $\exists x \phi(x) \wedge \neg \Box \phi(x)$.

Basics

Definition (∼R)

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

Definition (¬R)

A *semi-rigid* property ϕ 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 Γ iff it is subsumed by a rigid property carrying Γ.

Definition

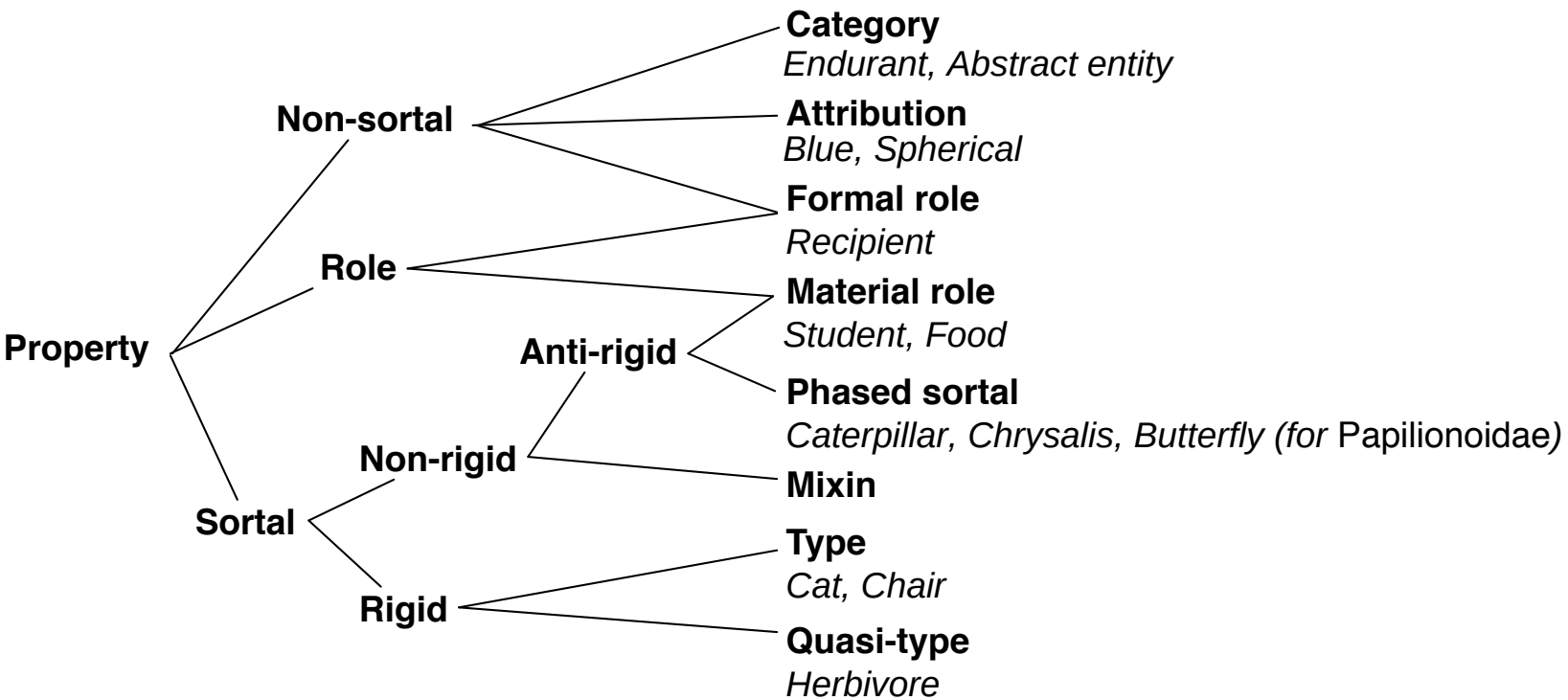
A property ϕ supplies an IC Γ iff i) it is rigid; ii) it carries Γ; and iii) Γ is not carried by all the properties subsuming ϕ. This means that, if ϕ inherits different (but compatible) ICs from multiple properties, it still counts as supplying an IC.

- 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

+O	+I	+R	+D	Type	Sortal
			-D		
-O	+I	+R	+D	Quasi-Type	
			-D		
-O	+I	~R	+D	Material role	
-O	+I	~R	-D	Phased sortal	
-O	+I	¬R	+D	Mixin	
			-D		
-O	-I	+R	+D	Category	Non-Sortal
			-D		
-O	-I	~R	+D	Formal role	
-O	-I	~R	-D	Attribution	
		¬R	+D		
			-D		

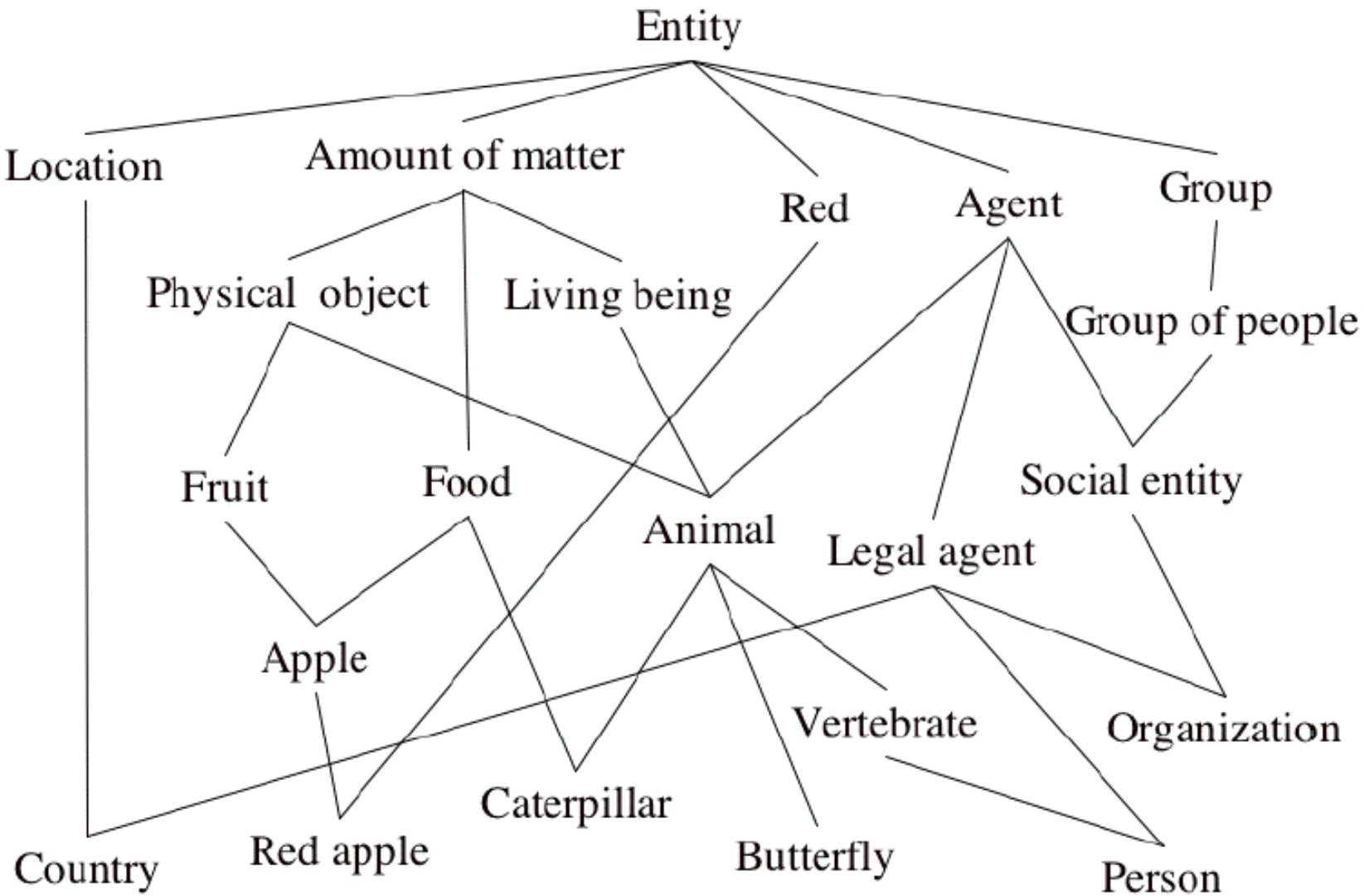
Formal ontological property classifications



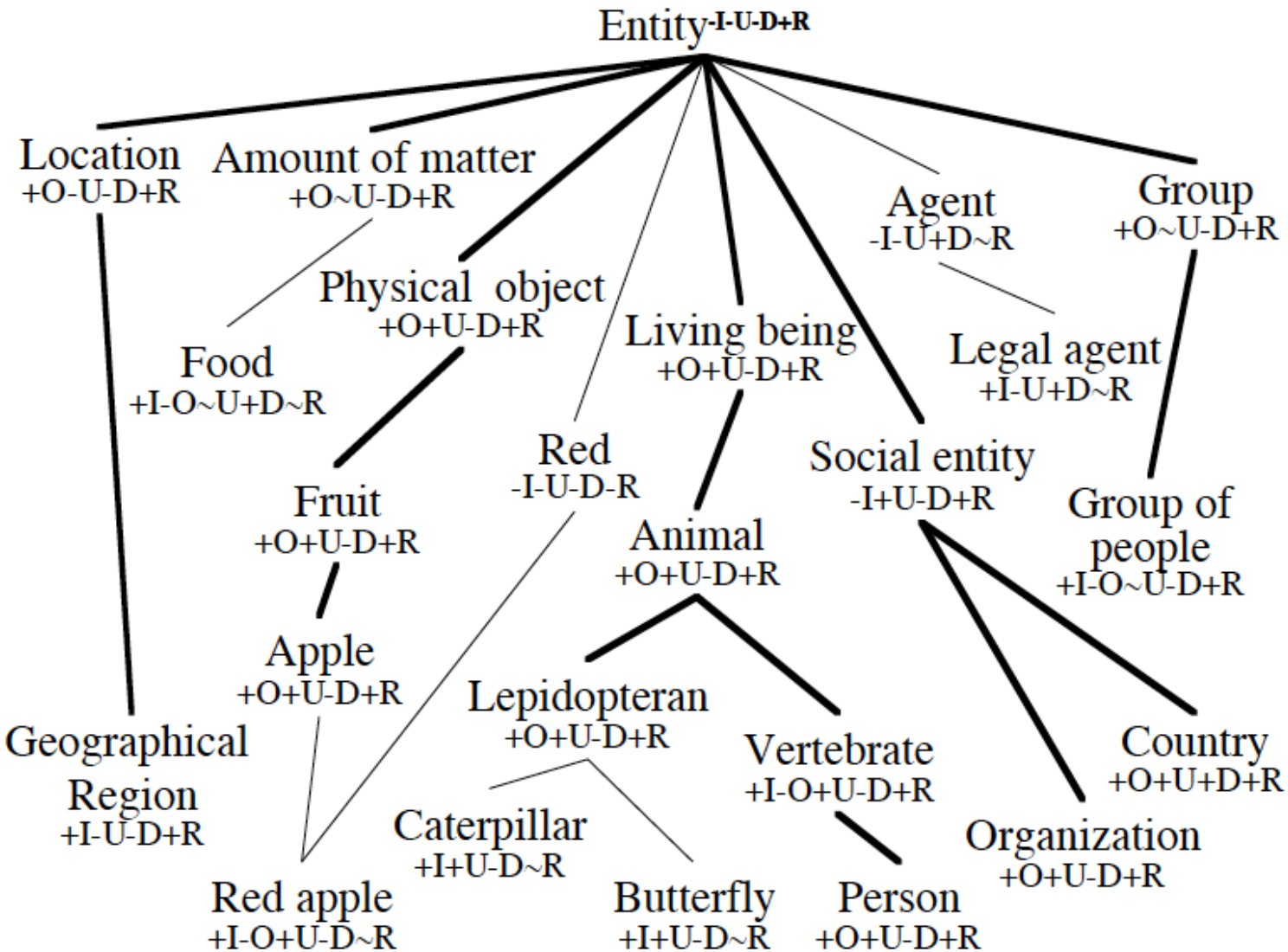
Basic rules

- 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 IC, then p must carry the same IC
  3. If q carries a UC, then p must carry the same UC
  4. If q has anti-unity, then p must also have anti-unity
5. Incompatible IC's are disjoint, and Incompatible UC's are disjoint
- And, in shorthand:
  6. $+R \not\subset\sim R$
  7. $-I \not\subset +I$
  8. $-U \not\subset +U$
  9. $+U \not\subset\sim U$
  10. $-D \not\subset +D$

Example: before



Example: after



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
- New reasoning services tailored to pinpointing the errors and explaining the entailments

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

- 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

- 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

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, i..e, a macro-level, processual information systems perspective; 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?

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, i..e, a macro-level, processual information systems perspective; 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 methodological steps:
  - 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:
  - 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 methodological steps:
  - 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:
  - 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 methodological steps:
  - 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:
  - 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)

Generalisation

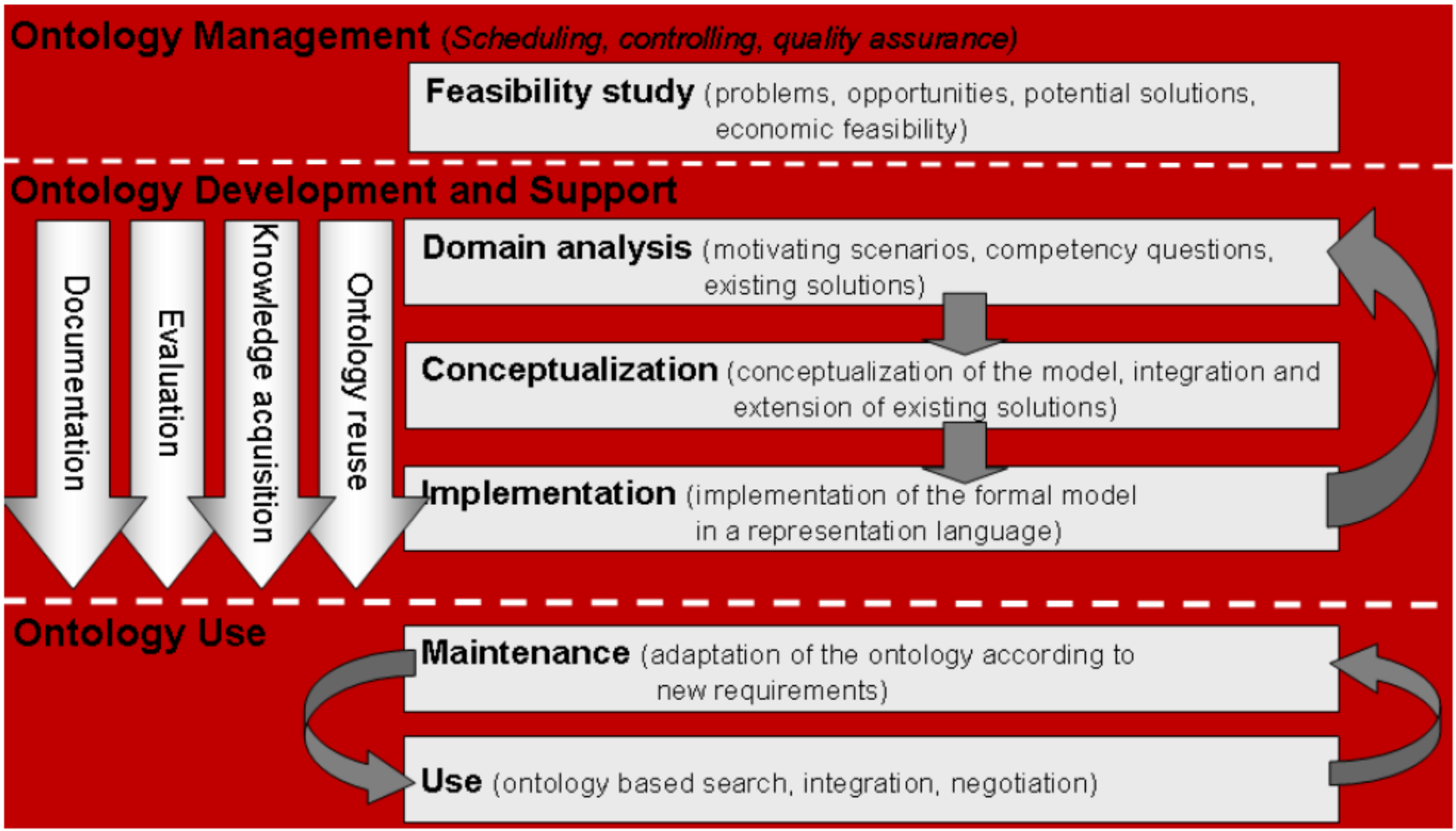


Figure: Main tasks in ontology engineering (Source: Simperl10)

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"[1]: use both *structural (formal) descriptions* and *more informal* and *semi-formal* descriptions of knowledge
- ⇒ access to the enterprise model **at different levels of formality**: informal, semi-formal and formal
- more info and demo 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"[1]: use both *structural (formal) descriptions* and *more informal* and *semi-formal* descriptions of knowledge
- $\Rightarrow$ access to the enterprise model **at different levels of formality**: informal, semi-formal and formal
- more info and demo 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"

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

Extending the methodologies

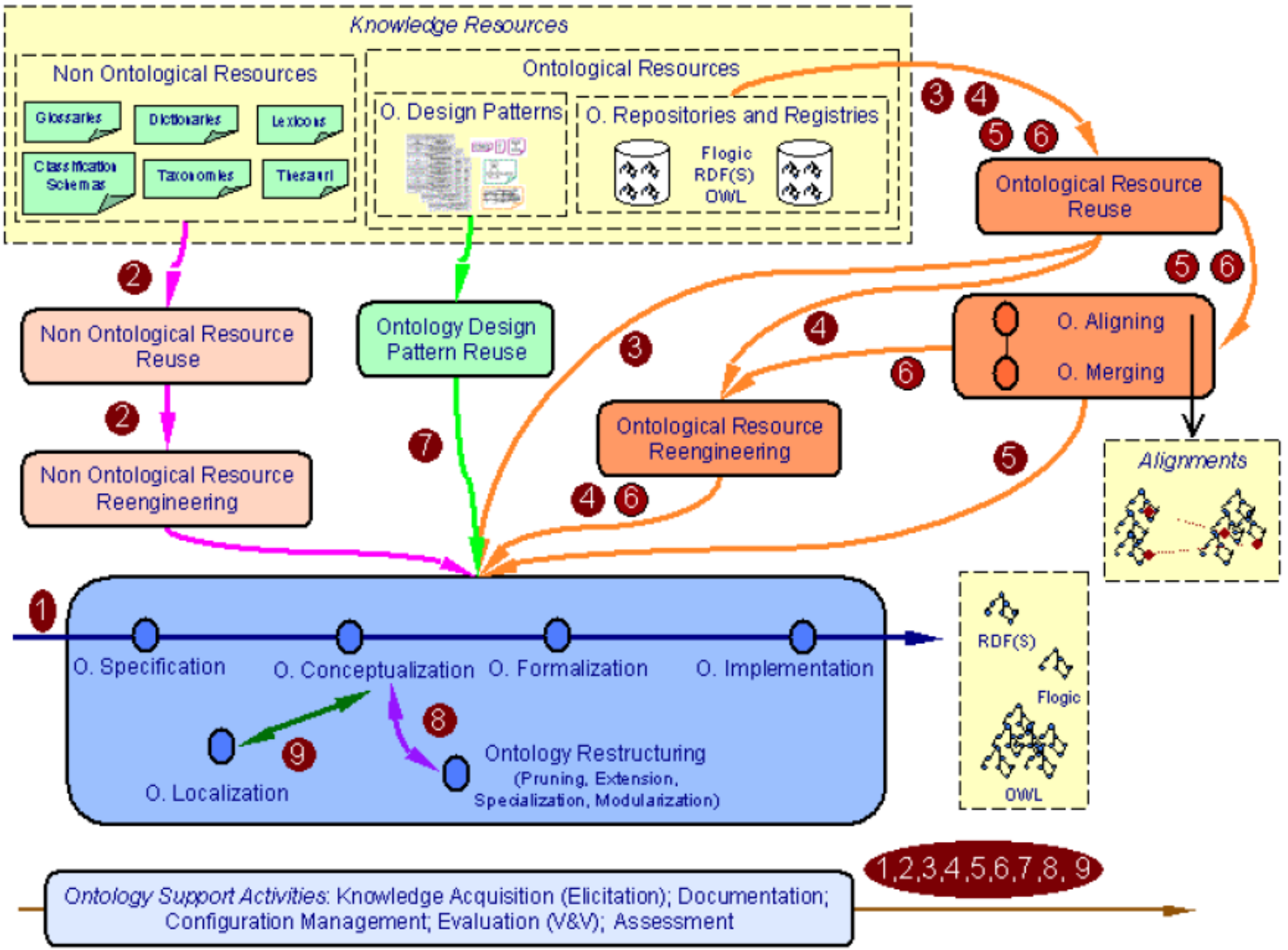
- METHONTOLOGY, MoKi, and others (e.g., On-To-Knowledge, KACTUS approach) are for developing one *single* ontology
- Changing landscape in ontology development towards building "ontology networks"
- Characteristics: dynamics, context, collaborative, distributed
- E.g., the emerging NeOn methodology

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) that
- Recognises there are *several scenarios for ontology development*, i.e., refining the typical monolithic 'waterfall' approach

(more info in neon_2008_d5.4.1.pdf)

Several scenarios for Building Ontology Networks



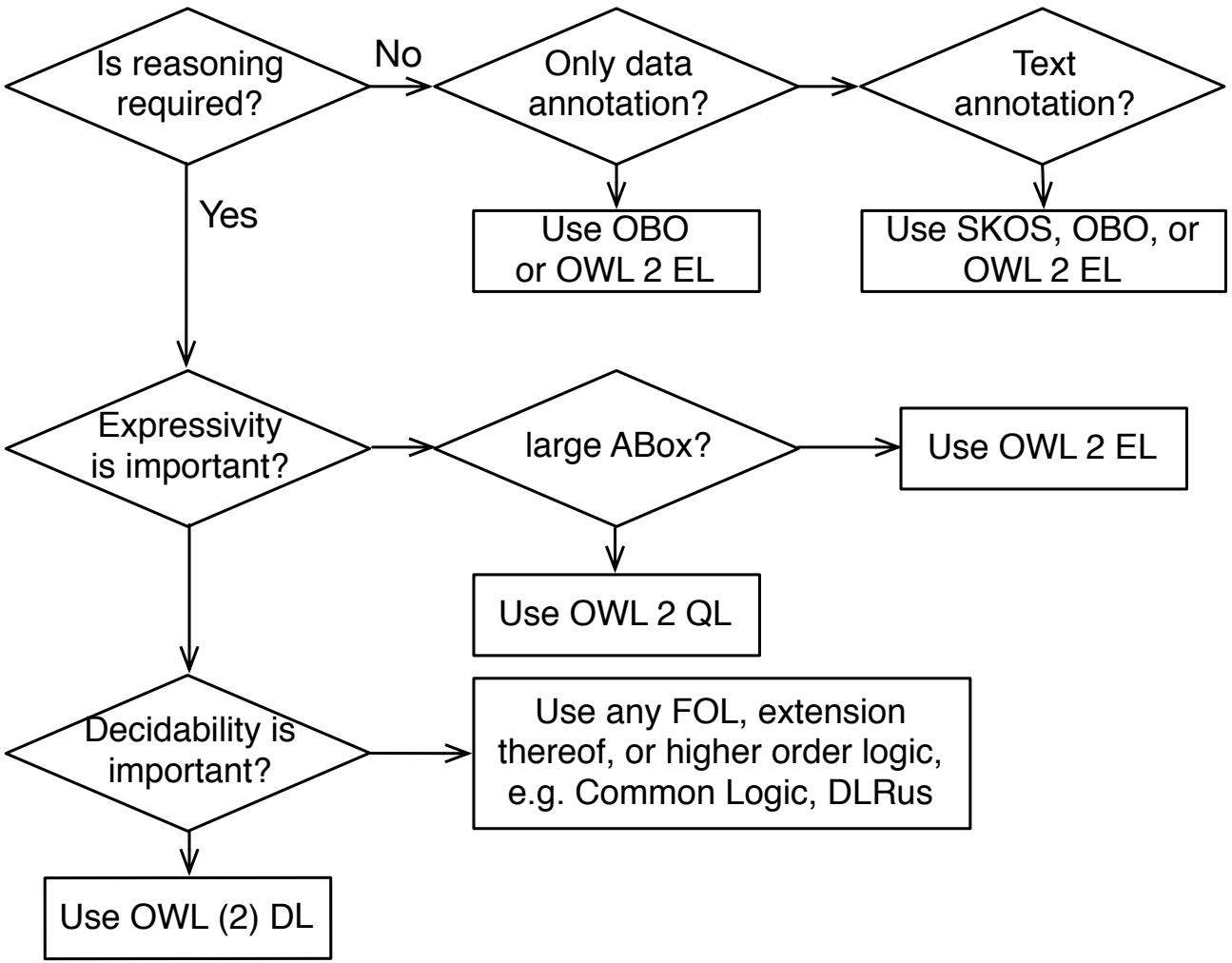
Micro-level methodologies

- Guidelines detailing how to go from informal to logic-based representations with instructions how to include the axioms and which ones are better than others
- To represent the formal and ontological details in an expressive ontology beyond just classes and some of their relationships so as to include guidance also for the axioms and ontological quality criteria
- Notably: OntoSpec and the "Ontology development 101"

More detailed steps (generalised from DiDOn)

1. Requirements analysis, regarding expressiveness (temporal, fuzzy, n-aries etc.), types of queries, reasoning services needed;
2. Design an ontology architecture, such as modular, and if so, in which way, distributed or not, etc.
3. Choose principal representation language and consider encoding peculiarities;

A few basic hints for choosing a language



More detailed steps (generalised from DiDOn)

4. Formalization, including:
   - examine and add the classes, object properties, constraints, rules taking into account the imported ontologies;
   - use an automated reasoner for debugging/anomalous deductions;
   - use ontological reasoning services for quality checks (OntoClean, RBox Compatibility);
   - add annotations;
5. Generate versions in other ontology languages, 'lite' versions, etc, if applicable;

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 with plugins
  - MoKi, Hozo, SWOOP, ...
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - 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 in the accompanying text and references*

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 with plugins
  - MoKi, Hozo, SWOOP, ...
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - 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 in the accompanying text and references*

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 with plugins
  - MoKi, Hozo, SWOOP, ...
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - 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 in the accompanying text and references*

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 with plugins
  - MoKi, Hozo, SWOOP, ...
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - 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 in the accompanying text and references*

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 with plugins
  - MoKi, Hozo, SWOOP, ...
  - RacerPro, RacerPorter. a.o.: sophisticated querying
  - 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 in the accompanying text and references*

Summary

1. Parameters and dependencies
2. Example methods: OntoClean and Debugging
   - Guidance for modelling: OntoClean
   - Debugging ontologies
3. Methodologies and tools
   - Macro-level methodologies
   - Micro-level methodologies
   - Tools