THE UNIVERSITY of LIVERPOOL

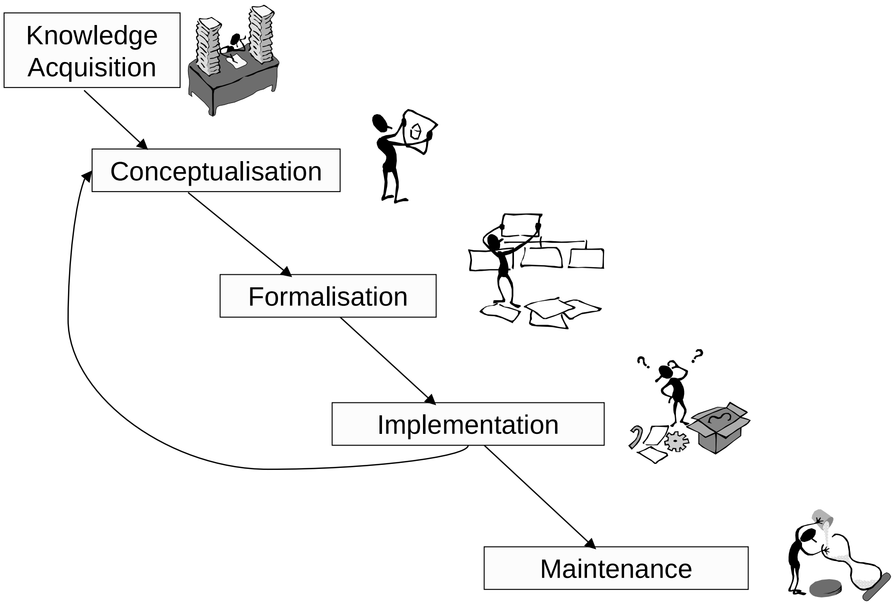
Ontologies and their applications in knowledge sharing

Valentina Tamma
Agent Applications, Research, and Technology group
Department of Computer Science
University of Liverpool

Outline

- → Knowledge sharing and reuse
- → Motivations for using ontologies
- → Ontologies and knowledge bases
- → Definitions
- → Ontologies and conceptualisations
- → Types of ontologies
- → Properties of ontologies

Problems in building KBS from scratch



Valentina Tamma Ontologies in knowledge sharing and reuse

A vision of the future

"We present a vision of the future in which the idea of knowledge sharing is commonplace. If this vision is realized,building a new system will rarely involve constructing a new knowledge base from scratch. Instead, the process of building a knowledge-based system will start by assembling reusable components. Portions of existing knowledge bases would be reused in constructing the new system, and special-purpose reasoners embodying problem-solving methods would similarly be brought in.

[…]

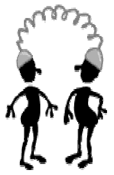
In our vision, the new system could interoperate with existing systems and pose queries to them to perform some of its reasoning. Furthermore, extensions to existing knowledge bases could be added to shared repositories, thereby expanding and enriching them."

Neches, R., Fikes, R.E, Finin, T., Gruber, T.R., Patil, R., Senator, T., and Swartout, W.R.: Enabling technologies for knowledge sharing. *AI Magazine*, 12, 3, pp. 36-56, 1991

Sharing and reuse

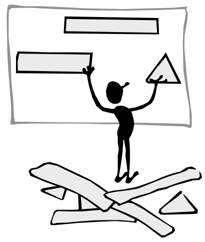
Sharing: different applications use the same resources

Reuse: components already built are used to build new applications



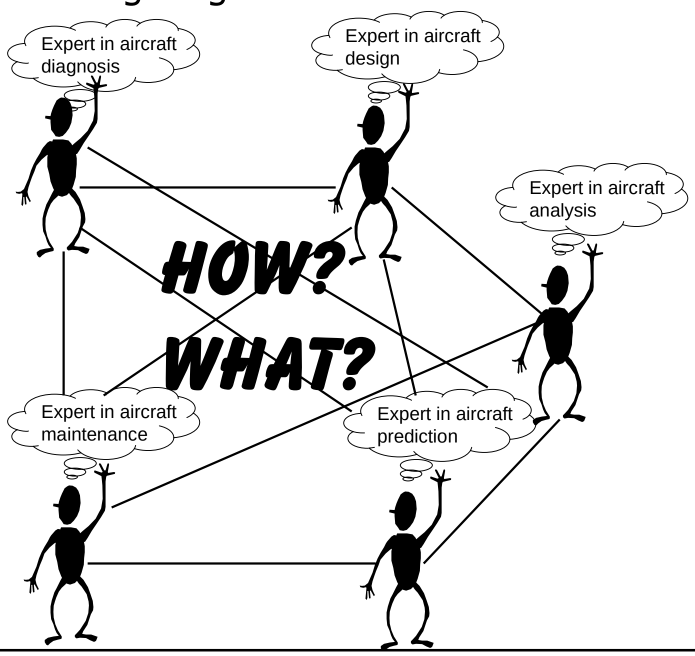
Areas

→ Software
→ Knowledge
→ Communications
→ Interfaces
→ ...



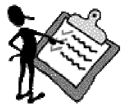
Advantages: Save money, save time, save resources

Sharing knowledge: agent networks



Reusing knowledge

A module satisfies some of the designer requirements:



Problems	Solution
1. Heterogeneity of knowledge representation formalisms 2. Heterogeneity of the implementation languages	Translators
3. Lexical problems 4. Semantic problems 5. Synonyms 6. Hidden assumptions	Ontologies
7. Loss of common sense knowledge	Ontologies + NLP + Machine Learning

The Knowledge Sharing Initiative

Interlingua Working Group → KIF: Knowedge Interchange Format
Fikes and Genesereth

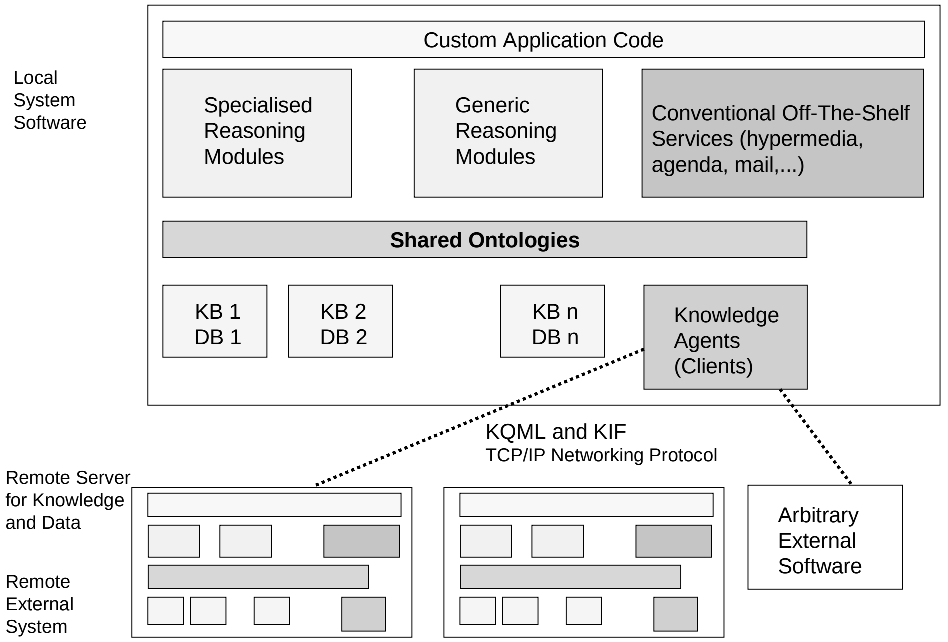
KR System Specification →
Swartout, Patel-Schneider

External Interface → KQML: Knowledge Query Manipulation Language
Finin

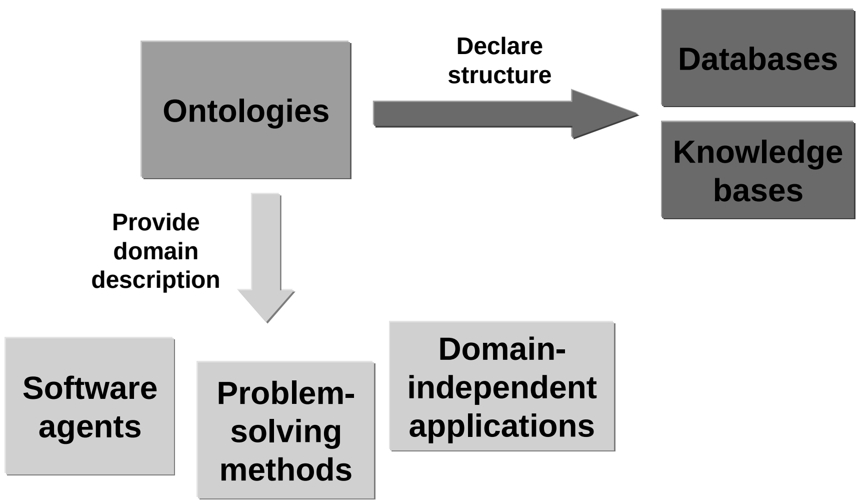
Shared, Reusable KB → Libraries of Ontologies
Gruber

Neches, R., Fikes, R.E, Finin, T., Gruber, T.R., Patil, R., Senator, T., and Swartout, W.R.: Enabling technologies for knowledge sharing. *AI Magazine*, 12, 3, pp. 36-56, 1991

Shared approach to KBS



An Ontology Is Often Just the Beginning



Why Develop an Ontology?

→ To share common understanding of the structure of information
  - among people
  - among software agents

→ To enable reuse of domain knowledge
  - to avoid "re-inventing the wheel"
  - to introduce standards to allow interoperability

More Reasons

→ To make domain assumptions explicit
  - easier to change domain assumptions (consider a genetics knowledge base)
  - easier to understand and update legacy data

→ To separate domain knowledge from the operational knowledge
  - re-use domain and operational knowledge separately (e.g., configuration based on constraints)

Different definitions of ontology

An Ontology is an explicit specification of a conceptualisation.
Gruber, T.R.: A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, pp. 199-220, 1993

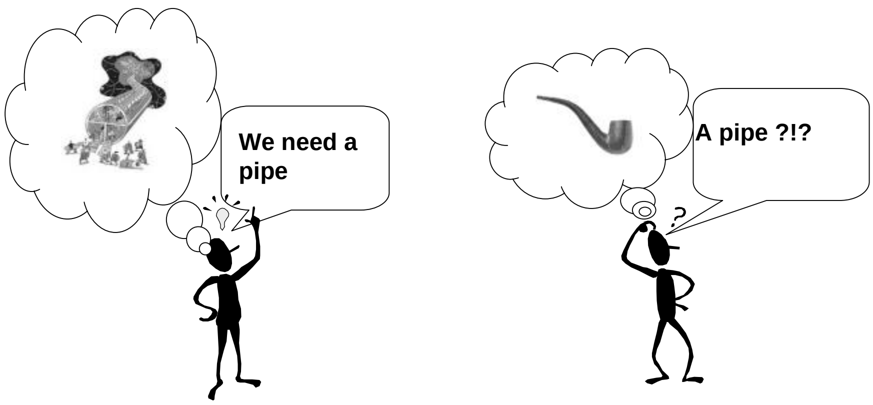
An Ontology defines the basic terms and relations comprising the vocabulary of a topic area, as well as the rules for combining terms and relations to define extensions to the vocabulary.
Neches, R., Fikes, R.E, Finin, T., Gruber, T.R., Patil, R., Senator, T., and Swartout, W.R.: Enabling technologies for knowledge sharing. *AI Magazine*, 12, 3, pp. 36-56, 1991

An Ontology provides the means for describing explicitly the conceptualisation behind the knowledge represented in a knowledge base.
Bernaras, A., Laresgoiti, I., and Corera, J.: Building and reusing ontologies for electrical network application. *Proceedings of the 12th ECAI*, pp. 298-302, 1996

An Ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualisation of the world. The intended models of a logical language using such vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualisation) by approximating these intended models.
Guarino, N.: Formal ontologies in information systems. *Proceedings of Formal Ontology and Information Systems*, 1998

Ontological commitment

Agreement on the meaning of the vocabulary used to share knowledge.



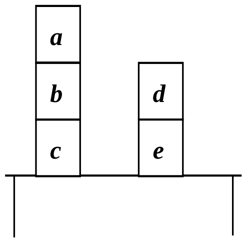
What is a conceptualisation?

<U, F, R>

U = Universe of discourse;

F = Functional basis set;

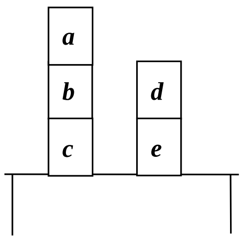
R= Relational basis set



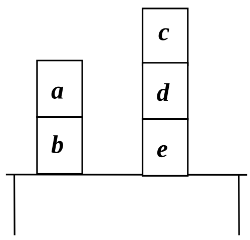
<{a, b, c, d, e }, {hat}, {on, above, clear, table }>

📖 Genesereth, M.R., and Nilsson, N.J.: *Logical foundation of Artificial Intelligence*. Morgan Kauffman, 1987

What is a conceptualisation?



Scene 1: blocks on a table

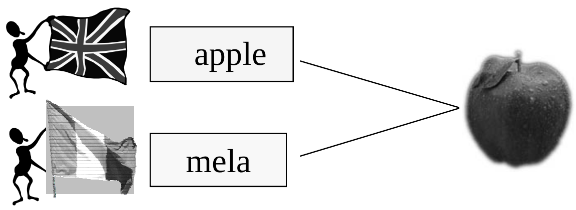


Scene 2: a different arrangement of blocks

The same conceptualisation?

What is a conceptualisation

- Conceptualisation: the formal structure of reality as perceived and organized by an agent, independently of:
  - the **vocabulary** used (i.e., the ***language*** used)
  - the actual occurrence of a specific ***situation***
- Different situations involving the same objects, described by different vocabularies, may share the same conceptualisation.



So what is an ontology then?

An ontology is a formal, explicit specification of a shared conceptualisation

formal: an ontology should be machine-readable

shared: an ontology captures consensual knowledge, that is not private to some individual, but accepted by a group

explicit: the types of concepts used, and the constraints on their use are explicitly defined

conceptualisation: an abstract model of some phenomenon in the world which identifies the relevant concepts of that phenomenon

Studer, R., Benjamins, V.R., and Fensel. D.: Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, 25, pp.161-197, 1998

Different levels of definitions

1. Ontology as philosophical discipline — Provides useful guidelines
2. Ontology as a specification of a conceptualisation
3. Ontology as an informal conceptual system
4. Ontology as a formal semantic account

(2–4: Knowledge Level)

5. Ontology as a representation of a conceptual system via a logical theory
6. Ontology as the vocabulary used by a logical theory
7. Ontology as a (meta-level) specification of a logical theory

(5–7: Symbolic Level)

Guarino, N., Giaretta, P.: Ontologies and Knowledge bases: Towards a Terminological Clarification. *Towards very Large Knowledge Bases: Knowledge Building and Knowledge Sharing,* Mars, N. (Ed). IOS Press, Amsterdam.

Components of an ontology

O = {*C, R, F, I, A*} + ontological commitment

Concepts = set of objects we want to talk about

Supervisor | Student | Professor

Relations = set of relations connecting those objects,

$R: C_1 \times C_2 \times \ldots \times C_n \rightarrow R(C_1, C_2, \ldots, C_n)$

Supervised-by: (Supervisor, Student)

Functions = set of functions defined on *C*

$F: C_1 \times C_2 \times \ldots \times C_{n-1} \rightarrow C_n \times C_{n+1} \times \ldots \times C_m$

Supervises: Supervisor → $Student_1$, $Student_2$, …, $Student_k$

Axioms = sentences that are always true

$\forall$ x Student(x) $\exists$ y Professor(y) : y = Supervisor(x)

Instances = the elements in the real world


Supervises:  → 

Different types of ontologies 1

Top-level ontologies: very general concepts or common sense knowledge. Domain independent;

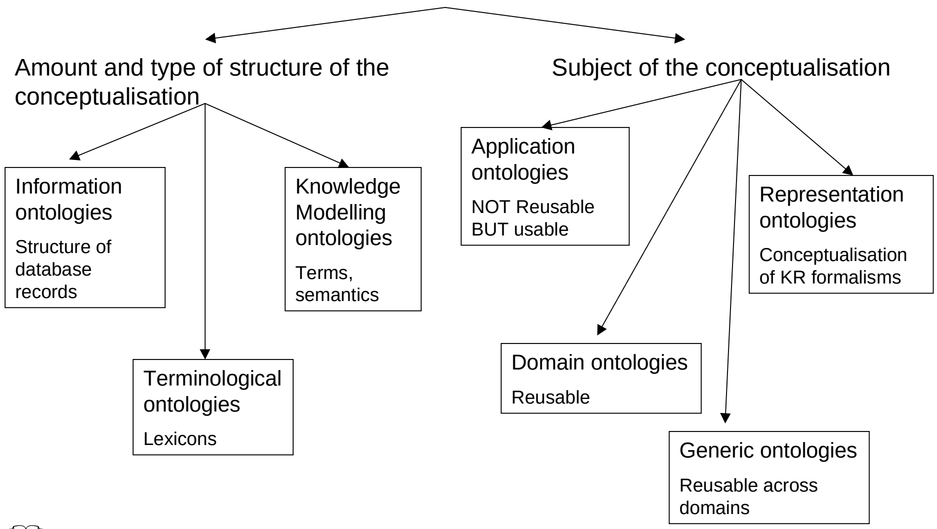
Domain ontologies: vocabulary related to a generic domain. Examples: medicine, physics;

Task ontologies: vocabulary related to a generic task or activity. Examples: diagnosis, selling;

Application ontologies: knowledge depending on both domain and task. It's usually the specialisation of both domain and task ontologies.

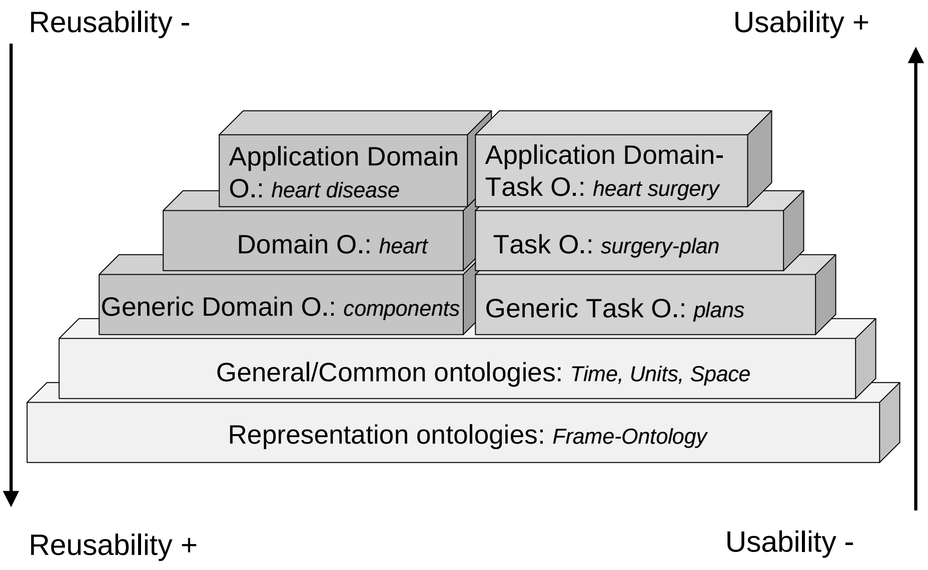
Guarino, N.: Formal ontologies in information systems. *Proceedings of Formal Ontology and Information Systems*, 1998

Different types of ontologies 2

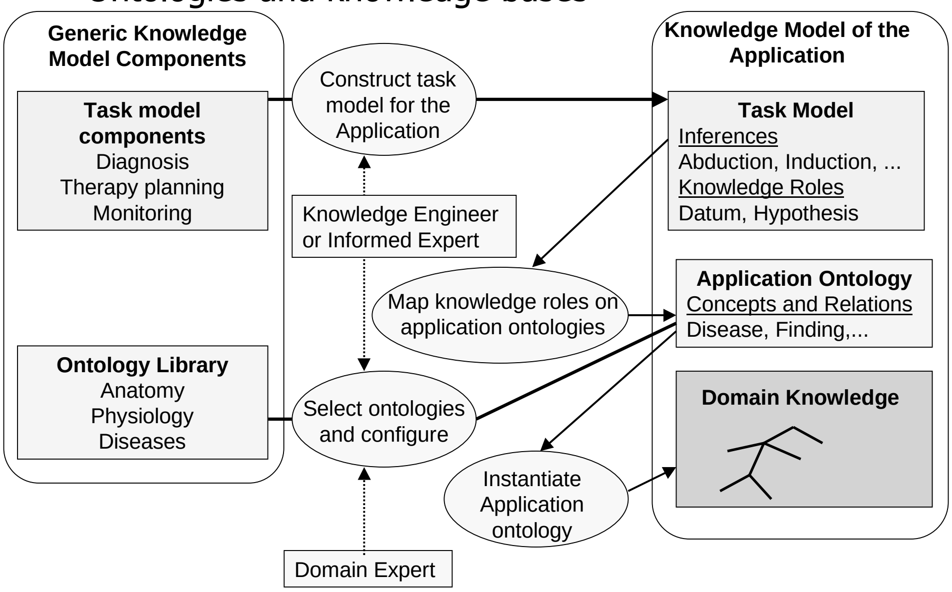


Van Heist, G., Schreiber, T., Wielinga, B: Using Explicit Ontologies in KBS. *International Journal of Human-Computer Studies*, 46, 2/3, pp. 183-192, 1997

Libraries of ontologies



Ontologies and knowledge bases



Van Heist, G., Schreiber, T., Wielinga, B: Using Explicit Ontologies in KBS. *International Journal of Human-Computer Studies*, 46, 2/3, pp. 183-192, 1997

Degree of formality of ontology representations

Highly informal: expressed in natural language;

Semi-informal: expressed in a restricted and structured form of natural language;

Semi-formal: expressed in artificial languages formally defined;

Rigorously formal: terms precisely defined with formal semantics, theorems, and proofs of desired properties;

Uschold, M., and Gruninger, M.: Ontologies: principles, methods, and applications. *Knowledge Engineering Review,* 11, 2, pp. 93-155, 1996

Lightweight and heavyweight ontologies

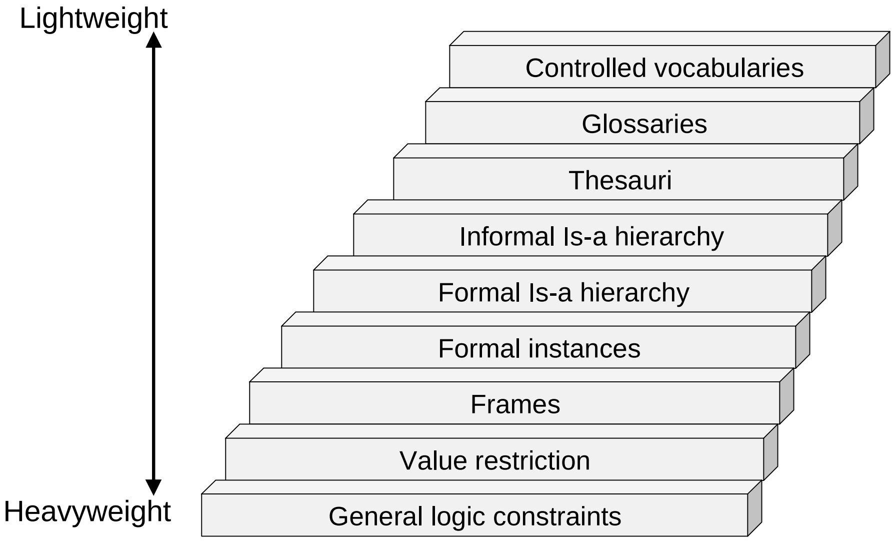
Lightweight Ontologies :

- **Include Concepts with properties and Taxonomies;**
- **Do not include Axioms and constraints;**

Heavyweight Ontologies :

- **Include all the components;**
- **Excellent, if they have a lot of axioms;**

Levels of ontological depth



Lassila, O., and McGuinness, D.: The role of frame-based representation on the semantic web. *Electronic transactions on artificial intelligence (ETAI): journal area the semantic web*, To appear.

Ontology Examples

- → Taxonomies on the Web
  - – Yahoo! categories
- → Catalogs for on-line shopping
  - – Amazon.com product catalog
- → Domain-specific standard terminology
  - – Unified Medical Language System (UMLS)
  - – UNSPSC - terminology for products and services

Properties applicable to ontologies

Mandatory:

1. Finite, controlled, extensible vocabulary;
2. Unambiguous interpretation of classes and term relationships;
3. Strict hierarchical subclass relationship between classes;

Typical:

4. Property specification on a per-class basis;
5. Individual inclusion in the ontology;
6. Value restriction on a per-class basis;

Desirable:

7. Specification of disjoint classes;
8. Specification of arbitrary logic relationships between terms;

Ontology Engineering

A large part of this part of the tutorial is based on:

"Ontology Development 101: A Guide to Creating Your First Ontology"

by Natalya F. Noy and Deborah L. McGuinness

"Knowledge sharing and reuse"

by Asuncion Gomez-Perez

Outline

- → What is ontological engineering
- → Methodologies
- → Step-By-Step: Developing an ontology
- → Going deeper: Common problems and solutions
- → Ontologies in the Semantic Web languages
- → Current research issues in ontology engineering

What Is "Ontology Engineering"?

Ontology Engineering:
Defining terms in the domain and relations among them

- Defining concepts in the domain (classes)
- Arranging the concepts in a hierarchy (subclass-superclass hierarchy)
- Defining which attributes and properties (slots) classes can have and constraints on their values
- Defining individuals and filling in slot values

Methodologies

- **Methodologies for building ontologies**
  - **Enterprise Methodology**
  - **TOVE Methodology**
  - **METHONTOLOGY**
- **Methodology for reengineering ontologies**
- **Methodologies for collaborative construction of ontologies**

Uschold and Gruninger's (enterprise) methodology

1. Identify Purpose and Scope
2. Building the ontology
   - Ontology Capture
   - Ontology Coding
   - Integrating existing ontologies
3. Evaluation
4. Documentation
5. Guidelines for each phase

Ontology Capture:

- Identify key concepts and relationships
- Produce unambiguous text definitions
- Identify terms to refer to such concepts and relations

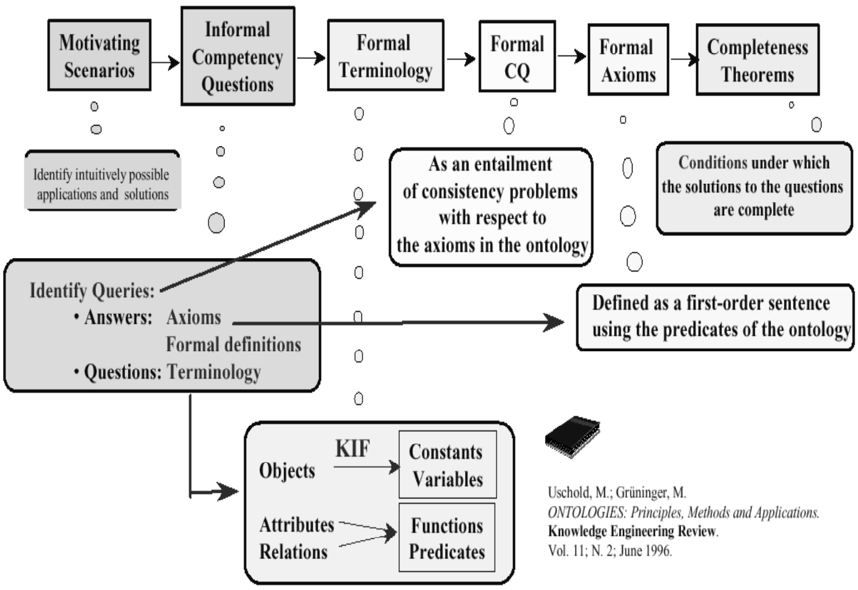
Ontology Coding:

- Commit to a meta-ontology
- Choose a representation language
- Write the code

Integrating existing ontologies:

- How and whether to reuse ontologies that already exist

TOVE Methodology



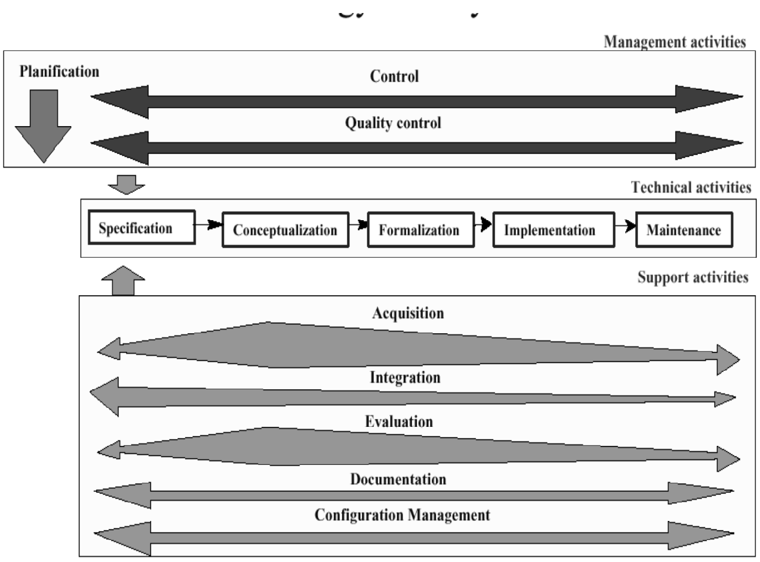
METHONTOLOGY: the framework

Development Process: Which activities

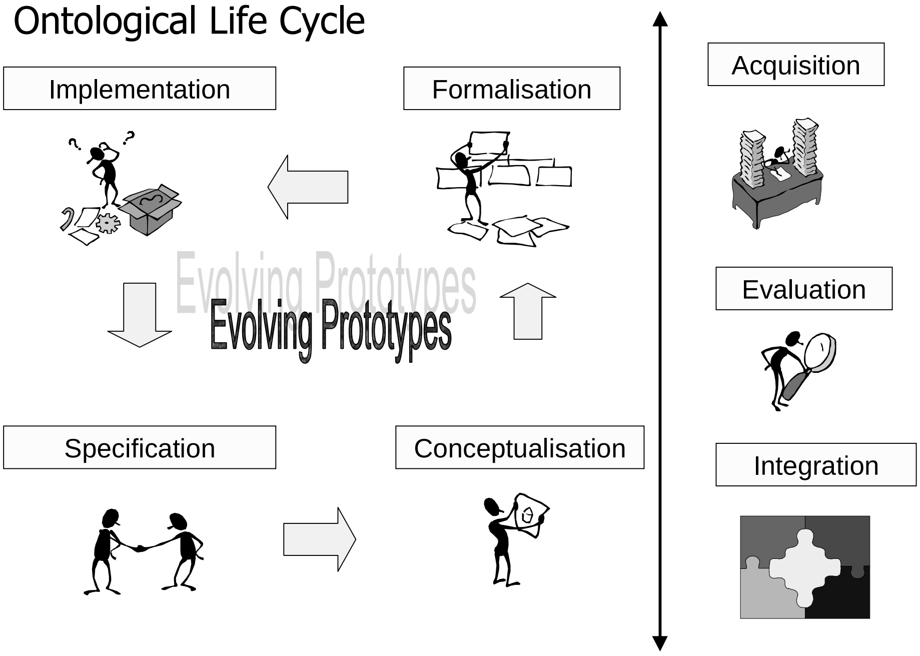
Life Cycle: Order of activities

Methodologies: How to carry out the activities

METHONTOLOGY: the lifecycle



Ontological Life Cycle



Principles for building ontologies 1

→ Clarity: to communicate the intended meaning of defined terms;

→ Coherence: to sanction inferences that are consistent with definitions;

→ Extendibility: To anticipate the user of a shared vocabulary;

→ Minimal Encoding Bias: To be independent of the symbolic level;

→ Minimal Ontological Commitments: To make as few claims as possible about the world;

📖 Gruber, T.R.: A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5, pp. 199-220, 1993

Principles for building ontologies 2

→ Not ‘over commit’ on representational choices: an ontology is a pre-package design of a KB;

→ Extensible: by adding domain-specific, intermediate or upper level concepts;

→ Extended depending on needs identified during actual use: in a consistent ad coherent way;

→ Not to be ‘stovepipes’: to allow vertical and horizontal integration;

→ Using organising principles: Linguistics, conceptual clustering, ...

📖 Swartout, B., Patil, R., Knight, K., Russ, T.: Toward distributed use of large scale ontologies. *Ontological Engineering, AAAI’97 Spring Symposium Series*, pp. 138-148, 1997

Principles for building ontologies 3

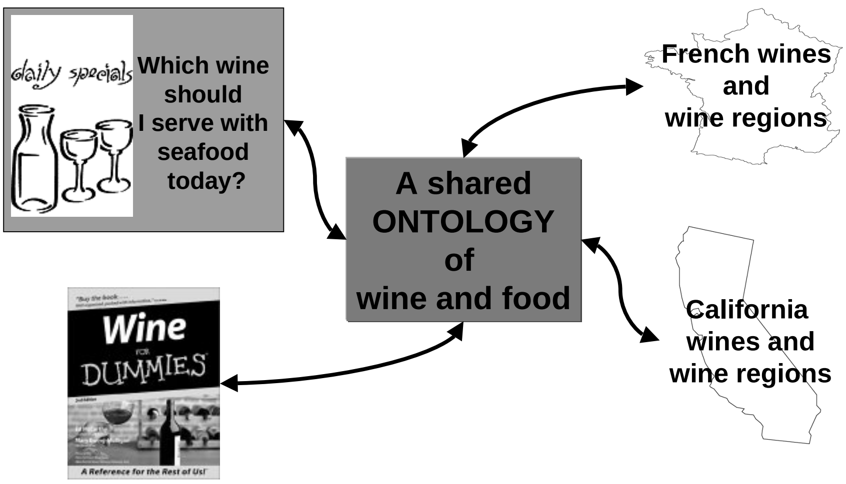
Modularisation:

To minimise coupling between modules;

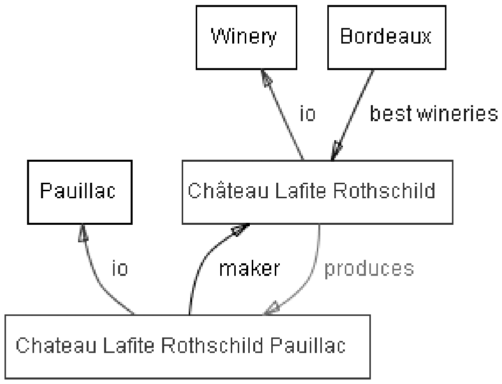
Hierarchical organisation:

To maximise cohesion within a module or a sub-ontology;

Bernaras, A., Laresgoiti, I., and Corera, J.: Building and reusing ontologies for electrical network application. *Proceedings of the 12th ECAI*, pp. 298-302, 1996

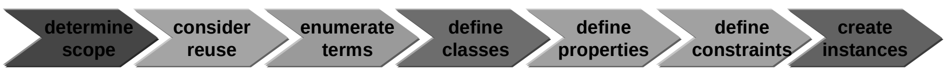


Our domain: Wines and Wineries

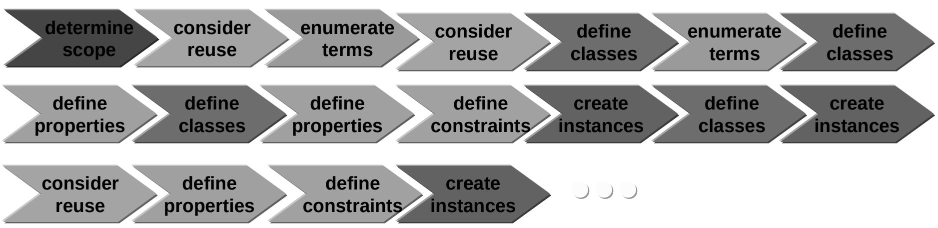


Ontology-Development Process

In this tutorial:



In reality - an iterative process:



Ontology Engineering versus Object-Oriented Modelling

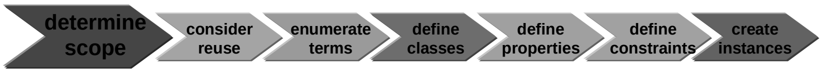
An ontology
- → reflects the structure of the world
- → is often about structure of concepts
- → actual physical representation is not an issue

An OO class structure
- → reflects the structure of the data and code
- → is usually about behavior (methods)
- → describes the physical representation of data (long int, char, etc.)

Preliminaries - Tools

- → All screenshots are from Protégé-2000, which:
  - – is a graphical ontology-development tool
  - – supports a rich knowledge model
  - – is open-source and freely available (http://protege.stanford.edu)
- → Some other available tools:
  - – Ontolingua and Chimaera
  - – OntoEdit
  - – OilEd
  - – WebOde

Determine Domain and Scope



- → What is the domain that the ontology will cover?
- → For what we are going to use the ontology?
- → For what types of questions the information in the ontology should provide answers (competency questions)?

Answers to these questions may change during the lifecycle

Competency Questions

- → Which wine characteristics should I consider when choosing a wine?
- → Is Bordeaux a red or white wine?
- → Does Cabernet Sauvignon go well with seafood?
- → What is the best choice of wine for grilled meat?
- → Which characteristics of a wine affect its appropriateness for a dish?
- → Does a flavor or body of a specific wine change with vintage year?
- → What were good vintages for Napa Zinfandel?

Grüninger, M., and Fox, M.S.: Methodology for the design and evaluation of ontologies. *Proceedings of the IJCAI'95 workshop on Basic Ontological Issues.*

Consider Reuse

→ Why reuse other ontologies?
- to save the effort
- to interact with the tools that use other ontologies
- to use ontologies that have been validated through use in applications

What to Reuse?

→ Ontology libraries
- DAML ontology library (www.daml.org/ontologies)
- Ontolingua ontology library (www.ksl.stanford.edu/software/ontolingua/)
- Protégé ontology library (protege.stanford.edu/plugins.html)

→ Upper ontologies
- IEEE Standard Upper Ontology (suo.ieee.org)
- Cyc (www.cyc.com)

What to Reuse? (II)

- → General ontologies
  - – DMOZ (www.dmoz.org)
  - – WordNet (www.cogsci.princeton.edu/~wn/)
- → Domain-specific ontologies
  - – UMLS Semantic Net
  - – GO (Gene Ontology) (www.geneontology.org)

Valentina Tamma
Ontologies in knowledge sharing and reuse

Enumerate Important Terms

- → What are the terms we need to talk about?
- → What are the properties of these terms?
- → What do we want to say about the terms?

Enumerating Terms - The Wine Ontology

wine, grape, winery, location,
wine color, wine body, wine flavor, sugar content
white wine, red wine, Bordeaux wine
food, seafood, fish, meat, vegetables, cheese

Define Classes and the Class Hierarchy

→ A class is a concept in the domain
- a class of wines
- a class of wineries
- a class of red wines

→ A class is a collection of elements with similar properties

→ Instances of classes
- a glass of California wine you'll have for lunch
- a glass of Locorotondo wine you'll have for dinner

Valentina Tamma Ontologies in knowledge sharing and reuse

Class Inheritance

→ Classes usually constitute a taxonomic hierarchy (a subclass-superclass hierarchy)

→ A class hierarchy is usually an IS-A hierarchy:

an instance of a subclass is an instance of a superclass

→ If you think of a class as a set of elements, a subclass is a subset

Valentina Tamma Ontologies in knowledge sharing and reuse

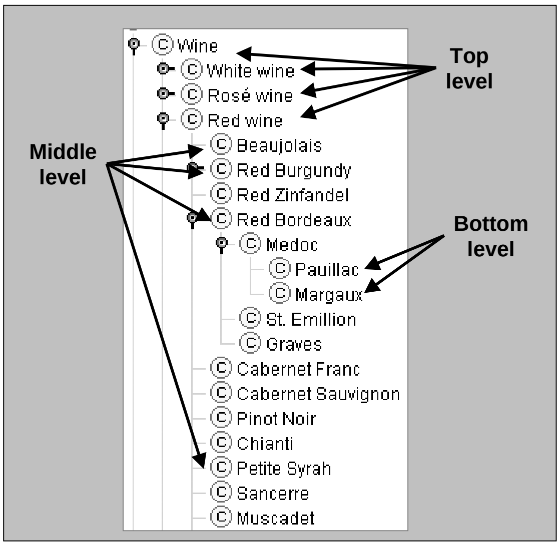
Class Inheritance - Example

→ Apple is a subclass of Fruit
Every apple is a fruit

→ Red wines is a subclass of Wine
Every red wine is a wine

→ Chianti wine is a subclass of Red wine
Every Chianti wine is a red wine

Levels in the Hierarchy



Modes of Development

→ top-down – define the most general concepts first and then specialize them

→ bottom-up – define the most specific concepts and then organize them in more general classes

→ middle-out – define the more salient concepts first and then generalize and specialize them

Uschold, M., and Grüninger, M.: **ONTOLOGIES:** Principles, Methods, and Applications. *Knowledge Engineering Review*, 11, 2, June 1996

Documentation

→ Classes (and slots) usually have documentation
  – Describing the class in natural language
  – Listing domain assumptions relevant to the class definition
  – Listing synonyms

→ Documenting classes and slots is as important as documenting computer code!

Define Properties of Classes – Slots

→ Slots in a class definition describe attributes of instances of the class and relations to other instances

Each wine will have color, sugar content, producer, etc.

Properties (Slots)

→ Types of properties

- “intrinsic” properties: flavor and color of wine
- “extrinsic” properties: name and price of wine
- parts: ingredients in a dish. Special relation *part-of*
- relations to other objects: producer of wine (winery)

→ Simple and complex properties

- simple properties (attributes): contain primitive values (strings, numbers)
- complex properties: contain (or point to) other objects (e.g., a winery instance) . They models relations.

Valentina Tamma Ontologies in knowledge sharing and reuse

Slots for the Class Wine

Template Slots

Name	Type	Cardinality	Other Facets
S body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
S color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
S flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
S grape	Instance	multiple	classes={Wine grape}
S maker	Instance	single	classes={Winery}
S name	String	single	
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

(in Protégé-2000 based on OKBC: Open Knowledge Base Connectivity)

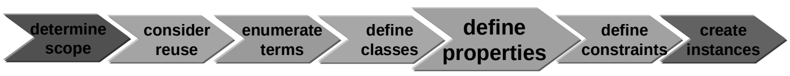
THE UNIVERSITY of LIVERPOOL

Ontologies and their applications in knowledge sharing

Valentina Tamma
Agent Applications, Research, and Technology group
Department of Computer Science
University of Liverpool

Recap

- ✓ Knowledge sharing and reuse
- ✓Ontologies: definitions, types and properties
- ✓Ontological engineering: methodologies
- ✓ Building your first ontology step-by-step



- ✓ SW Languages
- ✓ Hot research issues!

Slot and Class Inheritance

→ A subclass usually inherits all the slots from the superclass

If a wine has a name and flavour, a red wine also has a name and flavour

→ If a class has multiple superclasses, it inherits slots from all of them. This may create conflicts, but conflicting information might not be a problem in ontologies. It is an important issue in knowledge bases, where instances are considered.

Port is both a dessert wine and a red wine. It inherits “sugar content: high” from the former and “colour: red” from the latter

Port is both a white wine and a red wine. It inherits “colour:white” from the former and “colour: red” from the latter

My-Port (instance-of: Port) MUST inherit either “colour: white” or “colour: red” but not both.

Property Constraints

→ Property constraints (facets) describe or limit the set of possible values for a slot

The name of a wine is a string

The wine producer is an instance of Winery

A winery has exactly one location

Facets for Slots at the Wine Class

Template Slots

Name	Type	Cardinality	Other Facets
S body	Symbol	single	allowed-values={FULL,MEDIUM,LIGHT}
S color	Symbol	single	allowed-values={RED,ROSÉ,WHITE}
S flavor	Symbol	single	allowed-values={DELICATE,MODERATE,STRONG}
S grape	Instance	multiple	classes={Wine grape}
S maker	Instance	single	classes={Winery}
S name	String	single	
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Common Facets (from OKBC)

→ Slot cardinality – the number of values a slot has

→ Slot value type – the type of values a slot has

→ Minimum and maximum value – a range of values for a numeric slot

→ Default value – the value a slot has unless explicitly specified otherwise

Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D., and Rice, J.P.: OKBC: A programmatic foundation for knowledge base interoperability. *Proceedings of the AAAI 98*, pp. 600-607, 1998.

Common Facets: Slot Cardinality

- → Cardinality
  - – Cardinality N means that the slot must have N values
- → Minimum cardinality
  - – Minimum cardinality 1 means that the slot must have a value (required)
  - – Minimum cardinality 0 means that the slot value is optional
- → Maximum cardinality
  - – Maximum cardinality 1 means that the slot can have at most one value (single-valued slot)
  - – Maximum cardinality greater than 1 means that the slot can have only one value (multiple-valued slot)

Common Facets: Value Type

- → String:
  a string of characters ("Château Lafite")
- → Number:
  an integer or a float (15, 4.5)
- → Boolean:
  a true/false flag
- → Enumerated type:
  a list of allowed values (high, medium, low)
- → Complex type:
  an instance of another class
  - – Specifies the class to which the instances belong

  The Wine class is the value type for the slot "produces" at the Winery class

Domain and Range of Slot

→ Domain of a slot:
the class (or classes) that have the slot

- More precisely: class (or classes) instances of which can have the slot

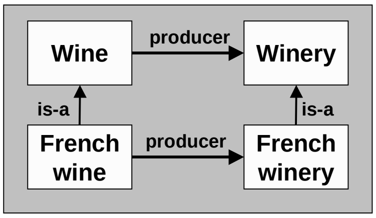
→ Range of a slot:
the class (or classes) to which slot values belong

Facets and Class Inheritance

→ A subclass inherits all the slots from the superclass

→ A subclass can override the facets to "narrow" the list of allowed values

- Make the cardinality range smaller
- Replace a class in the range with a subclass

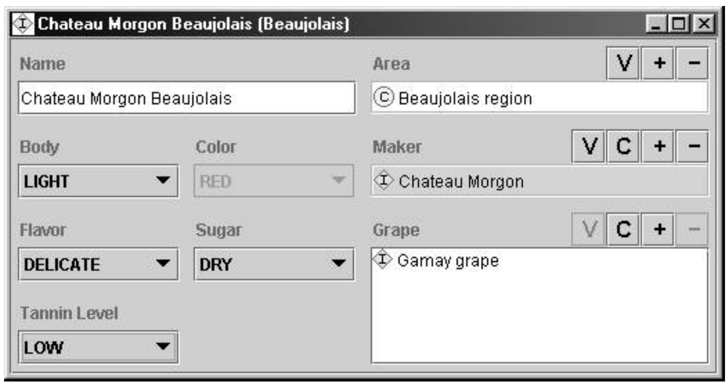


Create Instances

→ Create an instance of a class
  – The class becomes a direct type of the instance
  – Any superclass of the direct type is a type of the instance

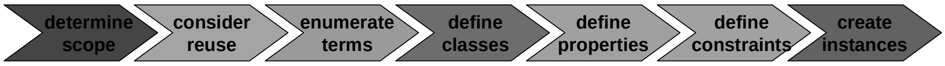
→ Assign slot values for the instance frame
  – Slot values should conform to the facet constraints
  – Knowledge-acquisition tools often check that

Creating an Instance: Example

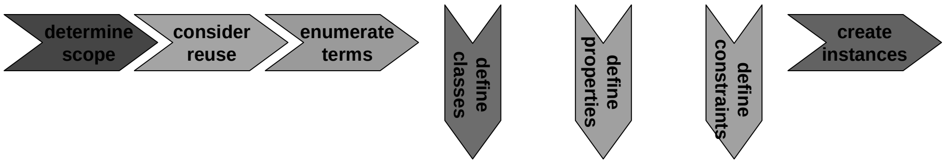


Going Deeper

→ Breadth-first coverage



- Depth-first coverage



Defining Classes and a Class Hierarchy

→ The things to remember:
- There is no single correct class hierarchy
- But there are some guidelines

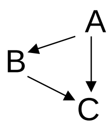
→ The question to ask:

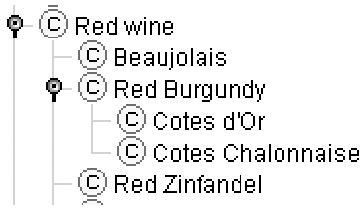
"Is each instance of the subclass an instance of its superclass?"

Transitivity of the Class Hierarchy

→ The is-a relationship is transitive:

B is a subclass of A
C is a subclass of B
C is a subclass of A

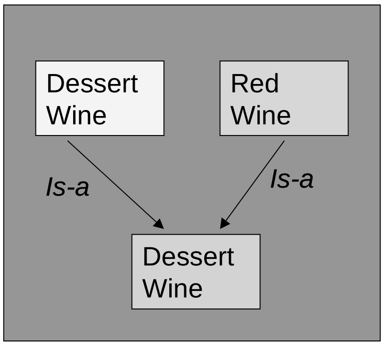




→ A direct superclass of a class is its "closest" superclass

Multiple Inheritance

- → A class can have more than one superclass
- → A subclass inherits slots and facet restrictions from all the parents
- → Different systems resolve conflicts differently



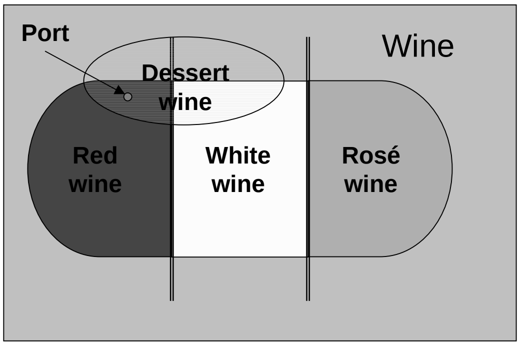
Disjoint Classes

→ Classes are disjoint if they cannot have common instances

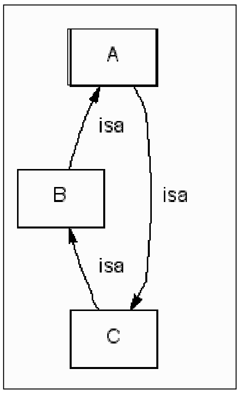
→ Disjoint classes cannot have any common subclasses either

Red wine, White wine, Rosé wine are disjoint

Dessert wine and Red wine are not disjoint



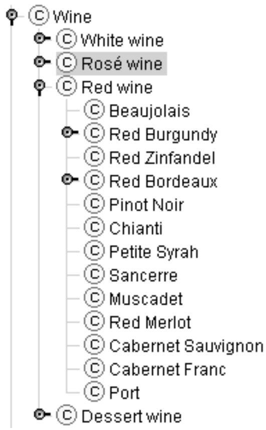
Avoiding Class Cycles



→ Danger of multiple inheritance: cycles in the class hierarchy

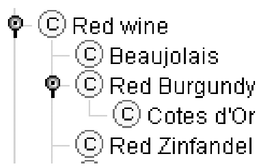
→ Classes A, B, and C have equivalent sets of instances
- By many definitions, A, B, and C are thus equivalent

Siblings in a Class Hierarchy



- → All the siblings in the class hierarchy must be at the same level of generality
- → Compare to section and subsections in a book

The Perfect Family Size: not too few...

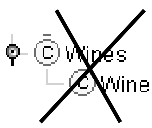


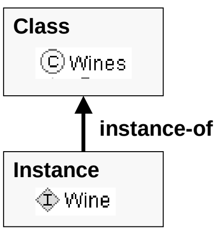
- → If a class has only one child, there may be a modelling problem
- → If the only Red Burgundy we have is Côtes d'Or, why introduce the subhierarchy?
- → Compare to bullets in a bulleted list

The Perfect Family Size: not too many...

- → If a class has more than a dozen children, additional subcategories may be necessary
- → However, if no natural classification exists, the long list may be more natural

Single and Plural Class Names





- → A “wine” is not a kind-of “wines”
- → A wine is an instance of the class Wines
- → Class names should be either
  - all singular
  - all plural

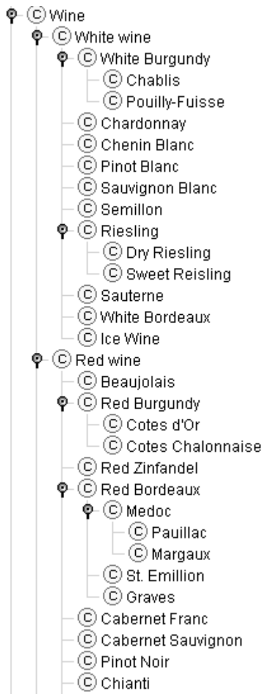
Classes and Their Names

→ Classes represent concepts in the domain, not their names

→ The class name can change, but it will still refer to the same concept

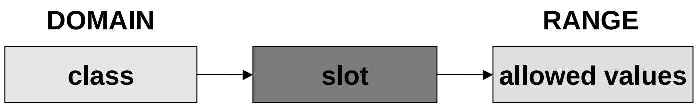
→ Synonym names for the same concept are not different classes
  - Many systems allow listing synonyms as part of the class definition

Valentina Tamma Ontologies in knowledge sharing and reuse



A Completed Hierarchy of Wines

Back to the Slots: Domain and Range

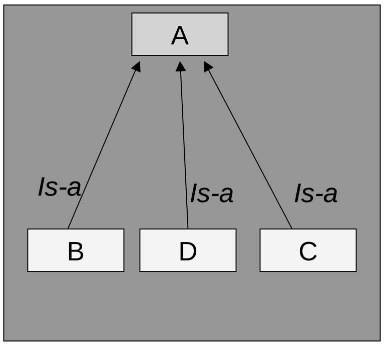


- → When defining a domain or range for a slot, find the most general class or classes
- → Consider the flavor slot
  - – Domain: Red wine, White wine, Rosé wine
  - – Domain: Wine
- → Consider the produces slot for a Winery:
  - – Range: Red wine, White wine, Rosé wine
  - – Range: Wine

Valentina Tamma
Ontologies in knowledge sharing and reuse

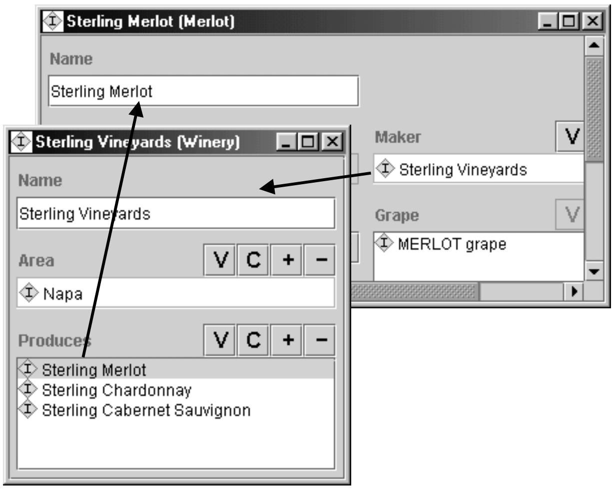
Defining Domain and Range

- → A class and a superclass – replace with the superclass
- → All subclasses of a class – replace with the superclass
- → Most subclasses of a class – consider replacing with the superclass



Inverse Slots

Maker and
Producer
are inverse slots



Inverse Slots (II)

→ Inverse slots contain redundant information, but
- Allow acquisition of the information in either direction
- Enable additional verification
- Allow presentation of information in both directions

→ The actual implementation differs from system to system
- Are both values stored?
- When are the inverse values filled in?
- What happens if we change the link to an inverse slot?

Default Values

- → Default value – a value the slot gets when an instance is created
- → A default value can be changed
- → The default value is a common value for the slot, but is not a required value
- → For example, the default value for wine body can be FULL

Limiting the Scope

- → An ontology should not contain all the possible information about the domain
  - – No need to specialize or generalize more than the application requires
  - – No need to include all possible properties of a class
    - Only the most salient properties
    - Only the properties that the applications require

Limiting the Scope (II)

→ Ontology of wine, food, and their pairings probably will not include
  - Bottle size
  - Label color
  - My favorite food and wine

→ An ontology of biological experiments will contain
  - Biological organism
  - Experimenter

→ Is the class Experimenter a subclass of Biological organism?

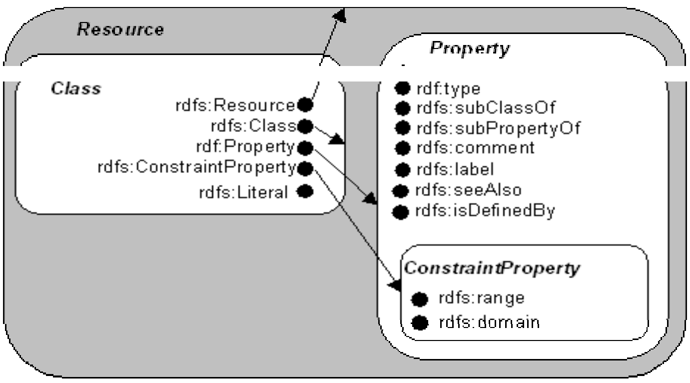
Ontologies and the SW Languages

→ Most Semantic Web languages are designed explicitly for representing ontologies
  - RDF Schema
  - DAML+OIL
  - SHOE
  - XOL
  - XML Schema

SW Languages

→ The languages differ in their
- syntax
  - We are not concerned with it here – An ontology is a conceptual representation
- terminology
  - Class-concept
  - Instance-object
  - Slot-property
- expressivity
  - What we can express in some languages, we cannot express in others
- semantics
  - The same statements may mean different things in different languages

RDF and RDF Schema Classes



RDF Schema Specification 1.0 (http://www.w3.org/TR/2000/CR-rdf-schema-20000327/)

RDF(S) Terminology and Semantics

→ Classes and a class hierarchy
  – All classes are instances of rdfs:Class
  – A class hierarchy is defined by rdfs:subClassOf
→ Instances of a class
  – Defined by rdf:type
→ Properties
  – Properties are global:
    A property name in one place is the same as the property name in another (assuming the same namespace)
  – Properties form a hierarchy, too (rdfs:subPropertyOf)

Property Constraints in RDF(S)

→ Cardinality constraints
  – No explicit cardinality constraints
  – Any property can have multiple values
→ Range of a property
  – a property can have only one range
→ Domain of a property
  – a property can have more than one domain (can be attached to more than one class)
→ No default values

DAML+OIL: Classes And a Class Hierarchy

- → Classes
  - Each class is an instance of daml:Class
- → Class hierarchy
  - Defined by rdfs:subClassOf
- → More ways to specify organization of classes
  - Disjointness (daml:disjointWith)
  - Equivalence (daml:sameClassAs)
- → The class hierarchy can be computed from the properties of classes

More Ways To Define a Class in DAML+OIL

- → Union of classes

  A class Person is a union of classes Male and Female
- → Restriction on properties

  A class Red Thing is a collection of things with color: Red
- → Intersection of classes

  A class Red Wine is an intersection of Wine and Red Thing
- → Complement of a class

  Carnivores are all the animals that are not herbivores
- → Enumeration of elements

  A class Wine Color contains the following instances: red, white, rosé

Property Constraints in DAML+OIL

- → Cardinality
  - – Minimum, maximum, exact cardinality
- → Range of a property
  - – A property range can include multiple classes: the value of a property must be an instance of each of the classes
  - – Can specify explicit union of classes if need different semantics
- → Domain of a property – same as range
- → No default values

Research Issues in Ontology Engineering

- → Content generation
- → Analysis and evaluation
- → Maintenance
- → Ontology languages
- → Tool development

Content: Top-Level Ontologies

→ What does “top-level” mean?
  – Objects: tangible, intangible
  – Processes, events, actors, roles
  – Agents, organizations
  – Spaces, boundaries, location
  – Time

→ IEEE Standard Upper Ontology effort
  – Goal: Design a single upper-level ontology
  – Process: Merge upper-level of existing ontologies

Content: Knowledge Acquisition

→ Knowledge acquisition is a bottleneck

→ Sharing and reuse alleviate the problem

→ But we need automated knowledge acquisition techniques
  – Linguistic techniques: ontology acquisition from text
  – Machine-learning: generate ontologies from structured documents (e.g., XML documents)
  – Exploiting the Web structure: generate ontologies by crawling structured Web sites
  – Knowledge-acquisition templates: experts specify only part of the knowledge required

Analysis

- → Analysis: semantic consistency
  - Violation of property constraints
  - Cycles in the class hierarchy
  - Terms which are used but not defined
  - Interval restrictions that produce empty intervals (min > max)
- → Analysis: style
  - Classes with a single subclass
  - Classes and slots with no definitions
  - Slots with no constraints (value type, cardinality)
- → Tools for automated analysis
  - Chimaera (Stanford KSL)
  - DAML validator

Evaluation

- → One of the hardest problems in ontology design
- → Ontology design is subjective
- → What does it mean for an ontology to be correct (objectively)?
- → The best test is the application for which the ontology was designed

Ontology Maintenance

→ Ontology merging
  – Having two or more overlapping ontology, create a new one
→ Ontology mapping
  – Create a mapping between ontologies
→ Versioning and evolution
  – Compatibility between different versions of the same ontology
  – Compatibility between versions of an ontology and instance data

Ontology Languages

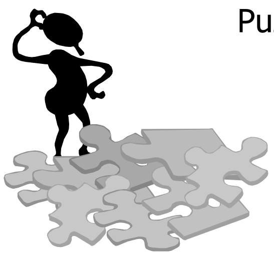
→ What is the “right” level of expressiveness?
→ What is the “right” semantics?
→ When does the language make “too many” assumptions?

Ontology-Development Tools

- → Support for various ontology language (knowledge interchange)
- → Expressivity
- → Usability
  - – More and more domain experts are involved in ontology development
  - – Multiple parentheses and variables will no longer do

Valentina Tamma
Ontologies in knowledge sharing and reuse

And that’s enough for now!!!



Puzzled? Interested in more details?

valli@csc.liv.ac.uk