



A tutorial report for SENG 609.22

**Agent Based Software Engineering
Course Instructor: Dr. Behrouz H. Far**

Ontology Development

Qing He

1 Why develop an ontology?

In recent years the development of ontologies—explicit formal specifications of the terms in the domain and relations among them have become common on the World-Wide Web. WWW Consortium (W3C) is developing the Resource Description Framework, a language for encoding knowledge on Web pages to make it understandable to electronic agents searching for information. **The Defense Advanced** Research Projects Agency (DARPA), in conjunction with the W3C, is developing DARPA Agent Markup Language (DAML) by extending RDF with more expressive constructs aimed at facilitating agent interaction on the Web. Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED and the semantic network of the Unified Medical Language System. Broad general-purpose ontologies are emerging as well. For example, the United Nations Development

Program and Dun & Bradstreet combined their efforts to develop the UNSPSC ontology which provides terminology for products and services.

On the other hand, why would someone want to develop an ontology? Some of the reasons are:

- To share common understanding of the structure of information among people or software agents
- To enable reuse of domain knowledge
- To make domain assumptions explicit
- To separate domain knowledge from the operational
- To analyze domain knowledge

Among the reasons, sharing common understanding of the structure of information among people or software agents is one of the more common goals in developing ontologies. For example, suppose several different Web sites contain medical information or provide medical e-commerce services. If these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

2 What is in an ontology

An **ontology** is a formal explicit description of concepts in a domain of discourse (**classes**), properties of each concept describing various features and attributes of the concept (**slots**), and restrictions on slots (**facets**). An ontology together with a set of individual **instances** of classes constitutes a **knowledge base**. In reality, there is a fine line where the ontology ends and the knowledge base begins. Classes are the focus of most ontologies. Classes describe concepts in the domain and Slots describe properties of classes and instances

In practical terms, developing an ontology includes:

- defining classes in the ontology,
- arranging the classes in a taxonomic (subclass–superclass) hierarchy,
- defining slots and describing allowed values for these slots,
- filling in the values for slots for instances.

We can then create a knowledge base by defining individual instances of these classes filling in specific slot value information and additional slot restrictions.

3 How to develop an ontology?

It is difficult to say there is one “correct” way or methodology for developing ontologies. However, we would like to emphasize some fundamental rules in ontology design. These rules may seem rather dogmatic and can help to make design decisions in many cases.

- There is no one correct way to model a domain— there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.
- Ontology development is necessarily an iterative process.
- Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

As a result, we will almost certainly need to revise the initial ontology. This process of iterative design will likely continue through the entire lifecycle of the ontology.

1) Determine the domain and scope of the ontology

when we start the development of an ontology by defining its domain and scope , we can try to answer several basic questions: 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? Who will use and maintain the ontology? Besides, one of the ways to determine the scope of the ontology is to sketch a list of questions that a knowledge base based on the ontology should be able to answer, competency questions (Gruninger and Fox 1995)

2) Consider reusing existing ontologies

It is almost always worth considering what someone else has done and checking if we can refine and extend existing sources for our particular domain and task . Many ontologies are already available in electronic form and can be imported into an ontology-development environment that you are using.

3) Enumerate important terms in the ontology

It is useful to write down a list of all terms we would like either to make statements about or to explain to a user. What are the terms we would like to talk about? What properties do those terms have? What would we like to say about those terms? For example, important wine-related terms will include **wine**, **grape**, **winery**, **location**, a wine's **color**, **body**, **flavor** and **sugar content**; different types of **food**, such as **fish** and **red meat**; subtypes of wine such as **white wine**, and so on.

4) Define the classes and the class hierarchy

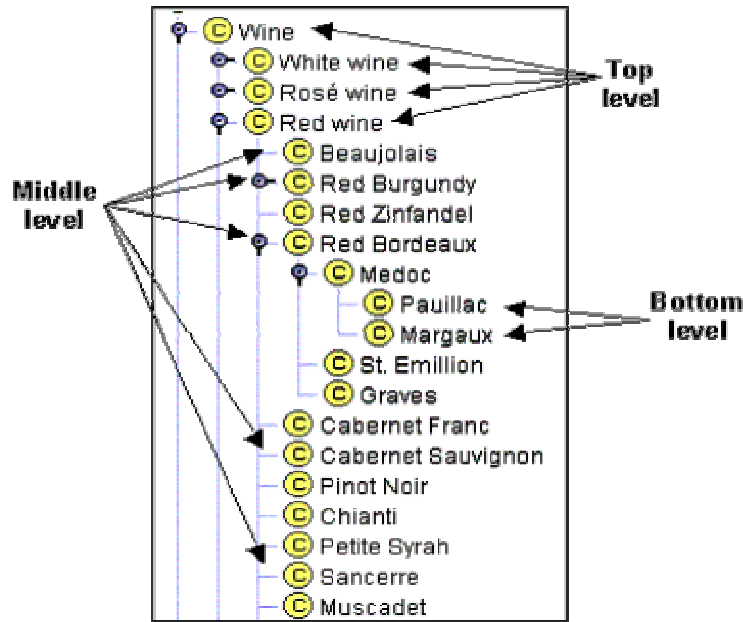


Figure 1. The different levels of the Wine taxonomy.

5) Define the properties of classes—slots

Once we have defined some of the classes, we must describe the internal structure of concepts, it is called slots. In general, there are several types of object properties that can become slots in an ontology including intrinsic, extrinsic properties, parts and relations to the other individuals.

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 ^I	Instance	single	classes={Winery}
S name	String	single	
S sugar	Symbol	single	allowed-values={DRY,SWEET,OFF-DRY}

Figure 2 The slots for the class Wine

6) Define the facets of the slots

Slots can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the slot can take.

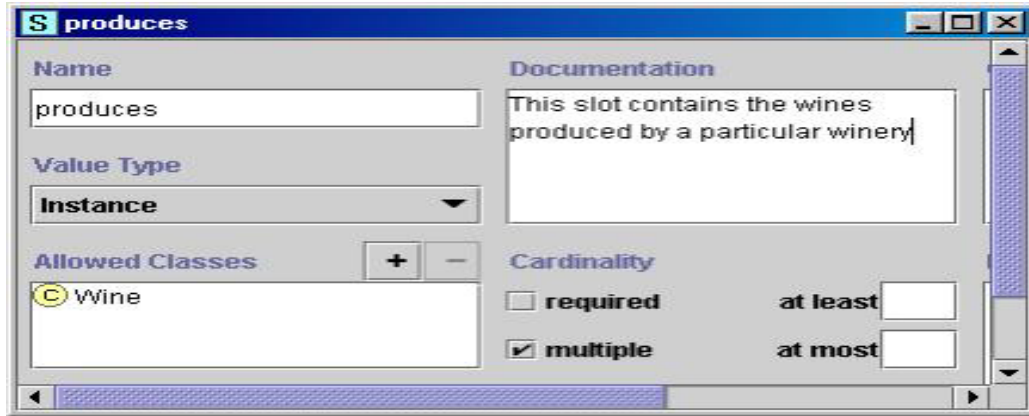


Figure 3 The definition of a slot produces

7) Create instances

The last step is creating individual instances of classes in the hierarchy. Defining an individual instance of a class requires (1) choosing a class, (2) creating an individual instance of that class, and (3) filling in the slot values



Figure 4 The definition of an instance of a subclass of the wine

4 Defining classes and a class hierarchy

This section discussed things to look out for and errors that are easy to make when defining classes and a class hierarchy.

1) Ensuring that the class hierarchy is correct

- **An “is-a” relation**

The class hierarchy represents an “is-a” relation. A subclass of a class represents a concept that is a “kind of” the concept that the superclass represents

- **A single wine is not a subclass of all wines**

- **Transitivity of the hierarchical relations**

A subclass relationship is transitive: If B is a subclass of A and C is a subclass of B, then C is a subclass of A

- **Evolution of a class hierarchy**

As domains evolve, it is difficult to maintain a consistent class hierarchy.

- **Classes and their names**

The name of a class may change if we choose a different terminology, but the term itself represents the objective reality in the world.

- **Avoiding class cycles**

We should avoid **cycles** in the class hierarchy. We say that there is a cycle in a hierarchy when some class A has a subclass B and at the same time B is a superclass of A. Creating such a cycle in a hierarchy amounts to declaring that the classes A and B are equivalent: all instances of A are instances of B and all instances of B are also instances of A. Indeed, since B is a subclass of A, all B's instances must be instances of the class A. Since A is a subclass of B, all A's instances must also be instances of the class B.

2) Analyzing siblings in a class hierarchy

- **Siblings in a class hierarchy**

All the siblings in the hierarchy (except for the ones at the root) must be at the same level of generality.

- **How many is too many and how few is too few?**

There are no hard rules for the number of direct subclasses that a class should have. However, many well structured ontologies have between two and a dozen direct subclasses. Therefore, the following two guidelines:

If a class has only one direct subclass there may be a modeling problem or the ontology is not complete. If there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary

3) When to introduce a new class (or not)

There are several rules of thumb that help decide when to introduce new classes in a hierarchy. Subclasses of a class usually have

- additional properties that the superclass does not have
- restrictions different from those of the superclass
- participated in different relationships than the superclasses

However, sometimes it may be useful to create new classes even if they do not introduce any new properties. Classes in terminological hierarchies do not have to introduce new properties. Another reason to introduce new classes without any new properties is to model concepts among which domain experts commonly make a distinction even though we may have decided not to model the distinction itself.

4) A new class or a property value?

When modeling a domain, we often need to decide whether to model a specific distinction (such as white, red, or rose wine) as a property value or as a set of classes again depends on the scope of the domain and the task at hand. If the concepts with different slot values become restrictions for different slots in other classes. Otherwise, we represent the distinction in a slot value.

If a distinction is important in the domain and we think of the objects with different values for the distinction as different kinds of objects, then we should create a new class for the distinction

5) Limiting the scope

As a final note on defining a class hierarchy, the following set of rules is always helpful in deciding when an ontology definition is complete: The ontology should not contain all the possible information about the domain: you do not need to specialize (or generalize) more than you need for your application (at most one extra level each way). Similarly, the ontology should not contain all the possible properties of and distinctions among classes in the hierarchy.

5 Protégé-2000 and Other Resources

We have used Protégé-2000 as an ontology-developing environment for our examples. Protégé-2000 is a tool which allows the user to construct a domain ontology, to customize knowledge-acquisition forms, to enter domain knowledge and a platform which can be extended with graphical widgets for tables, diagrams, animation components to access other knowledge-based systems embedded applications and a library which other applications can use to access and display knowledge bases as well. Currently, researchers emphasize not only ontology development, but also ontology analysis. For example, Chimaera (McGuinness et al. 2000) provides diagnostic tools for analyzing ontologies.

6 Conclusions

In this tutorial, we have described an ontology-development methodology for declarative frame-based systems. We listed the steps in the ontology-development process and addressed the complex issues of defining class hierarchies and properties of classes and instances. However, after following all the rules and suggestions, one of the most important things to remember is the following: there is no single correct ontology for any domain. Ontology design is a creative process and no two ontologies designed by different people would be the same. The potential applications of the ontology and the designer's understanding and view of the domain will undoubtedly affect ontology design choices.

References

Booch, G., Rumbaugh, J. and Jacobson, I. (1997). *The Unified Modeling Language user guide*: Addison-Wesley.

Chimaera (2000). Chimaera Ontology Environment. www.ksl.stanford.edu/software/chimaera

Duineveld, A.J., Stoter, R., Weiden, M.R., Kenepa, B. and Benjamins, V.R. (2000).

WonderTools? A comparative study of ontological engineering tools. *International Journal of Human-Computer Studies* **52**(6): 1111-1133.

Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**: 199-220.

Gruninger, M. and Fox, M.S. (1995). Methodology for the Design and Evaluation of Ontologies. In: *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, Montreal.

Protege (2000). The Protege Project. <http://protege.stanford.edu>