THE AUTONOMOUS SYSTEMS LABORATORY
UNIVERSIDAD POLITÉCNICA DE MADRID

A Simplified Guide to Create an Ontology

Julita Bermejo

ASLab R-2007-004 v 0.1 Draft
May 22, 2007

One of the cornerstones of ASys is the ontology that supports the engineering process and the very internals of autonomous systems. The purpose of this document is to describe the major elements and steps when developing an ontology. It does not attempt to be a comprehensive description on how to do it, but to guide newcomers on what to consider when attempting to develop an ontology. Let's start from the beginning.

Contents

1 Why to bother developing an ontology?

If one has a look at the literature, there are several reasons that people provide to claim the benefits of developing an ontology.

Basically, they can be summarize as follows:

- To share common understanding of the domain and the related information either among humans (the initial purpose) and among software agents: let's use the same concepts and names among researchers!
- To be able to reuse knowledge: how many times projects, developments and so forth start from scratch?
- To separate domain knowledge from operational one: it is not the same what is used than how to use it.
- To analyze domain knowledge: the thorough analysis usually encompassed in an ontology development, provides an excellent knowledge of the domain.

2 What IS an ontology?

I will not get into giving a review of the all available definitions, although the best suitable one for our purposes is (by Gruber and Borst):

> *"An ontology is a formal, explicit especification of a shared conceptualization.* ***Conceptualization*** *refers to an abstract model of some phenomenon in the world by having identified the* relevant concepts of that phenomenon. ***Explicit*** *means that the type of* concepts used, and the constraints on their use are explicitly defined. ***Formal*** *refers to the fact that the ontology should be* machine-readable. ***Shared*** *reflects that notion that an ontology captures consensual knowledge that is, it is not private of some individual, but* accepted by a group*".*

Read it twice, there is more meat than meets the eye in this definition!

3 What IS IN an ontology?

Basically, any ontology should contain:

- **Classes** that represent concepts (either physical/specific or abstract/conceptual). They could be organized in taxonomies to define superclass- subclass hierarchy.
- **Relations** that represent association between concepts. They are usually binary.
- **Attributes** (also called properties, slots...) to describe the features of the concepts.
- **Formal axioms** to model sentences that are always true.
- **Functions** are special case of relations
- **Instances** that represent elements or individuals in an ontology.

4 Types of ontologies

There too many types out there. To keep our focus, and as starting point, let's say we will develop a **domain ontology**, i.e., a conceptualization of concepts, relations and attributes belonging to a particular field of our interest.

5 A methodology: general ideas

Once again, there are several well established and defined methodologies in the literature. In practical terms, to develop an ontology one should:

- Define concepts, i.e., classes
- Organize them somehow in a taxonomy (remember inheritance issues among superclass-subclass)
- Define relations among the classes
- Define the attributes and which values they can take
- Define instances, i.e., "real" elements in our domain

- If we have time and knowledge enough, axioms and functions

Note: in general, ontology people consider it a lightweight ontology or taxonomy if you not reach the axiom part!!. But they tend to be picky on what an ontology is. Sometimes, a well defined taxonomy could be good enough.

Before we go any further, stick these basic ideas throughout the development process:

- There is **not a correct way** to model a domain: you can have an idea on the domain that will for sure differ from mine. Let's discuss it and reach an agreement. None of them are wrong.
- The ontology development is an **iterative process**: I have not found anyone capable to develop an ontology in one day to full extent. It seems more a "try and mistake" process, in several iterations.
- If you are really lost, start with **nouns** and **verbs** from your knowledge sources (i.e. the documents you are using and your own knowledge on the domain): it is an old rule of thumb among modelers. A noun will be a class, attribute, instance. A verb will be a relation. Obviously, refinement and iterations will be needed to further clarify this.

6 OASys Methodology: Steps to follow

6.1 Determine the domain and the scope or purpose of your ontology

Basically, try to find an answer to questions such as:

- Which domain are you thinking of?
- What will you use the ontology for?
- Is it going to be just one, or will you need different subontologies to make it clearer?
- Who will use the ontology?

It is the hardest part to develop the ontology. It might be further clarified, but at least you need a good initial scope or purpose for your ontology.

Formally, it is done using *competency questions*, which I think might prove useful once the ontology is started.... Not the other way round. And remember, you can

go through them as many times as needed. New questions will arise and former answers might change. Some other people prefer to make *use scenarios* of how/what for the ontology will be used.

Example: OASys (Ontology for Autonomous Systems) is an ontology for the domain of autonomous systems, understanding as such systems capable of fulfilling a goal in an environment by adapting to changes to some extent (not to mention cognitive capabilities). We will try to use it to describe what such systems need to work as well as who and how they will be developed. Probably, we will need several subontologies (structure, behavior, agents,...), as the domain seems complex enough. The ontology is intended for autonomous systems developers and engineer (if we get lucky for Model-Driven Engineering).

6.2 Know your sources: documents, experts and existing ontologies

Either you are an expert on the domain or more common you have a partial knowledge of the domain.

In the first case, you can be proud of yourself and go ahead with the ontology. In the second case, you will need more knowledge that can come from:

- Experts: if available, ask everything you want to know. Use them for your purposes, always keeping your manners. Remember, it is iterative, so you will have to do it several times. Grab their terminology, i.e., the words and terms they are familiar with when talking about the domain. Make notes.
- Documents: if you do not have people, you might have literatures, documents, technical information, etc. on the domain. Get a highlighter and start underlying nouns and verbs, which make sense both to you and the domain!! Make as many notes as you wish.
- Existing ontologies: your domain might have been modeled before with a different viewpoint or purpose. Research on possible ontologies to be fully or partially reused. There is a lot out there. Look for them, analyze the level of granularity (fancy word to describe if the existing ontology covers the same level of detail you want), select them and evaluate them. Usually, you are not the first one who has thought about the same domain.

Example: For OASys, I do not have the experts around...but some miles away. Therefore, I focused on available documents from them (Ignacio's thesis, Ricardo's papers) and others (literature on autonomous systems, agents...). I had also searched and found ontologies that can be reused (ontologies for agent-based systems, taxonomies on autonomous systems by NIST, ontologies for cognitive systems by Franklin, Guarino, etc). And even so, I will have not found everything!. Different highlighters to underline nouns and verbs have been used. Several notes with

ideas on the margins too. Everything dated, to track the time I thought about a term and what meant to me. It changes, remember the iterative process.

6.3 Build the ontology

An ontology development usually encompasses several tasks. Different methodologies order them differently, but in general you should.

1) Enumerate important terms

This is usually called among practitioners as Glossary of Terms, i.e., **make a list and/ or graphs** of all the **nouns and verbs** you have considered. Use any tool you feel comfortable with: hand-writing lists, tables, graphs **in any format**. If you work within a group, **agree on a common formalism and tool** (remember the ontology is for sharing understanding, do not ruin by using different tools!!).

For each term, try to **write down at least: a name, synonym, a natural language description, type, source (to remember why you put it there), comments**...

Next, **try to decide whether a noun in your list might be a concept, attribute or instance**. *Concepts tend to be nouns standing on their own, attributes look more like nouns that can describe type of things, and instances tend to be nouns about specific things. It will not come in one step, but a pre-classification can be useful. In case of doubt, leave it blank*.

Then, **verbs will end up as relations**, you can **make notes to remember between which concepts**.

Remember, **it is not a closed list**. The iterative process will uncover concepts and relations that did not show up initially.

Example: For OASys, after underlying every possible noun and verb that made sense to me to explain the domain, I came up with a list of more than 500 terms which I have listed in an Excel file. I did not have clear for all the nouns, whether they are concept, attribute or instance. But I classified those clear. REFSENO formalisms will probably help to get in more detail.

2) Define concept taxonomies

The idea is to **classify the concepts in a hierarchy** (called among practitioners as taxonomy). Not all concepts will own a hierarchy, but as you were writing them down, some nouns seem to be related as types (subclasses) of other (superclasses).

Traditionally, taxonomies/hierarchies are done following top-down (from general to specific), bottom-up (from specific to general) or combination processes. Choose one, but I find it more sensible to use a combination (up and down).

There are different types of taxonomic relations, i.e., how the subclasses are related to the superclasses:

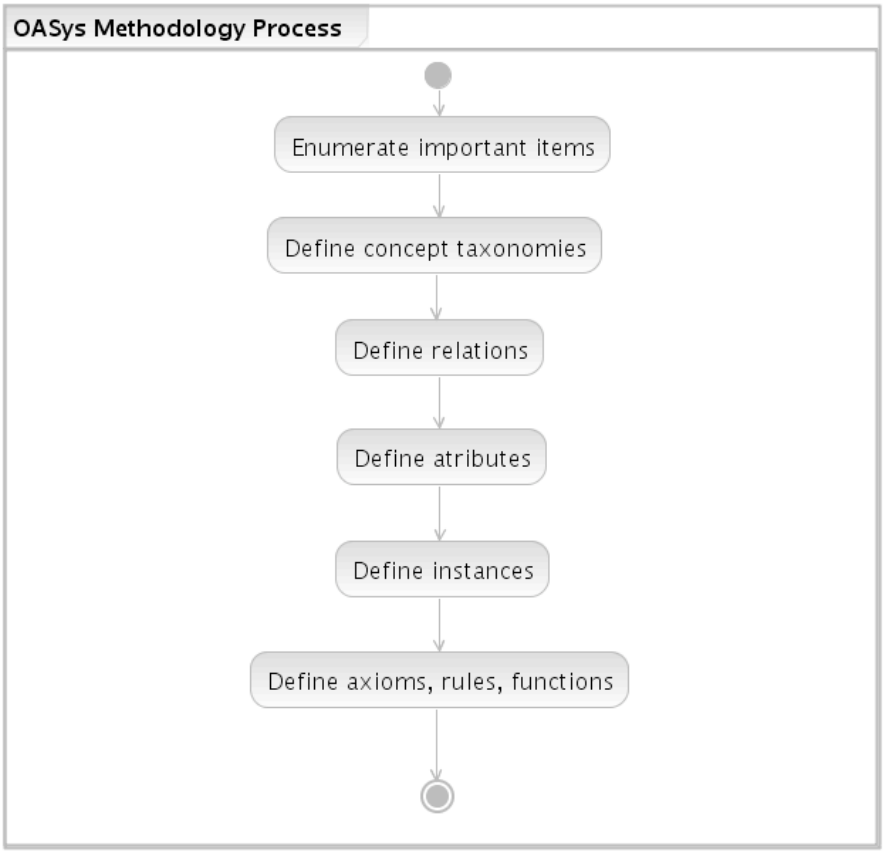

Figure 1: OASys ontology development tasks.

- Subclass: a concept C1 is subclass of concept C2, iff (if and only if) every instance of C1 is also instance of C2.
- Disjoint decomposition of C: set of subclasses of C that do not have common instances and do not cover C.
- Exhaustive decomposition of C: set of subclasses of C that may have common instances and subclasses and do cover C.
- Partition of C: set of subclasses that do not share common instances but cover C.

Probably, defining concepts and taxonomies is the most difficult part of developing an ontology. Several errors may have been made by the developers. They are summarized in 7.

3) Define relations

Between concepts, there will be relations that so far, could have been represented as hand-made diagrams. It is time to describe each diagram and the **relations in detail by giving a name, source concept, target concept, cardinality (how many instances of a concept are related with how many of the others), inverse name** (you can read from A to B, but also from B to A. Sometimes, the distinction is important).

You can further define the relation depending on a **type**, as it happens on UML or use predefined types on your chosen formalism (such as REFSENO). You can always use this as an alternate/complementary representation.

4) Define attributes

At this stage, some of your nouns in the list could have been considered **attributes, i.e., terms used to describe others**.

Ontologists distinguish between class attributes (terms to describe concepts which take their values in the class they are defined, and they are not inherited in the hierarchy) and instance attributes (terms to describe concepts that take their values in the instance, and may be different for each instance).

There is not a unique way to do this, but some guidelines could be:

- This step and the definition of taxonomies are intertwined: some classes might end up being attributes to describe the different classes and/or instances.
- Try to attach the attribute to the most general class/concept that can have that property.
- If it can have a well defined type (integer, string, float) it is an attribute, not a class.
- Try to define type attributes (integer, string, float, ...). You can define your own types, or use the traditional ones. If you follow a formalism, such as REFSENO , you will have to stick as much as possible to it.
- Try to define a range, value, precision, related classes....

Gather all the information **about each attribute: name, concept name related to it, value type, range, value....**

5) Define instances

An instance is an individual of a class, you can describe in detail relevant instances that may appear by giving them a **name, concept to which they are related, attribute names and values.**

6) Define axioms, rules and functions

As said before, if you do not get here, ontologists consider your development a taxonomy. Some require axioms and rules to be described before describing instances. It is up to you.

7 Taxonomy Evaluation: Defining Classes and Hierarchies

At the end, your ontology should be **precise, consistent, complete and concise**. Evaluating somehow is important.

This section attempts to describe common errors as described by others who know a lot on the topic, as well as my own errors. It is compulsory to evaluate the concepts and the taxonomies, attributes and relations you may have made. Otherwise the ontology might be useless.

7.1 Class definition

- Remember a **class is a concept**: ontologists and modelers interchange the two terms. Generally, *they prefer to talk about concepts in the end*, as classes seem to be too related to other domains. However, I use them as synonyms.
- **Classes represent concepts in the domain and not the language/words to denote them**: your nouns represent concepts regardless of being in English, Spanish or even the noun you have chosen. You can have a class named Car, which could be later change as Automobile...or even translated as Coche, Buga, Auto, Carro (name of a car in Mexican Spanish)...Remember, they are the same underlying concept !!
- **Synonyms for the same concept do not represent different classes**: when listing your nouns, look for and detect synonyms referring to the same concept. Chose one and write down the rest for documentation.
- **A class is not only "real" entities in the domain**: a common mistake is to think only of concepts that exist as real objects or entities in your domain**.** Firstly, a class can model abstract/mental concepts if needed (can you touch meaning, intelligence, etc?). Secondly, a class can be also be used to introduce concepts for modeling reasons (upscale modeling, but keep it mind).
- A new class?: one of the hardest decisions is when to introduce a new class or to consider the distinction as an attribute. There is not a rule to solve this problem. Some people argue that you introduce new classes to model new properties...but you might also need them for modeling purposes.
- Keep a balance: it is as bad to have few classes as to have too many. The same rules for subclasses, if you have only one, there might be a problem. If

you have too many subclasses, your ontology might be crying for some new intermediate classes to be introduced.

- Class or attribute?: when modeling, you need to decide whether a distinction is modeled as a class or an attribute value. The answer lies in the scope defined for the ontology. Think on how important the distinction is for you, and especially for the domain. If you think it is important, and different values make you think of different types of entities (objects), create a new class. If not, it will be probably an attribute with different values.

- Class or instance?: Once again, this depends on how the ontology will be used. It has to do with the level of granularity, i.e., the level of detail you need/want to achieve. Generally speaking, instances are the most specific concepts represented in the knowledge base.

7.2 Class hierarchy

- **Avoid class cycles (circularity errors)**: if class A has a subclass B, which in turn, is superclass of A, you have a cycle. It is more likely that they are equivalent. Try to redefine them...if possible.

- **Classifying classes where they do not belong (semantic inconsistency error):** when classifying classes or instances, you can classify a concept as a subclass of other concept to which it does not really belong. Be careful.

- **Careful with your classification (partition error):** if you stick to the four types described before, be aware of instances and classes being common when they should not.

- **Incompleteness of taxonomies:** although you should not model all possible things, you have to define the ones you need for your level of detail and scope.

- **Redundancy:** it happens when you have to classes or instances with the same formal definition. Look for them.

7.3 Other hints

- **Forget the implementation level:** it is difficult not to think of the final implementation in a language (UML, SysML, OWL...) when developing the ontology. Keep it neutral with tables and graphs. You will later implement them on an specific language with a specific tool. Ontologies are conceptualizations not matter how they are implemented...Be aware!

- **Try not to use "reserved" words in your names:** obvious, but unless you are developing an ontology as a metamodel to be further instantiated, using terms such as class, attribute, and so as concept names do not make much sense.

- **Choose a naming convention:** if singular, singular; if plural, plural. Capital letter vs. lowercase.... Choose an existing one and stick to it (note, although you should not consider the implementation, sometimes following a formalism such as UML notation helps).

- **Limit the scope:** among practitioners, it is called the *possible minimal conceptualization.* In other words, your ontology should not contain all the possible information about the domain, but the one you need for your application. Do not attempt to include all possible concepts, their attributes and relations. Even so, document everything you have though of. It might be obvious for you during your development, but a later reuse of your ontology might need some relations or attributes you skipped in your development for a particular domain and application.

References

Akerman, A. and Tyree, J. (2006). Using ontology to support development of software architectures. *IBM Systems Journal*, 45(4):813–825.

Anquetil, N., de Oliveira, K. M., and Dias, M. G. B. (2006). Software maintenance ontology. In Calero, C., Ruiz, F., and Piattini, M., editors, *Ontologies for Software Engineering and Software Technology*, chapter 5, pages 153–173. Springer-Verlag Berlin Heidelberg.

Avancha, S., Patel, C., and Joshi, A. (2004). Ontology-driven adaptive sensor networks. In *Proceedings of MobiQuitous 2004. First Annual International Conference on Mobile and Ubiquitous Systems:Networking and Services*.

Baclawski, K., Kokar, M., Kogut, P., Hart, L., Smith, J., Letkowski, J., and Emery, P. (2002). Extending the unified modeling language for ontology development. *Software Systems Modeling*, 1:142–156.

Cranefield, S. and Purvis, M. (1999). UML as an ontology modelling language. In *Proceedings of the 16th International Conference on AI (IJCAI'99)*, Germany.

Falbo, R. A., Natali, A. C. C., Mian, P. G., Bertollo, G., and Ruy, F. B. (2003a). ODE: Ontology–based software development environment. In *Proceedings of IX Congreso Argentino de Ciencias de la ComputaciÛn*, pages 1124–1135, La Plata, Argentina.

Falbo, R. A., Natali, A. C. C., Mian, P. G., Bertollo, G., and Ruy, F. B. (2003b). ODE: Ontology–based software development environment. In *Proceedings of IX Congreso Argentino de Ciencias de la Computación*, pages 1124–1135, La Plata, Argentina.

Guarino, N. (1998). Formal ontology in information systems. In *Proceedings of FOIS'98*, pages 3–15, Trento, Italy. IOS Press, Amsterdam.

Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31.

Kogut, P., Cranefield, S., Hart, L., Dutra, M., Baclawski, K., Kokar, M., and Smith, J. (2002). UML for ontology development. *The Knowledge Engineering Review*, 1(17):61–64.

Mizoguchi, R. and Ikeda, M. (1996). Towards ontology engineering. Technical Report AI-TR-96-1, The Institute of Scientific and Industrial Research, Osaka University.

Nowostawski, S. C., Purvis, M., and M. (2000). Is in an ontology or an abstract syntax?: Modelling objects, knowledge and agent messages. In *Workshop on Applications of Ontologies and Problem–Solving Methods*, pages 16.1–16.4.

Noy, N. F. and Hafner, C. D. (1997). The state of the art in ontology design. *AI Magazine*, 18(3):53–74.

Noy, N. F. and McGuinness, D. L. (2001). Ontology development 101: A guide to creating your first ontology. Technical Report KSL–01–05, Stanford Knowledge Systems Laboratory.

Pease, A. (2001). Evaluation of intelligent systems: The high performance knowledge bases and ieee standard upper ontology projects. In *Proceedings of the Workshop Measuring the Performance and Intelligence of Systems, PerMIS'2001*, Mexico D.F.

Ruiz, F., Vizcaíno, A., Piattini, M., and García, F. (2004). An ontology for the management of software maintenance projects. *International Journal of Software Engineering*, 14(3):323–349.

Tamma, V. (2001). *An Ontology Model supporting Multiple Ontologies for Knowledge Sharing*. Phd, University of Liverpool.

Uschold, R. J. and M. (1999). A framework for understanding and classifying ontology applications. In *12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*, Banff, Alberta.