Semantic Modeling Guidelines

Version 1.0

metaphacts

metaphactory is an enterprise knowledge graph platform that helps capture knowledge and domain expertise in explicit semantic models using a visual interface. This document reflects common best practices and guidelines for semantic modeling in alignment with metaphactory software. Readers of this guide may use metaphactory to follow along and reproduce the ontology shown in the examples. To learn more, visit metaphacts.com or check the documentation website.

Contents

1 Introduction

This document serves as a guide to allow you to learn the basic concepts involved in modeling and get started with modeling on your own. In this section the concept of modeling will be introduced with some motivating examples. In later sections, you can learn about modeling concepts and begin your own semantic modeling exercises.

1.1 What is semantic modeling and how does it help?

If you are new to **semantic modeling** you may be wondering what all of it is for anyway. Are we learning about computer programs or artistic design? Is modeling about drawing diagrams for a presentation? Or is it for grammar and linguistics? The answer is actually a little bit of all these things. When we model, we are describing things in an explicit way, so that people and machines can both understand and make use of information in the way it was intended. In this section, we will talk a little about the motivations behind modeling and give examples of how it can help you accomplish your tasks.

Whether you work as a data analyst, a marketing specialist or a software engineer, doing your job efficiently will involve plenty of collaboration and communication with colleagues. These may be people in your team or department, but they may also work in other areas, like different business units or may even be outside of your company altogether. Very often, you'll be asked to create a report or review documents without much context, and you might find yourself wondering what something means because you lack the context to understand it on its own. It can feel like all you see are words and numbers without any meaning, even though you know it is there.

Imagine that you need to talk with service providers like the car mechanics who fixed your salmon beetle the other day. You're not a machine (we hope), so you probably understand that in the previous sentence, the beetle mentioned refers to a car. Because we mentioned a car mechanic, it should also be fairly obvious that we're talking about a salmon-colored Volkswagen Beetle. However, a computer might not grasp this distinction since it only understands the information it sees right in front of it and not the surrounding context.

Misunderstandings like this happen on a daily basis, especially in complex business environments. Important semantic context for the data we create and exchange is often missing for those we don't usually collaborate with. And if this context is missing for too long, misunderstandings and omissions can arise that have a serious impact on business success.

Let's take our car example and re-frame it in a business context. Imagine you're a newbie data analyst at a car and motorcycle manufacturer and you need to prepare a report that answers questions such as "Which suppliers do we use for wheels?". To get this information you will probably need to access tens (or even hundreds) of individual systems and applications and manually compile this data. You might not even know which applications to look at and might need to ask colleagues for help. But when you ask, they come around with a follow-up question like "Do you mean wheel suppliers for 4-wheel or 2-wheel vehicles?" And then while you dig through your spare parts database(s), you might ask yourself "How do I know which wheels are used for which vehicle?". Just like that, what seemed to be a simple question turned into a major headache.

This is what we call the semantic gap: When colleagues who work for the same company, maybe even the same department—but also humans and machines—don't speak the same business language. Ideally, we could include the explicit meaning of the words and terms we use to refer to certain things alongside the content we are creating, so that no matter whether you're a human or a machine, you can look at a spare parts database and immediately understand what a specific wheel type is used for, who manufactures it, where it's shipped from and where it appears on the final vehicle.

Using tools like the diagram in Figure 1, which organizes the information from our example in a structured way, we can help address parts of the semantic gap. From reviewing the diagram, we as humans may be able to interpret that there are different kinds of vehicles and that the Beetle we

mentioned is a specific 4-wheel vehicle, using a specific set of wheels from the supplier Pirelli, and has a color called “salmon”. The purple area of the diagram tries to convey that “salmon” is a type of orange, and humans will probably also understand that it implies that “orange” here is a color too, not a fruit. However, the diagram still leaves quite a lot of ambiguity and, in order for every person looking at it to derive the same interpretation, it likely requires additional communication, context or experience. Additionally, the diagram can not be exchanged or grasped by machines. So while more useful than the ambiguous sentence we started out with, this diagram still does not fully bridge the semantic gap. But this is exactly what we can achieve with semantic modeling.

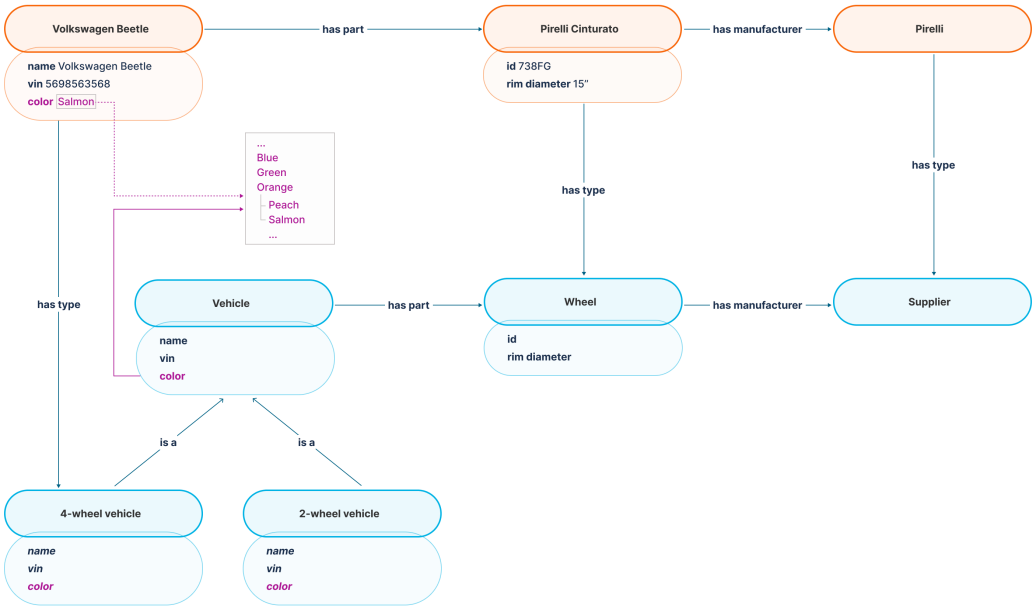


Figure 1: Simple diagram representing information about vehicles

Unlike a simple visual diagram, semantic modeling excels at making implicit information explicit by conveying context and meaning for unambiguous interpretation by both humans and machines. In semantic modeling, this is achieved through formal **semantics** and definitions built into visual editing software, which uses strict syntactic definitions for anything it represents. This means that the visual language used in semantic modeling has a direct correspondence to a computer-understandable language that you can make use of in your own applications. And by following standards for interoperability, whatever has been modeled can be queried, reasoned about, validated, serialized, exchanged and interpreted across a variety of tools and systems.

Using semantic modeling, we can help address and bridge the semantic gap by structuring and capturing the meaning of data in a human-readable format, while at the same time teaching machines what to make out of all this information and providing a foundation for reasoning that AI applications can operate with.

Semantic modeling explicitly captures the relevant context and relationships for concepts related to an organization’s business domain - or any domain for that matter. Modeling allows you to surface the story behind your data so that humans and machines can accurately and efficiently interpret it. This enables more advanced analysis and exploration of your data and ultimately fosters better collaboration and communication among colleagues because you are now all on the same page about what your data means and the language used to describe it.

1.2 What is involved in semantic modeling?

Now that we’ve talked about semantic modeling and how it can help you, let’s have a look at some of the semantic tools that we are working with.

The blue part of the visual diagram in the previous section tries to describe the types of concepts we have in our data – like vehicles or wheels – as well as the attributes and relations that are used to describe them – like the fact that a vehicle has a name, an identification number and a color, as well as parts such as wheels. In semantic modeling, this is what we can capture through an **ontology** using clearly defined and unambiguous semantics and a machine-readable format. The logical structure that can be captured in an ontology comes naturally and is evident to a human, like me and you, but not to a machine. Also, where some users might hold valuable expertise on vehicles in the context of parts, others might carry insight in the context of suppliers, which is why capturing the relations between concepts is so important because, in the end, this provides data consumers with a big picture of all relevant data points. The ontology is central to everything we do as part of our modeling process as it ultimately enriches our data with semantic context that humans and computers can interpret.

For some properties or relations – like colors – it's sometimes useful and important for consistency to use a pre-defined collection of terms ordered by a hierarchy that supplements the ontology with relevant terminology. This is the type of information that you see in the purple part of the diagram above and what can be solved using the mechanism of a controlled **vocabulary**. Using a controlled vocabulary is helpful in this case for two reasons. Firstly, it adds valuable semantics to a term – which is how we know we're talking about colors and not fruit or fish. Secondly, we can ensure that everyone uses the same terms to refer to certain colors. In our example above, we enforce "peach" as an official shade of orange, but not "apricot", though we can still document it as a synonym to support discoverability for all those who prefer using it.

1.3 How does the model relate to data?

We've discussed what is included in a semantic model, but how does this relate to the actual data – or what you see in the orange part of the diagram? In the end, a semantic model is nothing else but a structured way to describe your data points and their meaning. It serves as a basis and guide to link the actual things we model so that they are represented in the same explicitly structured way.

Therefore, because in the ontology we can model that vehicles have a name, a color and a relation to wheels, we now know that our Volkswagen Beetle – which is a type of vehicle – has a name and a color and is related to a specific type of wheels – namely the 165 HR 15 Pirelli Cinturato CA67. This is what we call **instance data**.

2 Getting started with modeling

In this section, we will talk about the elements you are working with when you model and show what you can do with them to represent your information and its semantics. As mentioned in the introduction, ontologies are central to everything you are doing with your modeling, so they will be the focus of this section, even though other artifacts also use these semantic elements as well.

Beginners should interact with the ontology editor while reading the examples and create a new ontology that they can experiment with so that they can see for themselves how things work. Throughout this section we frequently use examples from the ontology in Figure 2. You will start with simple examples and progressively learn more complex expressions so that you can later use them to create an ontology of your own.

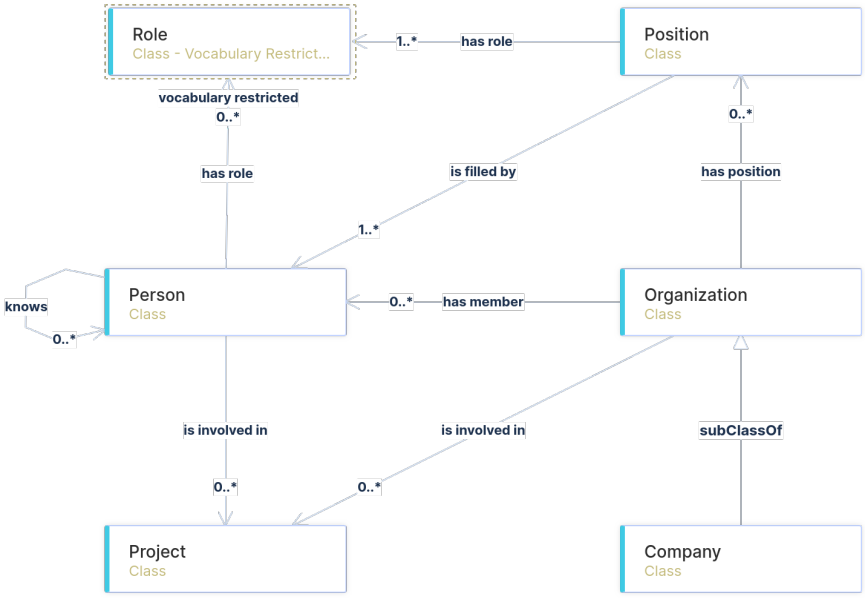


Figure 2: Example ontology

2.1 Basic elements in the semantic modeling language

There are three fundamental elements you use when thinking about and modeling with ontologies: **Class**, **Attribute** and **Relation**. In this section, we'll give a quick, general introduction to each and expand on what they are in subsequent sections.[1]

2.1.1 Class: How do you define the things you model?

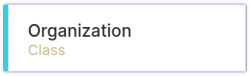
Classes are the fundamental building blocks for ontologies. A **Class**[2] represents an idea or object, which can be either physical or conceptual. To better understand what they are, let's first create a simple class called **Person** to try it out. If you create more classes as you read, make sure to give

[1] Here we are providing a very general first picture for someone new to modeling, the formal semantics and specifications for topics in this section are more comprehensive.

[2] If you are familiar with object oriented programming be careful not to mix up these ideas! While there are many similarities, ontology classes are for representing semantics and not programming so there are some critical differences.

them a good name and not random characters or fillers like “TEST” or “x” so that you remember what they are when you come back later.

(a) A physical thing



(b) A conceptual thing

(c) An abstract concept

Figure 3: Example classes

Now that you have a class, let’s look at different kinds of things you can represent with classes.

In Figure 3a, you start with the most straightforward type of example, where you model a physical object or thing in the world. Here is a class called **Person** just like the one you created, which obviously exists in the real world, as you and I are both people. People usually have attributes like a birth date and a name, and they are also usually related to other people and things in ways you can describe with your ontology - but we’ll get to this a bit later. Typically, you name your class in the singular case so that you can say for any instance of your class that it `is a` **Person**.

Classes can represent all sorts of things, even things that might not occur to you right away as classes. For example, you can also have classes that represent abstract concepts about real things, such as the **Organization** in Figure 3b. In this case, there are probably some physical aspects to what you are talking about, but here the concept is more about the legal or formal entity involved rather than, for instance, the building where it may happen to be located.

In the end, any set of objects with shared semantics related to your ontology can be a class, even if it is very abstract. This can be seen in Figure 3c, where we have a class that represents a **Role** that someone may have in an organization. The class here is meant for things like director or coordinator, where the instances are distinct from actual physical things and you are defining something like a job description. Even though it is totally conceptual, **Role** is a class because you want to model things about a **Role** irrespective of the **Person** who may fill it at any particular time or the **Organization** it may belong to. You can later use this abstract class to augment your ontology by describing the connections between elements of **Organization**, **Person**, and **Role** like you will see in the following sections.

Class Naming Tips

- A class name is usually singular (e.g., **Person** not People).
- Class names are usually capitalized (e.g., **Organization**).
- Classes generally use nouns or noun phrases (e.g., **Organization** or **Software Business**).
- For classes named with noun phrases, you typically use title case (e.g., **Software Business**).

2.1.2 Attribute: How do you describe your classes?

Attributes contain the type of information a class of objects can use to describe themselves. For example, the class representing a **Person** shown in Figure 4 may have a `birth date` or a `name` attribute; since these are attributes shared by real-world instances of this class. Try to add a few attributes to your class from the previous example to see how they work. Attributes in semantic modeling are abstract and do not assume that you know a value for them already, so setting up the expected types and structure should be sufficient when you are modeling.

Figure 4: An example of attributes on a class

When you model, you also want to think about attributes that are shared between many classes to make sure you are not creating many slight variations of the same attribute. For example, some attributes like `name`, which is usually represented with `string`, can easily apply to many classes at the same time without any issue capturing the same semantics. `string` attributes like `name` can also have an additional language identifier in case your data includes labels in different languages.

Other numerical attributes such as a `birth date` for a **Person** and a `foundation date` for an **Organization** are both `dates`, however, they may have slightly different semantics since you probably wouldn't use `birth date` to describe an **Organization**, so it makes sense to have them as distinct attributes.

By default, you can assume that a class can have anywhere from zero values for an attribute to as many as you could possibly want[3]. However, it is also possible to define limits for the number of values that an attribute can have. This can be helpful when you know there is no possible way the information could appear differently and still be correct, such as a **Person** always having exactly one `birth date`. In case there is some inconsistency and you find a **Person** with two `birth dates`, a computer will be able to tell right away that there is an issue.

Whenever you have information like `string`, `integer`, `date`, etc. about classes in your ontology, you should usually use attributes to represent them. If you have many attributes that don't seem

[3]This number can be unspecified but it cannot be infinity, in case you are mathematically inclined and looking for trouble.

to fit well into your class, often this is an indication that you need to break down the problem into more classes that you can connect together, as you see in the next section.

Attribute Naming Tips

- Attributes are typically named after the internal properties of a class (e.g., `length`, `birth date`, `color`, etc.).
- Attribute names are usually lowercase.
- Grammatically, attributes are usually either:
  - Nouns (e.g., `length`) - most common
  - Verbs/verb phrases (e.g., `has length of`) - less common
- When a verb is used for an attribute name, it should be present tense (e.g., `has`, not had).
- Multi-word attributes should use spaces to enable human-readability, although some prefer camel-case.
- No matter which style you choose, you should always be consistent!

2.1.3 Relation: How do you relate your classes to one another?

Relations model connections between two classes, as you can see in Figure 5 where a second class called **Project** is connected to **Organization**.

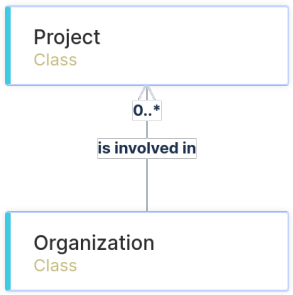


Figure 5: An example relation

Technically, a relation is not just a connection between two classes, but also has a direction, as is indicated by the arrow in the diagram. This direction is really important and has an impact on the meaning of the relation. For example, Figure 5 shows that an **Organization** `is involved in` a **Project**[4], but not the other way around.

Relations are quite flexible, and you can even relate a class to itself. This might be helpful if you have a class **Person** in your ontology and you want to say that people `know` each other, or to say it differently, that a **Person** `knows` another **Person**.

You can also specify limits on relations in the same way that you can for attributes. This allows you to use your ontology to say things like "A **Triangle** `has` at least 3 **Sides**" and "A **Triangle** `has` at most 3 **Sides**"[5] so that you can make sure that there is always an appropriate connection between

[4]The **Organization** can actually be involved in any number of **Projects**, including 0, as you can see in the 0..*.
[5]Having both a minimum and a maximum number that are the same means that it must always have exactly that number. (e.g. $x \leq 3$ and $x \geq 3$ means x = 3)

members of two classes. It is better to do this more sparingly with relations than attributes, and only when you definitely know that something is truly necessary: flexibility with relations will allow your ontology to better adapt to new situations that change how data may be presented when sharing and reusing.

In addition to relations that you can write yourself, there is also a very important relation that comes with its own pre-defined semantics called `subClassOf`.

The `subClassOf` relation

`subClassOf` is a special relation that is used frequently when modeling, as shown in Figure 6. This special relation is used to describe "`is a`"[6] relations, for example when you want to state that "Every **Square** `is a` **Rectangle**" or "If something is a **Cactus** then it `is a` **Plant**" and so on. The use of `subClassOf` relations generates hierarchical structures, essentially forming a **taxonomy** like you might remember from biology class in school.

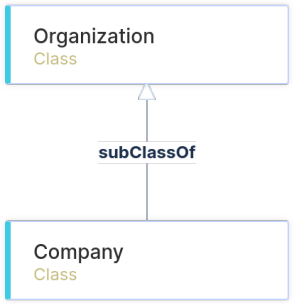


Figure 6: An example `subClassOf` relation

Again, this relation goes one specific way so you should take care that you are drawing connections in the correct direction to represent your semantics. You want to make sure you are saying that "A **Square** `is a` **Rectangle**", which is always correct, and not "A **Rectangle** `is a` **Square**" which is not true in the general case.

Since the use of `subClassOf` has very important consequences, you should only use it when you are sure that it is exactly what you mean. Oftentimes, things are very closely related so it may appear appropriate but the classes are not actually in a `subClassOf` relation. For instance, you can say "An **Organization** `is involved in` a **Project**", but you wouldn't want to say that "An **Organization** `is a` **Project**," even though these two things may always occur together. This is pretty simple to check if you think of a concrete example for the relationship between the classes: *my organization* `is involved in` a **Project**, but *my organization* isn't a **Project** itself because these are separate things.

One of the immediate effects of defining a `subClassOf` relation is the **inheritance** of class relations and attributes in the ontology. A class will automatically inherit anything from a class it is connected to via `subClassOf`, and this other class may itself include attributes and relations from its own `subClassOf` connections as well. If you say that "A **Company** `is an` **Organization**", using

[6] For these guidelines, `is a` is used in order to write more readable example English sentences. In practice `subClassOf` will be used with the ontology, but here you may consider them equivalent.

subClassOf like in Figure 6, then anything you state about an **Organization** in the ontology will automatically be inherited by a **Company**. If you also later add "A **Sports Team** is an **Organization**" with subClassOf, then the classes **Sports Team** and **Company** are both subclasses of **Organization**. This means that both will inherit from **Organization** in the same way without changing any attribute unique to **Company** or **Sports Team**[7]. The inheritance has a direction just like the relation, where the class that the subClassOf relation comes from inherits from the class it goes to.

Please keep in mind that inheritance will only apply to any attributes and relations on classes that are in a subClassOf relation with each other and not other relations. This can be extremely useful when you are modeling with shared attributes. If you have an attribute that holds for a large number of classes connected by subClassOf relations, then you can simply add it to the most general class and all subclasses will inherit this same attribute automatically. If you return to the previous example with **Company**, **Organization** and **Sports Team**, then assuming **Organization** has an attribute called foundation date, you know that a **Company** and a **Sports Team** will also have a foundation date, even though they probably have their own independent attributes as well.

Relation Naming Tips

- Relations use mostly the same naming conventions as attributes.
  - The exception is that grammatically, you usually want verbs and verb phrases for relations rather than nouns (e.g., "A **Person** knows a **Person**", "A **Person** lives in a **House**", etc.).
- A relation should derive its name from the intended connection between the two classes and indicate the semantics.
  - This allows us to distinguish between different connections between the same classes (e.g. "A **Person** eats **Food**" vs "A **Person** loves **Food**").

2.2 Connecting elements together to form an ontology

So far we have talked about ontologies from a very general and high-level perspective. Now that you have learned about all the various elements within ontologies, you can better understand what an ontology really is.

To put it more concretely now, you can define an **ontology** in the following way: An ontology is is a schematic model, or formal representation, that describes in a precise way the classes of objects you represent, the attributes they have and the relations between them.

The classes themselves are abstract even if they refer to real things or objects: If you have a class **Person**, you will probably have more than one person in your data. And those people may themselves have multiple relations to different classes of objects like **Organization** or **Role**. By defining the structure of how these objects relate in general using an ontology, you can take full advantage of the semantics in the way that you intended it.

Unlike the previous section where you simply looked at small elements, in ontologies you usually have many different classes and relations, as well as references to other ontologies. You usually say that a single ontology describes a specific part, or **domain**[8], of your data, since you are probably not interested in a more philosophical task of modeling the entire universe with everything it contains. This means that for simple things you may end up with small ontologies with relatively concise meanings. On the other hand, if you want to represent the data of a large organization or complex process, often this means you will need to subdivide this model into parts with different subdomains. In either case, you can use ontologies for the same purposes and use the same techniques for development that we discuss in the next section.

[7] Those familiar with object oriented programming should recognize this behavior as similar to polymorphism, though again you should be careful not to think of modeling as programming.
[8] More information about defining a domain can be found in Section 3.1.2.

Ontology Naming Tips

- An ontology should be named based on the main idea it tries to represent (e.g., Organization Ontology).
- Ontologies should be capitalized as if they were titles (e.g., Food Ontology).
- Often it helps to name an ontology with the word “Ontology” (e.g., Sports Ontology).
- It’s helpful to keep a title short and to the point, details can go inside.

3 Methodology and design principles for ontology modeling

In this section, we will consider how to actually start modeling with ontologies, providing a hands-on example and supporting it with best practices and methodologies to help us along the way.

3.1 Getting started: Modeling for the first time

Let's go ahead and get started with modeling an actual ontology based on the elements you have learned so far. All steps in the next subsections will build the same ontology, so if you follow along, you should be able to recreate the example ontology at the end of this section. Whenever you start a new ontology, it is good to follow these steps to make sure you have a good foundation regardless of whether you are new to modeling or an expert.

This section is meant for beginners in semantic modeling and tries to walk users through the process of defining and creating an ontology for the first time, which is why in your example you create a new ontology and all of its classes, attributes and relations from scratch. However, as you gain more experience with semantic modeling, we recommend also incorporating advanced techniques described in Section 3.2 such as defining modularization and researching existing ontologies to think about opportunities for reuse already in the scoping and design phase.

3.1.1 Planning the scope of your ontology

The first thing you should ask yourself when you begin planning your modeling activity is probably the easiest and the hardest question at the same time: "What am I modeling?" The answer might initially seem straightforward, however when you dig deeper, it often turns out that what you thought you were modeling is actually a part of some larger context that can be helpful to include in your ontology. At the beginning, it helps to first plan a bit about the subject of your ontology, the people who will use it, and the insights they will want to extract from the ontology.

In order to refine the subject of your ontology when you get started, you should write down a few competency questions[9] that you would like to be able to answer with your ontology. If you were modeling an ontology about companies that work with yours on various projects, you may write a question like "Which companies are involved in the same projects?", or for example a question like "Which people have been both a manager and a salesperson, independent of the company?". Depending on your data and on your needs, these questions can be very different, but they should all address the same point, namely: "What is my ontology about, and what will I use it for with respect to my data?"

While writing these questions down, make sure to pay attention to the objects involved in the sentences and the ways they might be related. See if there are any concepts you need to represent as classes in your ontology so that you can answer some of your competency questions. Try to generalize as much as possible so that the classes can conceptually cover as many variations in the data as you need. Maybe you already know how these classes are related, and, if so, you probably already have some idea of the general structure of parts of the ontology based on your experience with your data. Take this information and write down a few classes that you think should be in your ontology, and some relations if you know those as well. Looking at the questions from before, you might start with classes like **Company**, **Person** and **Project**.

Next, take a look at your classes and see if there is one that stands out, one that you think might be a central **focus** to your ontology modeling and what it is all about. Often, this class already corresponds to something in your competency questions or data and you know it before you start modeling. Sometimes, it is something you understand in your team and how you talk about the data even if it isn't explicit in the data itself. And in other cases, you might need to make a small inference to capture what you want to model. For example, you might take the three classes from before and decide that **Company** is the main focus of your modeling. But looking ahead, you

[9] Section 3.1.3 will provide more detail about how you can elaborate on and also use these questions to validate and refine your modeling going forward after the initial planning stage.

realize that you also want to add information about other organizations that are related to the companies–for example a government agency or a non-profit organization, so you revise this and start with a class called **Organization** that can represent both types of things. This is an example of why it is important to generalize when you see the opportunity during this initial planning, since it may not be obvious before you start how to best cover all of your use cases.

Identifying this focus class like **Organization** is the first step in defining the domain of your ontology. Once you have it you can return to the other classes that you have identified from your competency questions and think about how they would connect to your data and how they relate to each other — like **Person** and **Project**. Similar to what you thought about with the focus class, see if any of these should be adjusted to be more abstract so that they all align with each other. In the beginning, these classes should all be of a similar generality: it won't be as useful to go straight down into all the little details right away.

Now that you have some classes ready to go, you can write down a sentence or two in natural language to explain what the classes are and how they represent your use case described by the competency questions. For example, you might write something like : "This ontology is intended to represent organizations, as well as the people who are members and the projects that they have worked on. It should be suitable for use with *[your example data source]* from *[your example company]*."

Writing down these sentences has two main purposes: Firstly, you can use them for your initial documentation so that others understand what you are working on, which is important to do anyway. Secondly, they help you think back to the big-picture use case for your ontology so that you can make sure your modeling targets how you want to use the ontology in general without getting lost in all the details. You might have so much fun modeling with your team that you keep modeling and modeling and your ontology is really awesome, but unless you define the general goal at the start for later reference, you might miss your original intention.

When you want to make sure that you are ready with your plan to move on to the next step, verify that you have some good initial ideas for the three main topics that you wrote down. As you build experience, you can spend more time planning once you get a feel for how modeling works and what you can expect. In any case, the really important part is to make sure you prepare some initial competency questions, define the core of your ontology with a starting domain, and document the high-level intention of your modeling before jumping off into all the details later on. It won't help you too much when you are learning to spend hours and hours planning every little aspect of the modeling process since, as you have already seen with the focus class example, once you start working things out, you often need to make adjustments anyways, and this is totally fine - ontology development is an iterative process.

Ontology Planning Check List

- ☐ Prepare a written plan that contains at least three main components:
  - ☐ Competency questions that describe specific questions you would like to answer using your ontology
  - ☐ A starting domain for your ontology that includes some classes, and possibly also attributes and relations if these are already known
  - ☐ A few sentences describing the high-level purpose of the ontology

3.1.2 Implementing the initial ontology design

Now that you have this initial bit of modeling planned out, you can create a new ontology[10] for your domain using the name of the focus class as the initial title. Usually, it helps if you also name it with the word "Ontology", like "Organization Ontology" for your example, just for clarity if you share it later. If you need to edit the title later, this is no issue – the label can be changed at any time. You

[10] If you have been following along with the guide so far, before you begin modeling make sure you open up a fresh ontology to replace anything you made for experimentation in Section 2.

are also free to come back later and edit classes and relations if you change your mind, so don't worry too much about things being perfect and set in stone at the beginning. The important part is getting everything organized and set up so that you are prepared to develop and expand on your ideas.

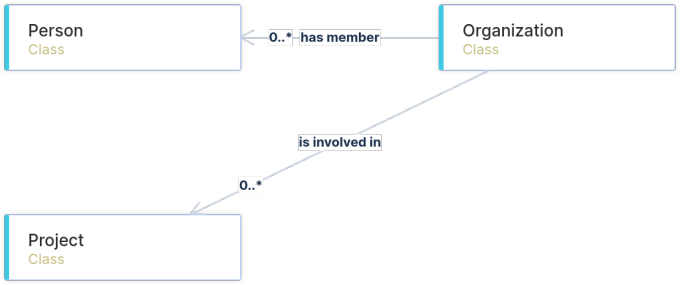


Figure 7: An initial ontology domain

Alright! Now you have an ontology open and ready to go, what next? First, let's add an initial focus class, the one with the same name as the ontology – "**Organization**", to start defining the domain. Next, you can use this class to expand the domain to cover the other essential concepts in your domain, making sure that new content always relates to your central idea – in this example "**Person**" and "**Project**". You know that **Organization**s have members that are people (each of them is a **Person**) so you can put this relation in between the classes like you can see in the diagram. Also, you know **Organization**s are involved in **Project**s so you can add this as well. In the end you should have something like Figure 7. Usually, you want to start with the classes directly connected to the focus class in the first stage, but you can also connect classes in a longer chain as long as they eventually link back to the others. The focus class will be like the hub of your ontology and all of your other classes should somehow connect back to it when you are done modeling.

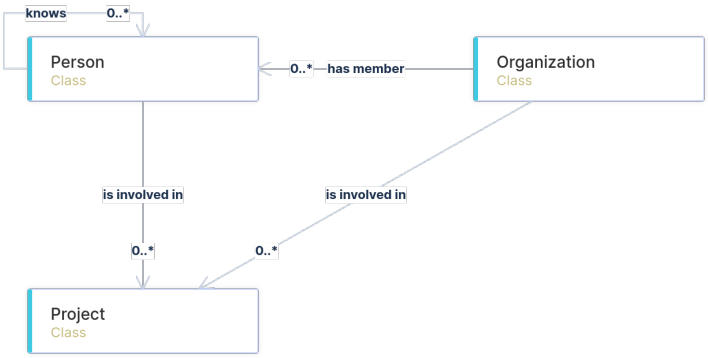


Figure 8: A refined ontology domain

Once you have connected your classes together, you can also check if you need some additional relations to make sure that the initial ontology expresses exactly the semantics you want it to.

Maybe you want to make sure you can represent which people are involved in which projects regardless of their organization, so you add this direct connection between **Person** and **Project** to the diagram. And it might be nice to be able to show which Person `knows` which, so you can add this relation as well and your ontology should look similar to Figure 8. You can add as many relations as you need to, but for now the main point is to just hook things up in a small ontology that roughly matches your intention.

After the classes are linked together with relations, check quickly if there are any attributes that you are already sure belong with classes and add them in before moving on. Easy attributes to start with are things like names and descriptions. Once you are happy with the attributes, it is time to move on to the next step.

Ontology Implementation Check List

- ☐ Create a new ontology that is appropriately named for the domain in your plan
- ☐ Add all classes from the planned domain to the ontology
- ☐ Ensure that all classes are connected to the focus class with relations, either directly or through other classes
  - These relations may already exist in your plan or they may need to be created based on the semantics of the connection between the classes
- ☐ Refine the ontology by adding additional relations and attributes as needed

3.1.3 Checking your work with competency questions

Now that you have started your ontology, it is a good idea to look back and check your **competency questions** to make sure your ontology still matches your initial intent. Any time you work on modeling this is very useful, not just the first time, so make sure to keep these questions somewhere you can get back to them. When you first started, you may have written questions that were very high level or no longer correspond exactly to the language your new ontology uses, so it's fine to briefly revise them at this point, though in general you shouldn't completely rewrite any of them unless your use case has fundamentally changed.

To give a little more detail about the purpose of these competency questions: they are natural language questions that you want to use the ontology to answer, according to the intended purpose you outlined in the description. At the beginning, they served as a tool for planning out what you wanted to do; now that you are actually doing it, they can help you validate that your modeling meets the requirements you set for it as the ontology grows more detailed.

To get started, you can take your initial questions and verify that they still make sense with your current ontology. It is also good at this point to add any questions that you would like to answer in the future but will require some additional modeling in case the initial starting point wasn't detailed enough to include them. Some examples of competency questions for the ontology you have seen so far are included in Figure 9.

1. What kind of job roles exist in which organization?
2. How many people have held multiple roles in an organization?
3. Which people have been a manager and a salesperson, independent of the organization?
4. Who is the most experienced person, having held the most roles?
5. Which cross-organizational projects have the largest number of active team members?
6. Which companies are involved in the same projects?

Figure 9: Example competency questions

Each ontology and use case will have unique competency questions, so these are examples of the form and type of language that would be recommended for your simple ontology and you should write your own questions to match new ontologies you create in the future.

Returning to the example ontology again now, you can see how you might use these competency

questions to validate your ontology. For example, you may want to set up a class to express these roles that you want to model with your ontology in your competency questions.

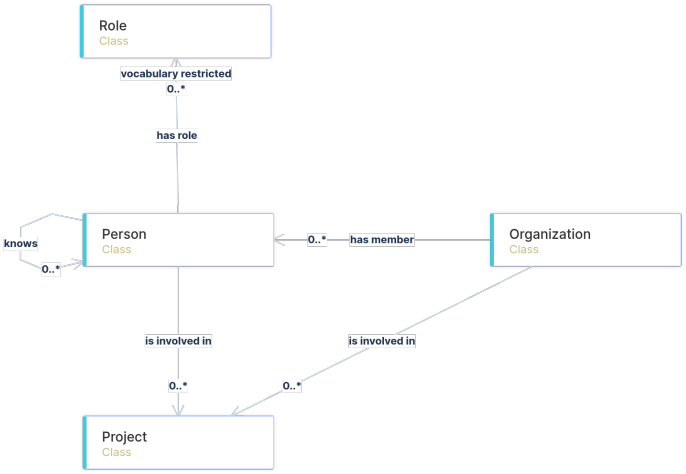


Figure 10: Adding the **Role** class

First, add a class for the **Role** that you might remember from previous sections, which should look like Figure 10. Looking now at your ontology and thinking about your data, you may realize that you already have a fixed set of **Role**s that you want to represent that aren't likely to change, and decide to use a vocabulary to define them[11]. A good example of when a vocabulary should be used is when you have a class that has many different types, which are all known in advance in a controlled way that possibly has some structure, but otherwise do not require further modeling themselves. This is just like the **Role**s you want to model, and when you connect it to the class it should look like Figure 11.

Now that you have this class for **Role**, you might notice a small issue with your representation. Your Organization Ontology has roles and people, but so far you can't directly connect the **Person** to the specific **Organization** where they held the **Role**. The relation between **Role** and **Person** tells you who has what role, and the relation between **Person** and **Organization** tells you about their membership, but these classes don't relate to each other in order to describe a **Person** with their **Organization** and **Role** at the same time, so you still have trouble with questions 1 and 2. Adding a relation between **Organization** and **Role** might get you halfway there so that you can answer question 1, but question 2 would still elude you since there is no way to represent instances of the **Person**, **Organization** and **Role** together.

You can solve this by connecting all three things with another class called **Position**, as you can see in Figure 12, which can represent someone performing a role at a certain time in a certain organization, outside of just the **Person** and just the **Role** by themselves. This can be useful if you want to reference, for instance, the difference between the time you had a **Role** at your **Organization**, and when your coworker also had the same **Role** at the same **Organization**: without adding this class you would have the **Person** and the **Role** but not the association of who did what at which company. Once you link all these things together you should be able to answer questions 1 and 2 now.

This is a good example of the kind of exercise you should do when checking your competency questions, where you always make sure the relevant parts of your data can be connected in your

[11] More information on vocabularies can be found in Section 4.1.

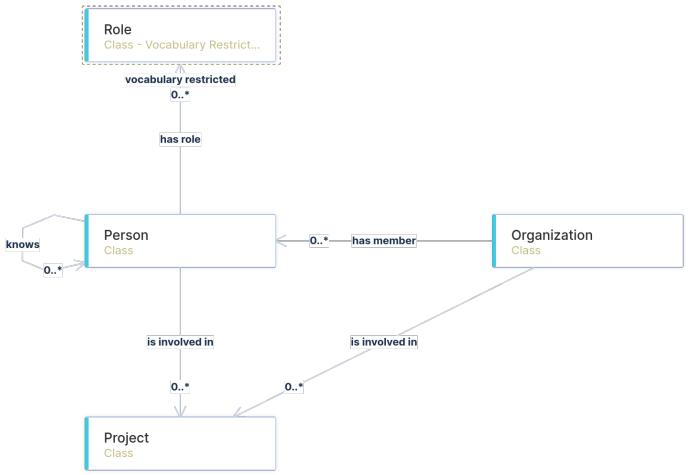


Figure 11: Adding a vocabulary to the **Role** class

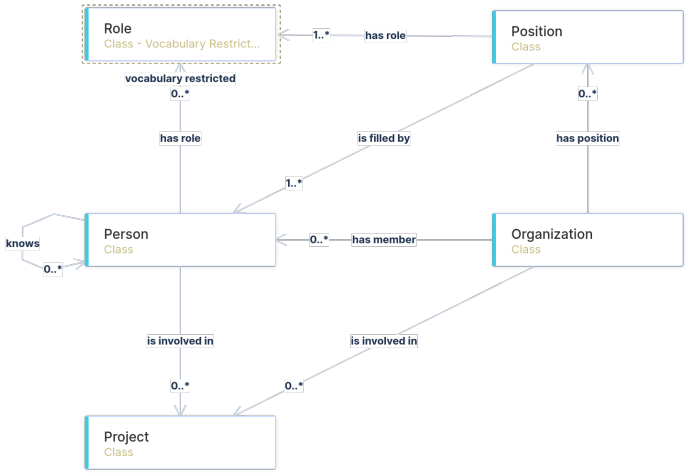


Figure 12: Adding a class for **Position**

ontology in order to answer meaningful questions.

After you start creating a few initial classes in your ontology, it is helpful to start reviewing these questions as you saw in the example. In the beginning, it is fine to have straightforward and obvious questions. You can tweak both questions and ontology in parallel so that your ontology design targets the questions you would like to answer and your questions are aligned with your current ontology. And when you are happy with your ontology, a review of these questions may help finalize the design. Overall, using competency questions should help you create cleaner and more concise ontologies, and help you along the way to plan your development.

Competency Questions Check List

- ☐ Before reviewing the ontology, refine and edit your existing competency questions based on the current ontology in case you made changes
  - Questions should be clear and concise
  - It is totally fine and often helpful to write seemingly obvious questions if they fit your use case
- ☐ Review the ontology to ensure that it can answer each competency question
  - Whenever there is no clear answer to a question, make changes to the ontology to correct this

3.1.4 Documenting your work so far

Before you dig further into modeling, you should take a moment to provide some basic metadata describing the domain you are trying to model. An example of metadata for your example ontology can be seen in Figure 13. This is highly recommended to increase visibility and findability later on, especially when working in a team, and can reduce duplication of modeling efforts by promoting ontology reuse[12].

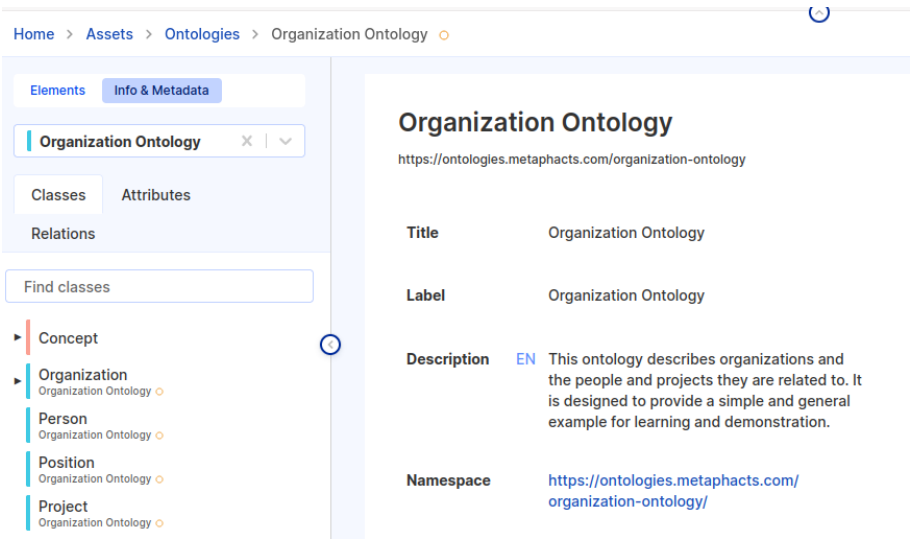


Figure 13: Example metadata

Some important information to add would be things like the description you wrote to describe your intent for the modeling plan, as well as the date that it was created and your name so that others know who to talk to if you share it afterward. The version number is typically handled automatically when a new ontology is started but can be seen here in case it should be edited. It is also possible to define things like contributors or a specific IRI namespaces for the elements of the ontology, but these things are advanced settings and managed automatically, so they are not required in the beginning.

Documentation Check List

- ☐ Add a description to your ontology metadata
  - This can be the same as the description in your plan
- ☐ Verify or add the current date to the ontology metadata
- ☐ Add your name as well as any other people you worked with to the ontology metadata as contributors

[12] More information on reuse can be found in Section 3.2.2.

3.1.5 Refining your ontology

Once you have your ontology set up after following the steps in this section, you should be prepared to iteratively expand, revise and refine your ontology to meet the needs of your use case[13]. Generally, it is recommended to replicate the steps in Sections 3.1.1—3.1.3 whenever you are modeling, except of course that you should use the ontology and plan you already have from your previous modeling sessions when you return. When you model an existing ontology, first take a look at your plan and ontology and integrate any new information with the plan that may have come to light since the last time you modeled. This could include things like new classes, updated competency questions, or a refined description. Next, make changes to the ontology if needed by updates in the plan, then review competency questions to validate what you have done. Repeating these steps helps you maintain consistency in your design while you model so that your ontology meets your requirements and aligns with your intended semantics.

3.2 Next steps: Advanced considerations

In the previous section, you read about how to get started with developing and modeling a new ontology. In this section, you can learn some general techniques that advanced modelers use to model effectively in their teams and organizations.

3.2.1 Modular development

When you have a lot of different classes that you want to model and your ontology starts to grow very large, you sometimes end up in a situation where your ontology is so big that you have trouble remembering or finding things that used to be obvious. Other times, you are modeling and the topics become so conceptually expansive that they no longer fit in your original domain and effectively need their own independent ontologies to describe them separately. Maybe you notice that some parts of your ontology are very similar to others and you wonder if there is a way to model this without so much repetition. In these situations, it is often helpful to take a modular approach to development from the beginning.

With modular development, you start with the highest-level classes of your domain and make a very general overview of the connections between the major classes. Then each of these major classes can be described in its own ontology about that subdomain, continuing to break down the subdomains further in this way until everything is represented. By modularizing the ontology modeling like this, you can compartmentalize the organization of all your ontology data as well as break down the modeling work involved. In teams where some members are experts in one subdomain while some specialize in another, this is an opportunity to split up some of the work between different groups for efficiency and accuracy.

To model in a modular way, you can simply connect two or more ontologies using **imports** to make statements about classes in a subdomain in a separate space from your current ontology that remains linked together semantically. Imports are a mechanism you can use to connect ontologies together and reference the same classes, attributes and relations from one ontology in others. This ability to connect ontologies together can help you divide the modeling task into modules so that you and your team can focus on refining a few particular classes at a time rather than doing everything together all at the same time.

For example, let's extend your "Organization Ontology" by making a class **Company** with `subClassOf` connected to **Organization**, as you can see in the example in Figure 14. This will allow you to describe the **Company** class you came up with in your initial planning in a separate ontology without affecting any of the other classes you modeled about **Organization**s.

Now you should create another ontology called "Company Ontology" to represent your new subdomain and import the "Organization Ontology" so that they are connected. You can then add the imported class to your diagram and use the same class in both ontologies, then follow the

[13] In future versions of this document you will further develop the methodology into a more detailed procedure, for this version the suggestions will be high level and sufficient for getting started.

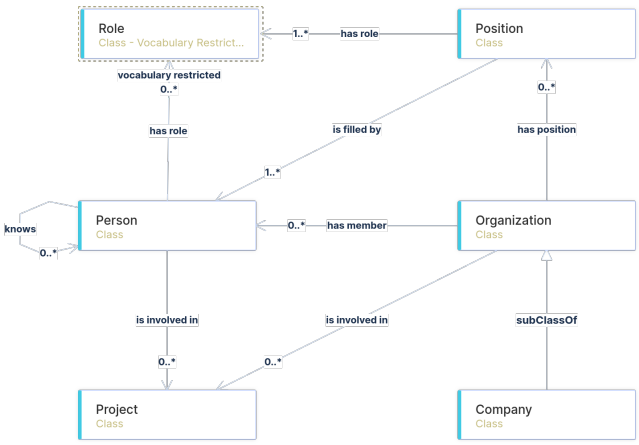


Figure 14: The Organization Ontology

same steps you would for starting a new ontology as discussed in Section 3.1. An example of an initial module for **Company** can be seen in Figure 15.

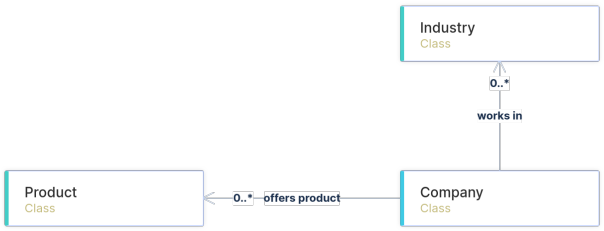


Figure 15: Example Company module

When working on modularized ontologies, remember that you should always edit a class in the ontology where it was created so that it remains consistent anywhere you use it. You can say new things about an imported class in a different ontology, but this will not change the original class and you cannot edit any attributes that it was imported with. Also make sure to still regularly synchronize your progress across teams when modularizing, to make sure everyone stays aligned with the common goal and can share any helpful developments with other knowledge stewards and ontology developers.

The modular approach is intended to be a top-down design process, however, there are times when you need a bottom-up strategy. If you are starting fresh, it is always better to try top-down first, and sit down as a team to design the abstract ontology together. However, when necessary it is still possible to rework this in a bottom-up fashion, where various people can describe their domain ontologies separately, and a designated data steward is then responsible for identifying commonalities between them and extracting a higher-level ontology that glues together the pieces.

3.2.2 Ontology reuse

With ontologies you have a powerful ability to use and reuse statements that you connect to from outside sources with imports. This capability to link to external ontologies has many advantages as you saw in the previous section about modularization. However, you should be careful how you go about reuse from sources outside of your own personal ontologies, since the semantics and use cases of ontologies can vary dramatically, to the point where something that may seem relevant to your situation may represent something that is meant for some other purpose entirely. On the other side of this issue, you also must take care to reuse whenever it makes sense, because if you are always creating new classes and relations for everything you model, then you do not benefit from the shared semantics in your connected data. And without reuse, you may even accidentally recreate something that already exists without noticing, so reuse can also help you spend your modeling time efficiently.

There are two main situations where ontology reuse comes into play. The first is when you internally import and reuse concepts within your own teams and organizations, like the example from the previous section on modularization. In the second case of ontology reuse, you can sometimes use external ontologies that were developed outside of your organization. There can be substantial benefits regarding broader interoperability outside of your organization, but you should be more cautious than before about how you are using these imports because sometimes those come with design trade-offs that add complexity which is not always obvious right from the beginning. In the following sections, we will briefly discuss both of these situations and some techniques you can use to improve your modeling.

Internal reuse

The primary concern you should address when reusing ontologies within your organization is to make sure that your group remains in sync across any of your various sub-domains modeling independently, and that you are each aware of the overall progress of the whole ontology modeling effort. This is typically supported through governance workflows implemented in the modeling software (including publishing workflows with peer review, versioning, notifications etc.), as well as regular sync meetings between groups in your organization. Without regular check-ins, it is likely that teams working separately in related subdomains will end up modeling the same things, so keeping up to date can prevent duplication of work and support integration of common concepts across your ontology.

External reuse

Similar to sharing code between programmers, there is no official rule in ontologies saying what is a good ontology and what is a bad ontology, so you should first and foremost try to make sure that you understand the primary purpose of any ontology you use from outside sources. For example, there are ontologies like schema.org that are primarily designed for the purpose for web annotations and publishing. And there are also other ontologies like the ISO 15926-14 ontology that is designed to support the integration and sharing of data related to the lifecycle of a process plan. Both ontologies have a unique use case that may support your design, or may not add much benefit at all depending on what your ontology is for.

You should always make sure that the ontologies you reuse have documentation so that you can read about the author's intended usage before you start reusing them, and this is even more critical when you use ontologies with no official publication attached. In addition, you should ensure that your externally reused ontologies come from sources that are published and reviewed in some way, whenever possible. Ideally, they are also actively maintained by a larger community.

Some public semantic artifacts are supported as a built-in functionality, like meta-languages for modeling (e.g., SKOS for vocabularies) or metadata standards (e.g., Dublin Core for metadata annotations) and do not need to be imported/managed as ontologies in order to use them.

4 Enhancing your modeling capabilities

This section describes some semantic artifacts that you can use to enhance your ontologies and improve the modeling process in various ways. Vocabularies are the primary focus of this section but there are many more resources to help us model that can be found in the metaphactory documentation and may appear in future versions of this document as mentioned in the conclusion.

4.1 Vocabulary

A **vocabulary** is a special type of semantic artifact that can connect with and enhance ontologies with hierarchical information. A good example of when a vocabulary should be used is when you have a class that has many different types of subclasses, which are all known in advance in a controlled way that possibly have some structure, but otherwise do not require further modeling themselves. Vocabularies can represent a complex hierarchy of concepts inside a single class for more concise modeling.

This means that when you have a lot of concepts that share the same attributes and don't need unique attributes of their own, rather than using a lot of subclasses, it is more efficient to use a controlled vocabulary. Another advantage to using vocabularies is that you can include new terms that you need without having to update the overall ontology. In Figure 16 you can see an example vocabulary that describes possible roles for an organization. Here you have a vocabulary that defines a hierarchy of controlled terms for Roles that are all fixed in advance and that could apply to the **Role** class in your ontology.

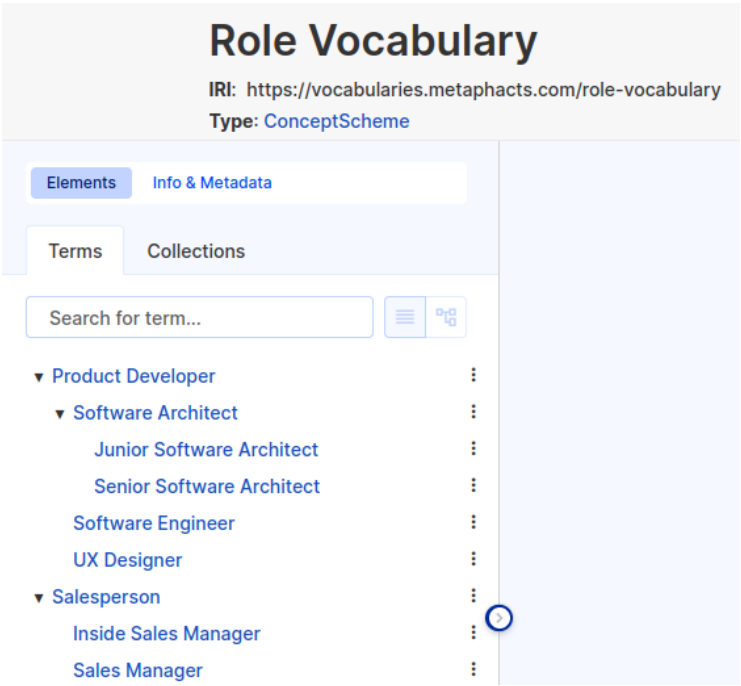


Figure 16: An example vocabulary for the **Role** class

With a vocabulary you can make terms as specific or general as you want, and some concepts may be more specific than others if that is what you need, like you can see in the example where Junior Software Architect is more specific than Sales Manager even though they may both be used in the same way. Terms in a vocabulary can also have multiple languages and synonyms like string attributes in case you have a single term with many names. Once a vocabulary you develop is ready, you can connect it to a class in your ontology to represent it like you can see in Figure 11 in Section 3.1.3. By itself, a vocabulary is not an ontology, but it can integrate seamlessly with ontologies.

Vocabulary Naming Tips

- A vocabulary should follow the naming suggestions for ontologies.
  - Except you should name them Vocabulary (not Ontology).
- Typically, the terms in a vocabulary are capitalized nouns or noun phrases. (e.g. Software Architect)
  - Avoid making longer and longer labels when making the hierarchy of terms deeper. The label you choose should be concise but still convey the meaning.

5 Conclusion and outlook

This is the end of the first version of the modeling guidelines. Much more content will be included in future versions to provide more guidance on advanced and detailed topics. Additional content is planned in the methodology section to develop it further and make the formal methodology explicit. Part of this is to expand on the recommendations for getting started to include similar walkthroughs about refining an existing ontology as well as other common modeling scenarios. The section on enhancing the modeling capabilities will also be expanded to cover other useful topics that can be used to improve modeling and your ontologies such as datasets and other artifacts. The formal semantics of the metaphactory visual editor that align with this guide may be covered in a future version as well.

Although this document is meant to grow over time, it is our intent that this version should suffice for a beginner to start modeling ontologies in an effective way.

Quick Reference

Naming Tips

Class

- A class name is usually singular (e.g., **Person** not People).
- Class names are usually capitalized (e.g., **Organization**).
- Classes generally use nouns or noun phrases (e.g., **Organization** or **Software Business**).
- For classes named with noun phrases, you typically use title case (e.g., **Software Business**).

Attribute

- Attributes are typically named after the internal properties of a class (e.g., `length`, `birth date`, `color`, etc.).
- Attribute names are usually lowercase.
- Grammatically, attributes are usually either:
  - Nouns (e.g., `length`) - most common
  - Verbs/verb phrases (e.g., `has length of`) - less common
- When a verb is used for an attribute name, it should be present tense (e.g., `has`, not had).
- Multi-word attributes should use spaces to enable human-readability, although some prefer camel-case.
- No matter which style you choose, you should always be consistent!

Relation

- Relations use mostly the same naming conventions as attributes.
  - The exception is that grammatically, you usually want verbs and verb phrases for relations rather than nouns (e.g., "A **Person** `knows` a **Person**", "A **Person** `lives in` a **House**", etc.).
- A relation should derive its name from the intended connection between the two classes and indicate the semantics.
  - This allows us to distinguish between different connections between the same classes (e.g. "A **Person** `eats` **Food**" vs "A **Person** `loves` **Food**").

Ontology

- An ontology should be named based on the main idea it tries to represent (e.g., Organization Ontology).
- Ontologies should be capitalized as if they were titles (e.g., Food Ontology).
- Often it helps to name an ontology with the word "Ontology" (e.g., Sports Ontology).
- It's helpful to keep a title short and to the point, details can go inside.

Vocabulary

- A vocabulary should follow the naming suggestions for ontologies.
  - Except you should name them Vocabulary (not Ontology).
- Typically, the terms in a vocabulary are capitalized nouns or noun phrases. (e.g. Software Architect)
  - Avoid making longer and longer labels when making the hierarchy of terms deeper. The label you choose should be concise but still convey the meaning.

Modeling Checklists

Ontology Planning

- ☐ Prepare a written plan that contains at least three main components:
  - ☐ Competency questions that describe specific questions you would like to answer using your ontology
  - ☐ A starting domain for your ontology that includes some classes, and possibly also attributes and relations if these are already known
  - ☐ A few sentences describing the high-level purpose of the ontology

Ontology Implementation

- ☐ Create a new ontology that is appropriately named for the domain in your plan
- ☐ Add all classes from the planned domain to the ontology
- ☐ Ensure that all classes are connected to the focus class with relations, either directly or through other classes
  - These relations may already exist in your plan or they may need to be created based on the semantics of the connection between the classes
- ☐ Refine the ontology by adding additional relations and attributes as needed

Competency Questions

- ☐ Before reviewing the ontology, refine and edit your existing competency questions based on the current ontology in case you made changes
  - Questions should be clear and concise
  - It is totally fine and often helpful to write seemingly obvious questions if they fit your use case
- ☐ Review the ontology to ensure that it can answer each competency question
  - Whenever there is no clear answer to a question, make changes to the ontology to correct this

Documentation

- ☐ Add a description to your ontology metadata
  - This can be the same as the description in your plan
- ☐ Verify or add the current date to the ontology metadata
- ☐ Add your name as well as any other people you worked with to the ontology metadata as contributors

Semantic knowledge modeling drives smart business decisions

metaphactory allows customers to capture explicit, domain-specific knowledge across functional and use case boundaries. Knowledge becomes a shared asset that is captured and published through one central portal and can be consumed by all relevant people, processes, or tools in the enterprise. This significantly increases data literacy and accelerates knowledge democratization.

Our semantic modeling guidelines are a useful resource for learning what modeling is and what its core benefits are. They provide a solid foundation and help you understand what the core elements in the modeling language are, how modeling works and how you can get started.

metaphacts

metaphacts GmbH
Wieslocherstr. 38 | 69190 Walldorf | Germany
www.metaphacts.com