

## Chapter 7

### Formal Upper Ontologies

We have so far looked at ontologies using fairly low-level knowledge representation tools, either one of the conceptual modeling languages or the metamodel based on OWL introduced in Chapter 4. In this chapter we consider the benefits of making use of richer formal ontologies as representation systems.

#### **Structures So Far Not Enough**

In the previous Chapters we have seen that interoperating systems exist in a world of objects, and that these objects can be organized into classes, the classes into subclass structures, and that objects can be organized as wholes with parts. The point of the present Chapter is that the structure so far is not nearly enough.

Remember that our task is to specify ontologies so that we can develop software agents to perform the interoperations more or less automatically. Using the structures we have so far, a world we develop may contain a large number of objects and a rich structure, but it is pretty featureless. An example will make this point clearer.

Consider a human making a shopping trip to a major department store, called say David Jones. In the course of the trip, our human may encounter a variety of objects of different classes. Let us list some of them:

**Table 7.1 Object Classes in Department Store**

David Jones itself	Top	Accounts clerk
Displays	Belt	Dressing Room
Camera	Handbag	Ask clerk for price
Camera Case	Shoes	Clerk tells price
Memory stick	Payment Point	Hand over money
Recharging Dock	Accounts Desk	Clerk gives purchase
Skirt	Floor clerk	Clerk gets item from back room

Let us say that the human purchases the *camera*, *case*, *memory* and *dock*; and also purchases the *skirt*, *top*, *belt*, *handbag* and *shoes*. They try on clothes in the *dressing room*, establish an account at the *accounts desk* assisted by the *accounts clerk*, and make the purchases at *payment points* with the help of *floor clerks*. In the case of the clothing the protocol is *ask clerk for price*, *clerk tells price*, *hand over money* and *clerk gives purchase*. In the case of the *camera*, the floor model is not purchased, so the additional step *clerk gets item from back room* is needed.

Our human sees a rich variety of kinds of classes of object. They know that they can purchase the *camera*, *skirt*, etc, but not the *displays*, *dressing room*, *purchase points*, *clerks*, etc. They know that to make a purchase they must deal with a *floor clerk* at a *purchase point*. Dealing with an *accounts clerk* at the *accounts desk* won't do. In fact they know to deal with the *clerk*, not the *purchase point* directly. And so on.

The human knows all this because they know how to operate in a department store. They generally will have learned as a child in the same sort of way they learn their native language. Should they come from a radically different environment, say one where shopping is done in vast bazaars where each purchase involves intense haggling, they will need to learn how to operate in a department store. (And vice versa, of course.)

Our ontology representation system can allow us to add some structure to the bare list of classes in Table 7.1. We know about the class/subclass structure, so can represent:

- *floor clerk* and *accounts clerk* as subclasses of a more general class *store staff*;
- the various transaction classes as subclasses of a more general class *transaction*; and
- *payment point* and *accounts desk* as subclasses of a more general *business access point*.

We also know about the part/whole structure. So we can model:

- *David Jones* as a whole with parts classes *dressings room, displays, business access point, store staff*;
- *Camera system* with part classes *camera, camera case, memory stick, recharging dock*;
- *Outfit* with part classes *skirt, top, belt, handbag, shoes*;
- *floor purchase transaction* with parts *ask clerk for price, clerk tells price, hand over money, clerk gives purchase*
- *stock purchase transaction* with parts *ask clerk for price, clerk tells price, clerk gets item from back room, hand over money, clerk gives purchase*

And can add the superclass:

- *purchase transaction* with subclasses *floor purchase transaction, stock purchase transaction*

Finally, we know about the set-instance relationship, so can model a class

- *Product* with instances the classes *camera, camera case, memory stick, recharging dock, skirt, top, belt, handbag, shoes*

(Recall the discussion in Chapter 5 of objects which can be both classes and instances.)

Now assume that you want to write a program to interact with an on-line equivalent of David Jones to make the same purchases. Ignoring the differences between being able to physically see and try on things and working from images, assume you are working from an ontology as in Table 7.1 with the additional structures specified above.

Your program will include messages from the *purchase transaction class*, addressed to parts of the *David Jones* whole, concerning instances of instances of the *product* class. Your program will attempt to assemble instances of the *camera system* and *outfit* classes, which involves selecting instances from their respective parts classes.

How does the program distinguish procedures needed to deal with the *purchase transaction* class from those needed to deal with the *camera system* or *outfit* class? How does it know to relate the *David Jones* part classes and *product* classes to *purchase transaction* part instances? How does it know to treat the *David Jones* part classes differently from the *camera system* part classes from the *purchase transaction* part classes? Certainly nothing in the ontology helps. The program knows this because you as a human understand the differences from implicit knowledge, and design the program appropriately.

Now suppose instead of writing a program to interoperate with David Jones, you write a program to interoperate with any of the several thousand players in an e-commerce exchange. It is clearly far too expensive to write a separate program for dealing with each player. The ontology supporting the exchange needs to distinguish different kinds of classes and to represent the different kinds of ways different kinds of classes relate to each other. This Chapter is a start in that direction, but by no means provides a complete solution.

## **Upper ontologies**

All of the ontologies we have discussed so far have been conceived of as in the context of some more or less general application domain. A number of people have been developing ontologies which are conceived of as independent of particular applications, including Cyc, SUMO and WordNet. These ontologies attempt to describe how the world is, and come mainly from the artificial intelligence and natural language processing communities.

We saw in Chapters 2 and 3 that information systems are largely concerned with institutional facts, which are enormously variable. Institutional facts depend very heavily on context and on background. An order entry system can be relied on for inventory control only if all and only movements in and out of inventory are recorded in the order entry system. This requires a set of background practices which prevent un-recorded inventory movements, which can be quite difficult to implement. It took a long time for supermarket bar-code readers at checkout to become sufficiently reliable that the record of sales produced could be used as accurate indicators of stock remaining on the shelves. This accuracy also depends on a rigorous control of shoplifting. So even an object as apparently simple as quantity in stock of a particular product can vary enormously in meaning from system to system. It is therefore