Pushing the Envelope: Challenges in a Frame-Based Representation of Human Anatomy

Natalya F. Noy,[1] Mark A. Musen,[1] José L.V. Mejino, Jr.,[2] Cornelius Rosse[2]

[1] Stanford Medical Informatics
Stanford University
Stanford, CA

[2] Structural Informatics Group
Department of Biological Structure
University of Washington
Seattle, WA

Abstract

One of the main threads in the history of knowledge-representation formalisms is the trade-off between the expressiveness of first-order logic on the one hand and the tractability and ease-of-use of frame-based systems on the other hand. Frame-based systems provide intuitive, cognitively easy-to-understand, and scalable means for modeling a domain. However, when a domain model is particularly complex, frame-based representation may lead to complicated and sometimes awkward solutions. We have encountered such problems when developing the Digital Anatomist Foundational Model, an ontology aimed at representing comprehensively the physical organization of the human body. We show that traditional frame-based techniques such as is-a hierarchies, slots (roles) and role restrictions are not sufficient for a comprehensive model of this domain. The diverse modeling challenges and problems in this project required us to use such knowledge-representation techniques as reified relations, metaclasses and a metaclass hierarchy, different propagation patterns for template and own slots, and so on. We posit that even though the modeling structure imposed by frame-based systems may sometimes lead to complicated solutions, it is still worthwhile to use frame-based representation for very large-scale projects such as this one.

Keywords: knowledge representation, frame-based systems, ontology development, medical informatics

1 Are Frame Formalisms Expressive Enough?

With the recent availability of tools for knowledge-base development that allow domain experts to participate actively in building models of their domains, the models themselves are becoming more detailed and complex. Domain experts include in the models many of the distinctions that knowledge engineers have missed before. Knowledge engineers simply never thought about the domain at such a level of detail and complexity. To an anatomist, for example, a human body is the most complex and intricate mechanism. To a structural engineer, a skyscraper is also extremely intricate and complex. A car designer may see as many parts, relations, and attributes in a car as an anatomist sees in the human body. As a result, models that we represent today become extremely large and complex. Therefore, we must reexamine existing formalisms to evaluate how well they will survive in the era of real-world large models, beyond toy examples from the blocks world.

First-order logic (FOL) provides the richness of representation sufficient to model most commonsense situations, dynamic systems, planning problems, and such medical tasks as diagnosis and treatment of diseases. However, a domain model represented as a set of FOL axioms usually is very difficult for domain experts to understand and does not allow for structured object-oriented modeling of a domain. Frame-based systems provide this structure to modeling, enabling a modular, object-centered representation of a domain, which is intuitive to many people [8]. Furthermore, the limited expressiveness of frame-based systems enables tractable inferencing.

Researchers generally agree that it is almost impossible to build very large knowledge-based systems without a structured approach. In fact, realizing that such a structured approach is essential to building large systems, researchers in FOL have proposed object-oriented approaches to organizing FOL axioms [1]. However, even in comparison with these approaches to FOL modeling, frame-based systems still provide a number of important advantages that are indispensable in building large domain models: (1) knowledge reuse through inheritance and (2) a modeling approach that is cognitively simple, intuitive, and understandable to domain experts.

However, we must pay a price for using frame-based formalisms. First, frames have limited expressiveness compared to FOL. Second, frame-based modeling imposes a certain structure on the representation and this structure sometimes can be awkward or somewhat unnatural.

We have tested the expressiveness and applicability of frame-based formalisms by implementing a large ontology of human anatomy as part of the Digital Anatomist Foundational Model project [19, 20]. The frame-based formalism that we use to encode this ontology is Protégé-2000 [13], a knowledge-modeling environment compatible with the Open Knowledge-Base Connectivity (OKBC) protocol [3]. The goal of the project is to represent declaratively detailed knowledge about the structural organization of the human body. We decided that a frame-based system would be an optimal knowledge-representation formalism to use in this project for several reasons: (1) the primary ontology designers are domain experts (anatomists, in our case) and they must be able to develop, browse through and inspect the ontology; (2) we expect the size of the ontology to be very large (more than 140,000 frames at the time of writing), and (3) the model must capture a large number of complex relationships.

We learned, however, that modeling some of the more complex anatomical relationships, such as adjacency, multiple partitions of the same organ, decomposition of organs into parts that are shared with other organs, and so on, forced us to explore the limits of the expressiveness of a frame-based formalisms. For example, it is not enough to say that one anatomical entity is adjacent to another. We need to specify whether the first entity is to the left or right of the second, whether it is in a superior or inferior location, and so on. A knowledge-based application developed to support a laparoscopic procedure, for example, must have knowledge of the nerve or blood vessel that could be injured if a surgeon inadvertently pushes the scalpel beyond its intended target. Another issue is representing different partitions, or different views, of an object. We can partition an organ based on its structurally distinct components or we can partition it based on its components located in different parts of the body.

Some of the structures we had to introduce to encode this information were counter-intuitive at first sight, hard to explain and understand. We used reified relations, metaclasses and a metaclass hierarchy, and different propagation patterns for template and own slots. While these modelling primitives are standard elements of OKBC and many frame systems support them, some of the more expressive features of a

frame-based formalism, such as metaclasses, are not present in many frame-based systems. The Protégé-2000 knowledge model includes extensive support for metaclasses, so using them was straightforward in this project.

The problems that we discuss in this paper are not unique to the representation of human anatomy. We have encountered similar problems (and suggested similar solutions) in modeling structural relationships in a car, for example.[1] In the end, however, we believe that using a frame-based formalism rather than a more expressive but much less structured representation system based on FOL, was the best choice: Without a structured approach, we could not have succeeded in developing a large-scale, consistent and detailed model that accounted for all the distinctions that we wanted to represent.

We start by introducing the need and the challenges for representing anatomical knowledge declaratively and discussing existing anatomical representations (Section 2). We then present the knowledge model of Protégé-2000 (Section 3). In Section 4, we discuss the alternatives that we considered for developing a comprehensive model of human anatomy, and in Section 5 we describe the model that we developed. We discuss the model in Section 0, present the lessons we have learned in Section 7, and draw our conclusions in Section 8.

2 Representing Anatomy Declaratively

Human anatomy is a fundamental science that underlies all fields of medicine. Most expert-system applications in medicine must have some "knowledge" of anatomy. Therefore, a well-defined, semantically sound, universally agreed upon anatomy ontology could be potentially reused by many applications. These applications include tutoring systems, which lead students of all health professions through their anatomy courses, reference systems for health-care providers, classification terminology for archiving and retrieving medical images, and knowledge-based protocols for configuring the geometry of radiation beams in cancer therapy.

2.1 Existing Models of Human Anatomy

We are not the first ones to discuss the complexity of representing anatomical knowledge in a declarative system. As early as 1988, Haimowitz and colleagues [7] published a paper entitled "Representing medical

knowledge in a terminological language is difficult." They described some of the representational problems they encountered when modeling medical and anatomical knowledge in NIKL [11], a description-logic language. Some of these problems stemmed from the language's inability to represent instances, to deal with multi-valued relations, to represent synonymous terms, number intervals and sequences. Modern frame-based knowledge-representation systems solve some of these issues, but, as the models become more detailed and complex, we are confronted by the next layer of modeling problems, which we describe in this paper.

Several declarative models of anatomy exist today. However, our study of existing terminology systems that include anatomical concepts, such as Terminologia Anatomica [5], source vocabularies of Unified Medical Language System (UMLS) [9], and GALEN [15], has shown that designers of these systems either have failed to state explicitly uniform and consistent classification principles or did not adhere to them. For example, we found that Terminologia Anatomica, the officially sanctioned anatomical term list, does not have a class structure and does not specify the relationships among its terms [17]. GALEN, a clinical-ontology developed with support from European Union, and other clinically oriented controlled terminologies regard anatomical entities as sites of diseases. They tend to forgo representation of structural attributes, which are required for assuring a consistent inheritance hierarchy in the anatomy domain [10].

There are at least two different types of categorization principles present in these classifications: (1) a categorization based on structural and compositional relations among anatomical entities; and (2) a categorization based on functions of anatomical entities. The resulting classifications in the existing anatomical models use one distinction at one level of the hierarchy and a different one at another level. In general, it is very easy to mix these two principles. For instance Terminologia Anatomica groups together the heart, arteries and veins as the "Cardiovascular System", because of the *function* they share in sustaining the circulation. At the same time, some of their subconcepts are grouped according to their *structure*.

Clinical Terms (the Read Codes), a clinical terminology developed for the British National Health Service, adopt a hybrid representation that merges the part-of and is-a relations [21]. For instance, the aorta and its

[1] Personal communication with Dr. Ingo Keutgen and Dr. Ruediger Klein, DaimlerChrysler AG

parts are all classified as “Aortic structure”. This representation does not specify the relationship between the aorta and its parts. In its most recent version, SNOMED, a clinical terminology developed by the College of American Pathologists, has adopted this imprecise representation through its integration with the Read Codes [22].

2.2 Challenges In Declarative Representation of Anatomy

At first glance, modeling the domain of anatomy may seem like a relatively simple task: The number of objects to enumerate is finite (although very large). Moreover, a lot of reference information is available about these anatomical entities in print and in electronic form. However, on a closer examination, construction of a detailed model of the anatomy domain poses an enormous number of knowledge-representation challenges, both in terms of knowledge modeling itself and in terms of knowledge-based environments to support such modeling. Here are some of the modeling challenges: representing complex structural relations, representing different levels of granularity, developing a model that is scalable to a very large number of concepts, and using consistent organizational principles in the model.

We decided to classify anatomical entities in the Foundational Model based on their structure, rather than their function. We made a conscious effort to adhere to this classification principle throughout the hierarchy, thus avoiding a common pitfall of anatomical models, which mix functional and structural classifications in the same hierarchy.

We started by using traditional tools of knowledge-representation formalisms: an is-a hierarchy, slot definitions, and slot restrictions. Almost any modern frame-based system possesses a similar arsenal. However, this simple traditional model does not suffice when we want to represent details about different kinds of meronymic or topological relations.

Detailed classification of part-of relations usually builds on the categorization of part-of relations developed by Winston and colleagues [23] and later by Odell [14]. GALEN, for example, uses and specializes for anatomy the following part-of relations from this categorization: component of an integral object, stuff an object is made from, portion of a mass, a place in the area, and member of a collection [16]. These representations, however, treat part-of relations as atomic relations without any attributes. In the development of the Foundational Model, we learned that a comprehensive model of anatomy must include

attributes for relations. In fact, not only do part-of relations have additional attributes, but also many spatial and topological relations have attributes that need to be modeled.

Many examples we cite in the remainder of this paper are not taken directly from the Foundational Model: We have simplified the anatomical content of the examples to illustrate the knowledge-representation challenges while introducing only a minimal amount of anatomical detail.

2.2.1 Attributed relations

In modeling anatomy, we not only need to represent the part-of relations, but also we need to qualify relations between a part and a whole with additional attributes. For example, parts of an organ can be *shared* (that is, they belong to several anatomical entities) or *unshared* (they belong to one anatomical entity). Blood vessels and nerves that branch within a muscle must be considered a part of both that muscle and the vascular or neural trees to which they belong. In contrast, the fleshy part of the muscle (made of muscle tissue) and the tendon (made of connective tissue) are unshared, since they are parts of the muscle exclusively (Figure 1). Similarly, we distinguish between *anatomical parts*, which are genetically determined, and *arbitrary parts*, which are designated for the convenience of description. For instance, the biceps muscle has several anatomical parts that are structurally distinct from one another: a long and a short head, a belly and a tendon (Figure 1). However, we must often make further distinctions by arbitrarily subdividing a structure in order, for instance, to record the site of an injury with precision. Therefore, we distinguish the intracapsular part of the tendon of the long head of the biceps (which is enclosed within the capsule of the shoulder joint) from the extracapsular part of the same tendon (Figure 1). Structurally, these *arbitrary* parts of the tendon of the long head are identical, yet we must distinguish them in the model since we need to specify a different location, adjacency, and other relations for these arbitrary parts.

Thus, each part-of relation not only expresses that A is a part of B, but also must be qualified with additional attributes. Similarly, each adjacency relation not only expresses that A is adjacent to B, but also must have additional attributes, qualifying the adjacency: left, right, inferior, posterior, and so on. For example, at the elbow (Figure 1A) a separate muscle, the brachialis, is located behind (*posterior to*) the biceps tendon. In fact, to store all the necessary information, we had to add attributes to virtually every structural relation between anatomical entities that we considered.

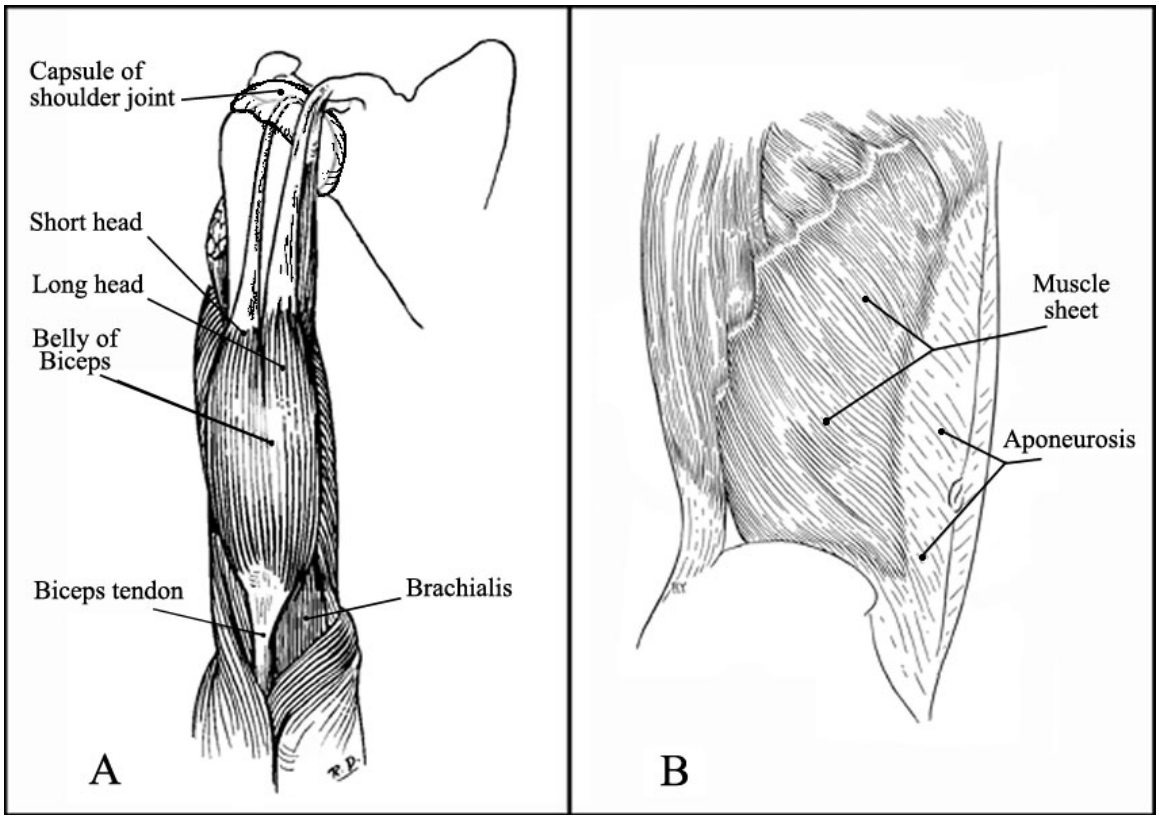

Figure 1. Graphical illustration of the kinds of parts differently shaped muscles can have. A. The arm between the elbow and the shoulder, viewed from the front, with the skin and other superficial structures removed to reveal the underlying muscles. The biceps is a fusiform (spindle-shaped) muscle, which ends below in a round tendon and above it is split into a short and long head, each of which has its own round tendon. B. A sheet-like muscle of the abdominal wall (external oblique) viewed from the right side after removal of superficial structures. The fleshy muscle sheet continues forward toward the midline and the umbilicus as a sheet-like tendon, an *aponeurosis*. Adapted from **[18]**

2.2.2 Inheritance of attributed relations

Attributed relations, such as the ones we have just described, appear at different levels in the hierarchy of anatomical entities. Consider the example in Figure 2. A muscle has a fleshy part and a tendon as its `anatomical unshared` parts. `Muscle-sheet`,[2] a subclass of `Muscle`, has the same `anatomical unshared` parts, but instead of a tendon, it has an aponeurosis, a flat, sheet-like tendon. Similarly, a fusiform muscle has a round tendon and not a generic tendon as one of its `anatomical unshared` parts. That is, the value of the attributed relationship (a particular part in this case) has become more specific, while the attributes (`anatomical unshared`) remained the same.

[2] Throughout the rest of the paper we use a `fixed-width` font for the names of concepts in the model.

Therefore, when attributed relations between classes are defined, their values may not necessarily stay exactly the same for subclasses of those classes.

2.2.3 Enforcement of slot value restrictions

Consider the `Muscle` class in Figure 2 again. Not only do we want to specify multiple attributes for relations and to narrow down these relations at lower levels in the hierarchy, but also we would like to enforce certain restrictions on those values.

Each attributed relation consists of two parts: (1) a frame or frames which are the "value" of the relation and (2) the attributes of the relation. For example, `Tendon` is a "value" of the `has-parts` relation for class `Muscle`, and `anatomical` and `unshared` are the attributes of the `has-parts` relation. In a subclass, we can narrow down the value of the relation (replacing it with a subclass), but we cannot change the value arbitrarily. For example, subclasses of `Muscle`, such as `Muscle sheet` and `Fusiform muscle`, can have subclasses of `Fleshy part` and `Tendon` as their parts. They cannot have a lobe instead of a tendon though.

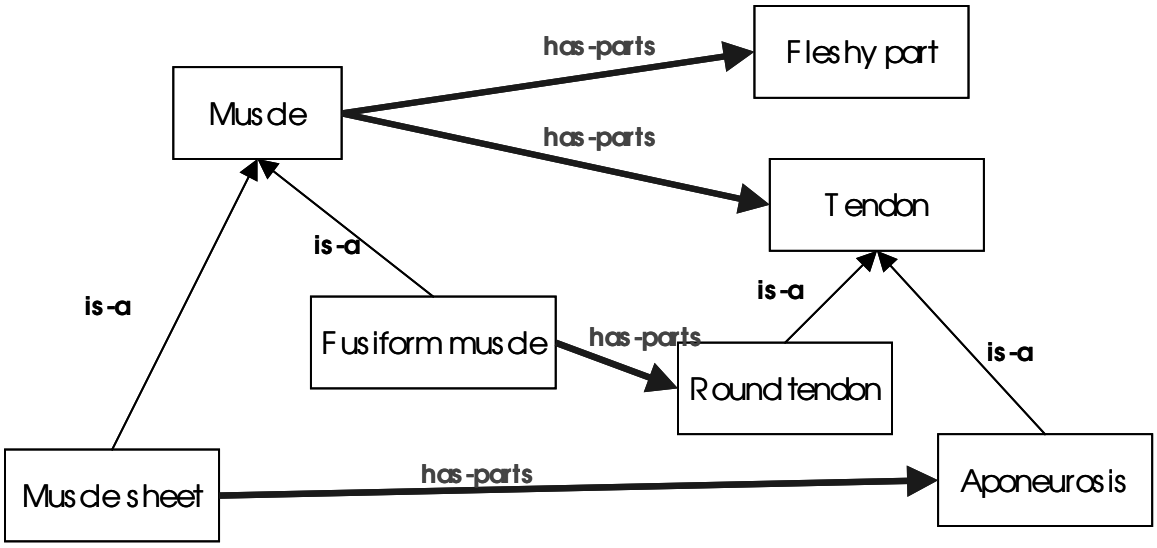

Figure 2. Parts of the `Muscle` class and its subclasses. The `Muscle` class has two parts, `Fleshy part` and `Tendon`. `Muscle sheet` and `Fusiform muscle` are subclasses of `Muscle`. They also have `Fleshy part` as their part. But their tendons are more specific than the generic `Tendon` for the `Muscle` class: `Aponeurosis` (a sheet-like tendon) for the `Muscle sheet` and `Round tendon` for the `Fusiform muscle`. All the parts in the figure are `anatomical unshared` parts.

2.2.4 Representation of levels of granularity

A comprehensive anatomy knowledge base must contain knowledge at very different levels of granularity. We must talk about "canonical" anatomical entities and relations among them (students learn how a prototypical human body looks); we also must be able to represent variations and exceptions from the prototypical state, because no two humans are exactly alike in their anatomy. Furthermore, we must capture differences even in canonical anatomy during different phases of embryological development (e.g., the structure of the embryo's heart changes as it develops). We must be able to extend the model to different species, particularly those that serve as experimental models for diseases such as cancer. Finally, we would like to be able to represent organs and organ parts of specific individuals (e.g., "John's heart").

3 The Knowledge Model

Because such a detailed representation of a domain as we described in Section 2.2 requires a rich knowledge model, we chose a knowledge model that implements the Open Knowledge-Base Connectivity (OKBC) protocol [3]. The goal of the OKBC protocol is to provide uniform access to many frame-based systems and therefore it defines a large variety of components that such systems can have. More specifically, we developed the Foundational Model using Protégé-2000—a frame-based ontology-editing and knowledge acquisition environment [6].[3] The knowledge model of Protégé-2000 [13] satisfies the OKBC protocol. Protégé-2000 has classes, instances of these classes, slots representing attributes of classes and instances, and facets expressing additional information about slots.

Furthermore, Protégé-2000 is aimed at making it easier for knowledge engineers and domain experts to perform knowledge-management tasks. Ontology developers can quickly access relevant information and can use direct manipulation for navigating and managing an ontology.

3.1 Classes, Instances, Slots, and Facets

Classes in Protégé-2000 constitute a taxonomic hierarchy. If a class `A` is a subclass of a class `B` then every instance of `A` is also an instance of `B`. For example, a class representing `Physical anatomical entity` is a subclass of the class `Anatomical entity` (Figure 3) and therefore every physical anatomical entity is also an anatomical entity. In Protégé-2000, both individuals and classes themselves can

be **instances** of classes. A **metaclass** is a class whose instances are themselves classes (see Section 3.3). **Slots** describe properties of classes and instances, such as parts of an organ, or synonyms of a class name. Each slot is itself a frame. In Protégé, as in OKBC, slots are first-class objects: Slots are defined independently of any class. When a slot is **attached** to a frame, it describes properties of that particular frame. For example, we can define a slot `has-parts` and attach it to the class `Anatomical entity`, since any anatomical entity can have parts.

We can specify *constraints* on allowed slot values through **facets**. The constraints specified using facets include cardinality of a slot (how many values a slot can have), restrictions on the value type of a slot (for example, integer, string, instance of a particular class), minimum and maximum value for a numeric slot, and so on. For example, if we want to say that parts of an anatomical entity are instances of other anatomical entities, we will define the slot `has-parts` to have a type `Instance` and **allowed class** for slot values to be `Anatomical entity`. We can **override** the facets for the subclasses of `Anatomical entity`. For example, we may say that parts of a `Physical anatomical entity` are always themselves `Physical anatomical entities` and therefore override the allowed class of the `has-parts` slot at the `Anatomical entity` class to be `Physical anatomical entity`.

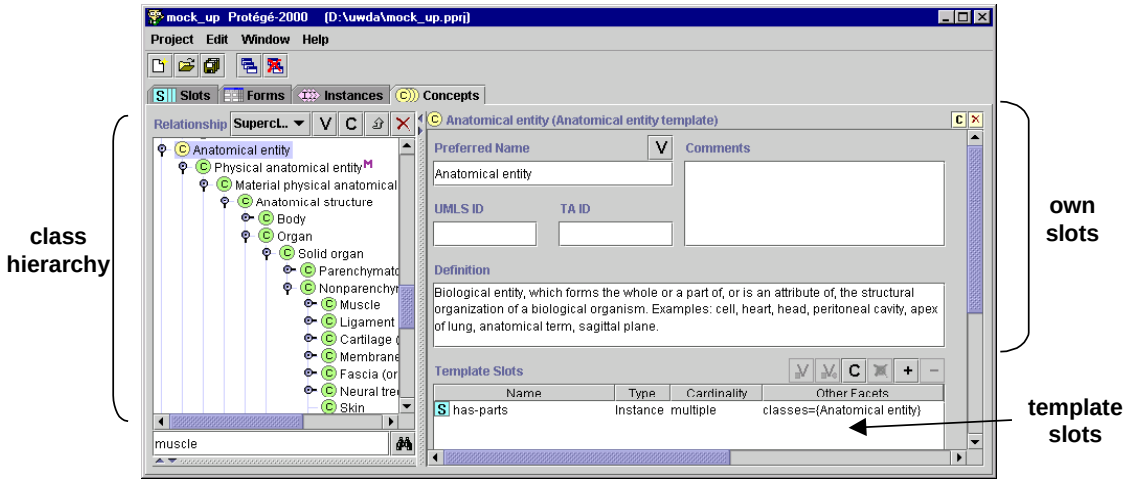

Figure 3. A Protégé-2000 view of the top level of the anatomy-ontology is-a hierarchy. The left-hand pane shows the hierarchy, the right-hand pane shows the definition of the selected class, `Anatomical entity`. The slots `name`, `documentation`, `UMLS ID`, `TA ID`, and `definition` are own slots for the class Anatomical entity. The slot `has-parts` is a template slot.

[3] http://protege.stanford.edu

3.2 Template and Own Slots

A slot can be attached to a frame in one of two ways: as a **template slot** or as an **own slot**. An own slot attached to a frame describes properties of an object represented by that frame (an individual or a class). Own slots attached to a class do not get inherited by the class' subclasses or propagated to its instances. Template slots can be attached only to class frames. A template slot describes properties of the class' instances. A template slot attached to a class is inherited by its subclasses. In addition, a template slot of a class becomes an own slot of the instances of that class (Figure 4).

For example, a slot containing parts of anatomical entity—a `parts` slot attached to a frame representing that individual instance of the class `Anatomical entity`—is an *own slot* attached to that frame (Figure 3).

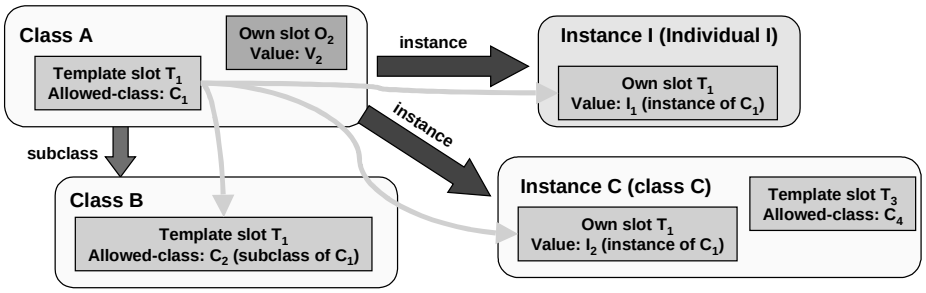

Figure 4. Propagation of template and own slots through the `subclass-of` and `instance-of` relations. The class `A` has both template and own slots. The class `B`, a subclass of `A` inherits the template slots from `A`. Template slots of the class `A` become own slots for its instances, `I` and `C`. Own slots of `A` are not propagated anywhere.

Classes can have own slots as well. For example, `documentation` for a class, which is a textual description of a class, is an own slot attached to that class since it describes the class itself rather than instances of that class. Similarly, if we represent synonyms for each class name, then the `synonyms` slot is an own slot for a class. For example, "Muscle of shoulder girdle" is a synonym for the class name `Muscle of pectoral girdle`. Since both `documentation` and `synonyms` in this example are own slots for a class, these slots and their values do not get inherited by subclasses. Indeed, synonyms of the class name `Muscle of pectoral girdle` are not related to the synonyms of its subclass `Pectoral muscle`. Similarly, bookkeeping information such as the corresponding IDs for the terms from other vocabularies (e.g., UMLS ID for Unified Medical Language System and TA ID for Terminologia Anatomica) are own slots for the classes.

Template slots describe properties that an instance of a class shall have. For example, instances of the class `Anatomical entity` in our example—individual anatomical entities of a specific person—have parts (Figure 3). The slot `has-parts` is a template slot for the class `Anatomical entity`. Every instance of the class `Anatomical entity` has this slot as its own slot with specific values. Any subclass of `Anatomical entity` also will inherit this template slot.

To summarize, own slots describe a property of a (class or individual) frame itself rather than properties of the instances of that frame. Template slots describe properties of instances of a class. Own slots do not propagate to either subclasses or instances of the class to which they are attached. Template slots get inherited as template slots by the subclasses, and they become own slots for instances.

Protégé-2000 does not allow direct attachment of own slots to classes or instances. An individual instance can acquire own slots only by being an instance of a class that has those slots as template slots; a class can acquire own slots only by being an instance of a metaclass that has those slots as template slots.

3.3 Metaclasses

A metaclass is a class whose instances are themselves classes. Template slots defined for a metaclass become own slots for classes that instantiate this metaclass. Thus, a metaclass is a *template* for classes that are its instances. A metaclass describes how a class that instantiates this template will look: namely, which own slots it will have and what are the constraints for the values of these slots. Similarly, a traditional class describes how instances of that class will look: which own slots the instances will have and what are the constraints for the values of these slots.

Just as with regular instances, own slots of a class—the slots that the class acquires from its metaclass—describe the properties of the class itself and not of its instances. For example, a class' `synonyms` pertain to the class itself and not to the class' subclasses or instances.

Protégé-2000 allows users to define their own metaclasses and to define new classes as instances of these user-defined metaclasses. For example, to have own slots for UMLS ID, and Terminologia Anatomica ID for each class, we have defined a metaclass—`Anatomical entity metaclass`—and the corresponding template slots for the IDs (Figure 3). We then define each new class as an instance of `Anatomical entity metaclass`.

4 Considering alternatives

Armed with the representation arsenal of the rich frame-based knowledge model that we have just described, we will now consider several options for representing attributed relations, their specification at different levels of the hierarchy, inheritance of these relations, and, finally, representing different levels of granularity. We will consider using facets to store the attributes, encoding attribute combinations in slot names, reifying relations, and specializing metaclasses.

4.1 Using facets to store the attributes

The first option to represent attributed relations that we will explore is to define facets, which enable us to provide additional restrictions on a binding between a slot and a frame. A *facet* (an *own facet*, to be more precise) describes properties of a slot associated with a frame. For example, a minimum-cardinality facet describes how many values a slot is required to have, a minimum-value facet (for a numeric slot) describes the minimum allowed value for a slot, and so on.

The OKBC specification defines facets as follows [4]:

> *A facet is a ternary relation, and each value V of own facet Fa of slot S of frame Fr represents the assertion that the relation Fa holds for the relation S, the entity represented by Fr, and the entity represented by V*

Thus, a facet value pertains to a frame–slot pair. At first glance, since facets are additional attributes on a frame–slot connection, they may seem to be a good place to store anatomical attributes. We can have a facet on a `has-parts` slot describing whether, for example, a part is arbitrary or anatomical and whether it is shared or unshared (Figure 5a).

However, by definition, facet values pertain to the whole slot not to individual values of a slot. In other words, if all parts of a muscle were anatomical unshared parts, that is, if all values of the slot had the same set of values for their attributes, we could use facets to describe their values. Figure 5b shows the use of facets when a slot has more than one value: There is still only one facet for the whole slot.

In our case, the attribute values are associated with *each* value of a slot and they can be (and often are) different for different values of the same slot for the same frame. For example, the fascia of a muscle is an *unshared* part of the muscle whereas an intramuscular blood vessel is a *shared* part of the muscle. Thus,

facets do not provide sufficient expressivity if different values of the same slot must have different attributes.

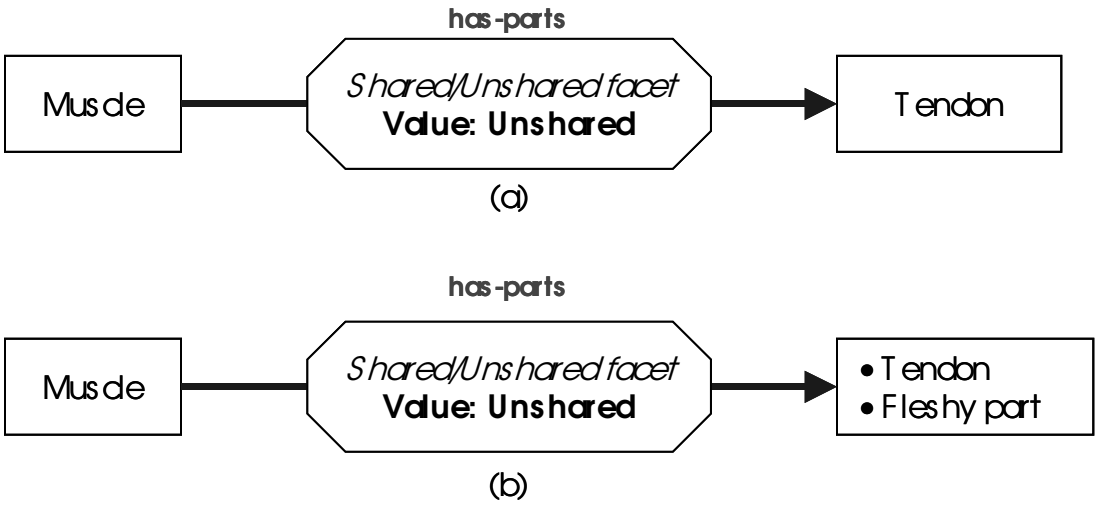

Figure 5. Using a facet to define anatomical attributes. In figure (a) the `has-parts` slot of the `Muscle` class has a `Shared/Unshared` facet associated with it (we use an octagonal to draw a facet). The value of the facet is `Unshared` and the values of the slot is `Tendon`. When we add another value to the slot (`Fleshy part` in figure (b)), the facet pertains to all the values at once.

4.2 Encoding attribute combinations in slot names

Another possible solution is to encode all combinations of attributes in slot names. Let us consider the `has-parts` slot again. Suppose there are only two sets of attributes: anatomical vs. arbitrary and shared vs. unshared. Instead of having a single slot "parts", we can have four different slots: anatomical unshared parts, anatomical shared parts, arbitrary unshared parts, arbitrary shared parts. Then `Tendon` will be one of the values for the first slot, and intramuscular blood vessel will be one of the values for the second slot, and so on.

However, there are a number of problems with having to enumerate all possible combinations of attributes *a priori*:

(a) This solution is not scalable. If we have a large number of attributes for a relation, or each attribute can have a large number of possible values, we get an exponential explosion in the number of slots.

(b) This solution makes it extremely cumbersome to add another attribute. We will need to generate a new set of slots that take into account the additional attribute and move the values from the old slots to the new ones.

(c) With this solution, we cannot specify partial information. If we do not know a value of one of the attributes, we do not know which slot to use.

(d) We cannot process queries such as "get me all shared parts" without having to parse slot names.

4.3 Reifying relations

So far, we have concluded that facets are not expressive enough for our purposes and enumerating all combinations of attributes is not scalable. We can reify attributed relations treating the value of a relation and the corresponding set of attributes as a separate frame in the knowledge base—an instance of a class of relations (Figure 6). Now, instead of using frames representing other anatomical entities as slot values (their parts, their adjacent entities, etc.), we can use a frame that represents the corresponding reified relation.

For example, we can have a class `Part relation`, which will have three slots (Figure 6):

- related anatomical entity (the primary value of the relation)
- anatomical/arbitrary attribute
- shared/unshared attribute

Thus, a value for the `has-parts` slot is an instance of the reified-relation class, `Part relation`.

Recall that anatomical entities themselves are classes and therefore can have two types of slots, own slots and template slots. So, we need to determine whether those instances of reified relation will be own-slot values, template-slot values, or something else.

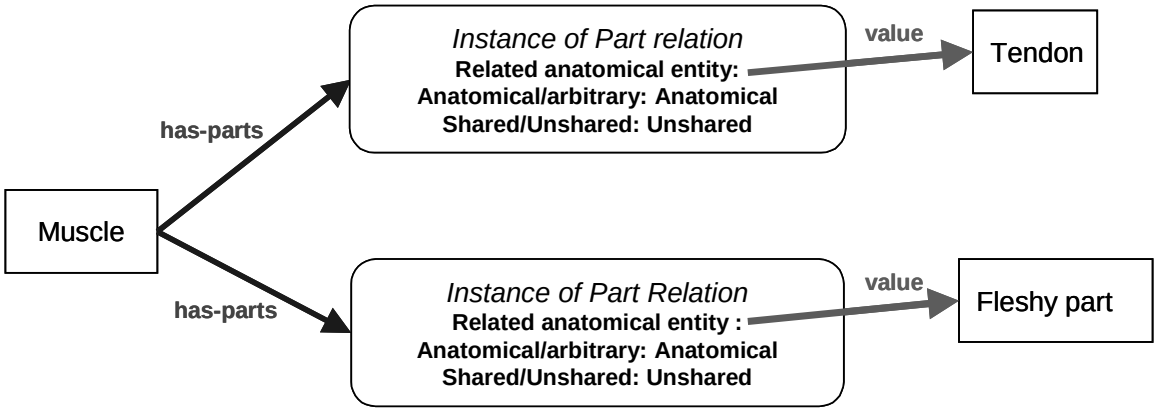

Figure 6. Using reified relations. The values of the has-parts slot for the `Muscle` class are instances of the class `Part relation`. These instances have three slots: `related anatomical entity` (pointing to the actual part), `anatomical/arbitrary, and shared/unshared` (specifying the attribute values).

4.3.1 Reified relations as template-slot values

We can store the instances of reified relations as template-slot values (Figure 7). By definition, if a class A has a value V for a template slot S, then all subclasses of A must have V as the value for their template slot S. Similarly, all instances of A must have V as the value of their own slot S. That is, all template-slot values are inherited *unchanged* by the subclasses. We can add new template-slot values in the subclasses but we cannot alter the inherited ones.

Now recall that, for example, in the case of the `Muscle` class, we do want to have more specific parts in the subclasses (Figure 2): not tendon, but rather either round tendon or aponeurosis. We create new instances of the `Part relation` to have `Round tendon` or `Aponeurosis` in their `Related anatomical entity` slot. However, the `has-parts` slot of the `Fusiform muscle` class would also have the instance of `Part relation` which points to the class `Tendon` as one of its values. Therefore, it will have both parts listed: tendon and round tendon.

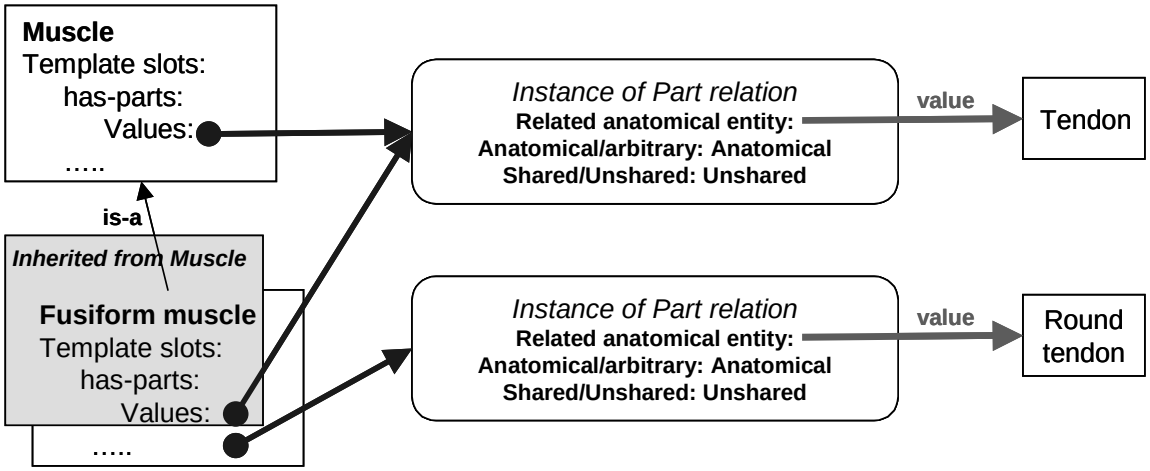

Figure 7. Reified relations as template slot values. The class `Muscle` has a template slot `has-parts`. This slot has an instance of the class `Part relation` as its value. This instance points to a `Tendon` class. The `Fusiform muscle` class, a subclass of `Muscle`, inherits this template-slot value from its parent. We cannot remove or override this inherited value. We can however add a new template-slot value—another instance of the class `Part relation`. This additional instance points to the `Round tendon` class.

4.3.2 Reified relations as own slot values

Another way to store the instances of reified relations in a class definition is as values of an own slot for a class. Own-slot values do not get propagated beyond the class, neither to subclasses, nor to instances of the class (Section 3.2). In fact, it is not even necessary that a subclass of a class has the same own slots as its superclass.

Therefore, to avoid inheriting information that cannot be overridden (as is the case with template-slot values), we can store instances of reified relations as own slot values for our classes. Since there is no relation between the values of own slots in a class and the values of own slots of its subclasses, we have no problem representing parts of `Muscle` (Figure 8). Each of the three classes—`Muscle`, `Fusiform muscle`, and `Muscle sheet`—has its own set of values for the part own slot. In fact, they actually share some of the instances of the `Part relation`, but not all of them.

This complete independence between own slot values of a superclass and those of a subclass could be "too much of a good thing" though: The `Fusiform Muscle` class does not inherit any of the own slot values from its parent, the `Muscle` class. Therefore, the parts of the subclass are not related to the parts of a superclass in any way. This extra flexibility therefore does not allow us to enforce own-slot value restrictions through the hierarchy (cf. Section 2.2.3).

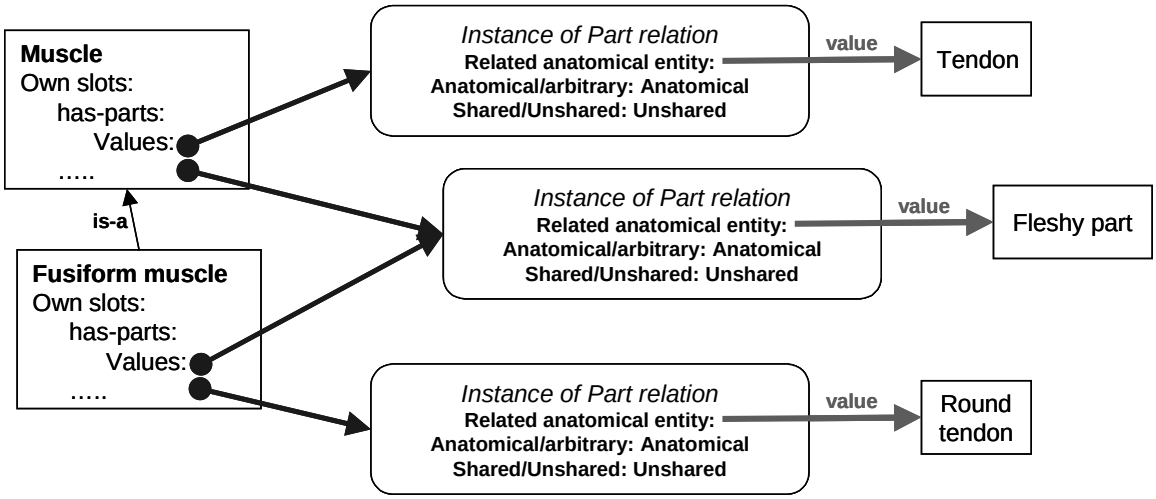

Figure 8. Reified relations as own slot values. The class `Muscle` has two instances of the reified relation as the values of its own slot `has-parts`. The class `Fusiform muscle`—a subclass of `Muscle`—should share one of these instances with its parent. However, `Fusiform muscle` does not inherit own slot values and so we need to specify the shared value again.

4.4 Specializing metaclasses

Recall that we are representing relations as instances of reified relations, and it is a reified relation class itself (such as `Part relation`) that restricts which classes the actual parts belong to. This restriction is specified as an allowed-class restriction for the `related anatomical entities` slot. In our example (Section 4.3.2), any anatomical entity can be related to any other anatomical entity through an instance of this relation. `Anatomical entity` simply restricts its `has-parts` slot to instances of

`Part relation`. However, related anatomical entities for instances of the `Organ` class should be only `Organ parts`. To express this fact, we need to create another class of reified relations, a subclass of `Part relation`, and to override the allowed class for the related anatomical entity to a subclass of `Anatomical entity`, thus making this relation more restrictive. We call this relation `Organ part relation`. However, now we need to define a new metaclass, `Organ metaclass`, the one that would restrict values of the `has-parts` slot to instances of the `Organ part` relation class (Figure 9).

Given that almost at each level in the hierarchy of anatomical entities we must define new slots or new restrictions for slots, we have to "mirror" a large part of the hierarchy of anatomical entities in the hierarchy of metaclasses. That is, for each of the hundreds of classes of anatomical entities we will need to create the corresponding metaclass. This approach is tedious and error-prone since it involves creating a lot of similar definitions and structures.

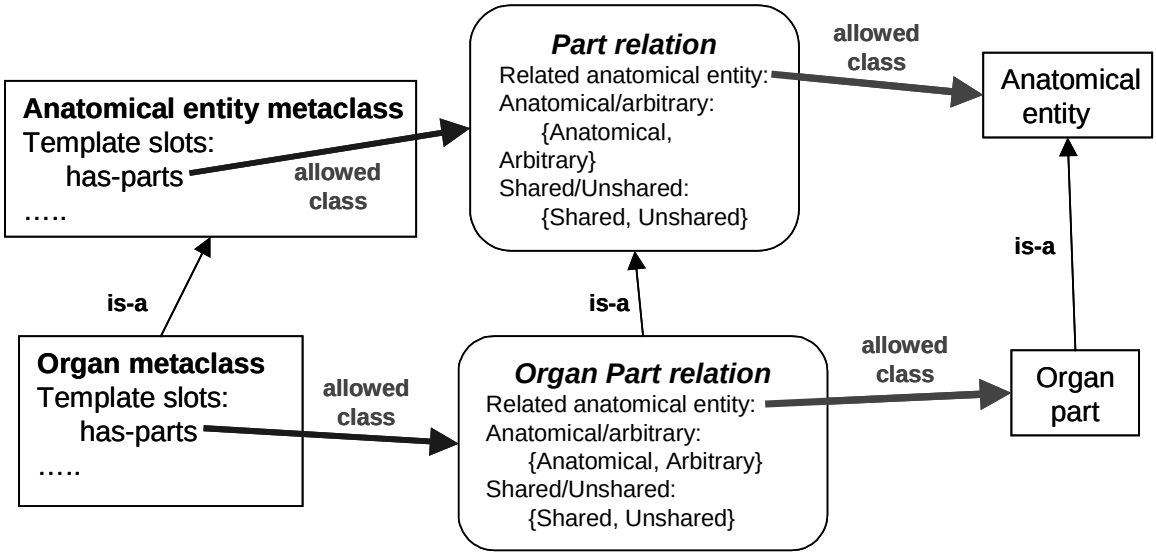

Figure 9. Specializing reified relations. The values for the `has-parts` slot for `Anatomical entity metaclass` are instances of `Part relation`. These instances can have any `Anatomical entity` as the value of their `related anatomical entity` slot. For the `Organ metaclass`—a subclass of `Anatomical entity metaclass`—the values of the `has-parts` slot are instances of `Organ part relation`. The `Organ part relation` class is a subclass of `Part relation` and the values of its `related anatomical entity` slot are restricted to instances of the `Oran part` class.

5 Current State of the Model

Keeping in mind all the considerations that we outlined thus far, we arrive at a model that uses metaclasses and reified relations extensively and satisfies most of the representation requirements we discussed in

Section 2. We developed the current model after experimenting with the alternatives that we described in Section 4. Some of the features of the model at first may be counter-intuitive. However, as we show in this section, they not only correctly represent the details of the model from the point of view of set-theoretic semantics, but also allow us to describe and control precisely the level of detail that our representation requires. Some of the design features on which we will elaborate in this section are:

- All the classes of anatomical entities are also metaclasses and serve as types for their subclasses.
- Each class is both a subclass and instance of its parent.
- Attributed relations are instances of reified relations.
- There is a hierarchy of reified relations.

We interpret the model in terms of set-theoretic semantics typically applied to frame systems: Classes are sets (collections) of objects. Instances of classes are elements of these sets. Subclasses are subsets of these sets.

5.1 Classes

The `Anatomical Entity` class is the root of the hierarchy of anatomical entities.

> *Each subclass of* `Anatomical Entity` *(including* `the Anatomical Entity` *class itself) is a collection, elements of which are collections of similar anatomical entities.*

In other words, each class in the anatomical entity hierarchy is a "set of sets". For example, `Organ` is a collection of different classes of `Organs`, such as solid organs (represented by the class `Solid organ`), cavitated organs, parenchymatous and non-parenchymatous organs and so on (Figure 10).

Now consider the relations between the class `Organ` and the class `Solid organ`. On the one hand, `Solid organ` is a *subclass* of `Organ`: Each instance of `Solid organ` is an instance of `Organ` (each instance of `Solid organ` is a collection of some solid organs, which is also a collection of some organs). At the same time `Solid organ` is also an *instance* of `Organ`: By our definition, instances of `Organ` are collections of types of `Organs`, and instances of `Solid Organ` are collections of some types of organs (namely, solid organs).

The same argument works for any other pair of classes in the hierarchy where one is a subclass of the other.

A is a subclass of `Anatomical entity` ⇒

A is a subset of Anatomical entities ⇒

If B is a subclass of A, B "carves out" some subset of A ⇒

All elements in the collection defined by B are also in the collection defined by A ⇒

B is a (smaller than A) collection of organs of some type ⇒

B is an instance of A

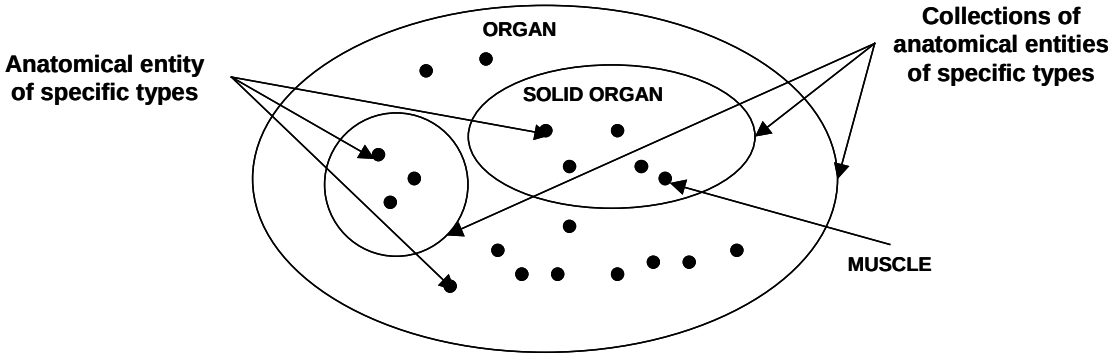

Figure 10. Classes as sets of sets. The `Anatomical entity` class represents a set of anatomical-entity sets. Subclasses of `Anatomical entity`, such as `Solid Organ`, `Muscle`, and so on, are also sets of sets.

Therefore, each class in the `Anatomical entity` hierarchy is both a subclass of its superclass and an instance of its superclass.

Let us consider concepts at the leaves of the hierarchy, such as `Esophagus`. Originally, `Esophagus` was represented as a *subclass* of `Cavitated Organ`. Subclasses (subsets) of the `Cavitated organ` class are *collections* of cavitated organs of specific type. Therefore, if `Esophagus` is a subclass of `Cavitated Organ`, it represents a *collection* of different types of `Esophagus`. This collection could contain a `Canonical Esophagus` and `Variant Esophagus`. The two latter concepts would be *instances* of the `Esophagus` class, since they themselves are not collections of different types of anatomical entities (Figure 11). `Canonical Esophagus` and `Variant Esophagus` are elements of the set represented by the `Esophagus` class.

5.2 Slots

Consider a class in the `Anatomical entity` hierarchy—for example, `Organ`. It has a number of template slots that describe own slots for instances of `Organ` and restrictions on the values of those own slots (Section 3). Consider now the class `Solid organ`, a subclass of `Organ`. `Solid organ` is not only a subclass of `Organ` but also an instance of `Organ` (Section 5.1). Therefore, `Solid Organ` inherits the set of template slots from `Organ` in two ways: (1) they become template slots of `Solid organ` (and their facets can be overridden) and (2) they become own slots of `Solid organ` (and can be instantiated to describe parts, locations, and so on of all `Solid organs`). Note that the two sets of slots that `Solid organ` inherits in two different ways —as own slots and as template slots—are distinct and completely independent from each other. They have very different meaning: The own slots describe the properties of the collection as a whole, in this case a set of all solid organs; the template slots describe which slots elements of the collection (instances of the `Solid Organ` class) will have and what the restrictions on their values are. We can assign values to the own slots and we can override facet values for the template slots. Figure 12 shows a complete definition of the `Muscle` class—a subclass of `Solid organ`—with its template and own slots.

We can also use template slot values: If we know that all instances of `Muscle` will have `Tendon` as one of its parts (and nothing more specific than `Tendon`), we can make an instance of the `has-parts` relation specifying `Tendon` as a template slot value. That value will be inherited by all the subclasses of `Muscle` and will be an own slot value for instances of all the subclasses of `Muscle`.

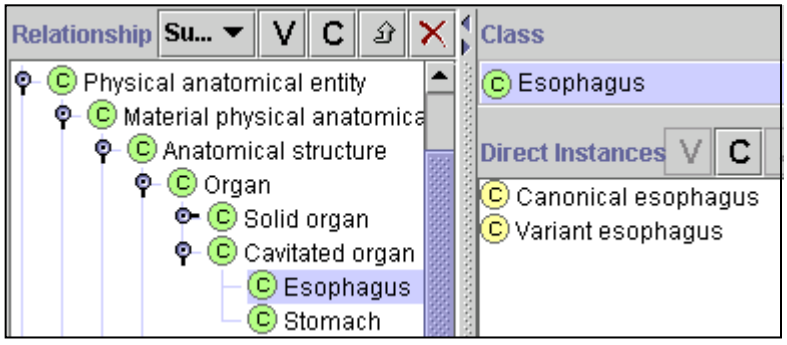

Figure 11. Part of the hierarchy including Esophagus metaclass. Esophagus has two instances (shown in the right column) which are themselves classes: Canonical esophagus and Variant esophagus.

5.3 Instances

In our model, `Esophagus` is a metaclass, therefore, its instances, `Canonical Esophagus` and `Variant Esophagus` would be classes themselves (but not metaclasses). Therefore, we could have John's esophagus as an (individual) instance of a `Canonical Esophagus`, provided it did not exhibit any abnormality or anatomical variation from the canon.

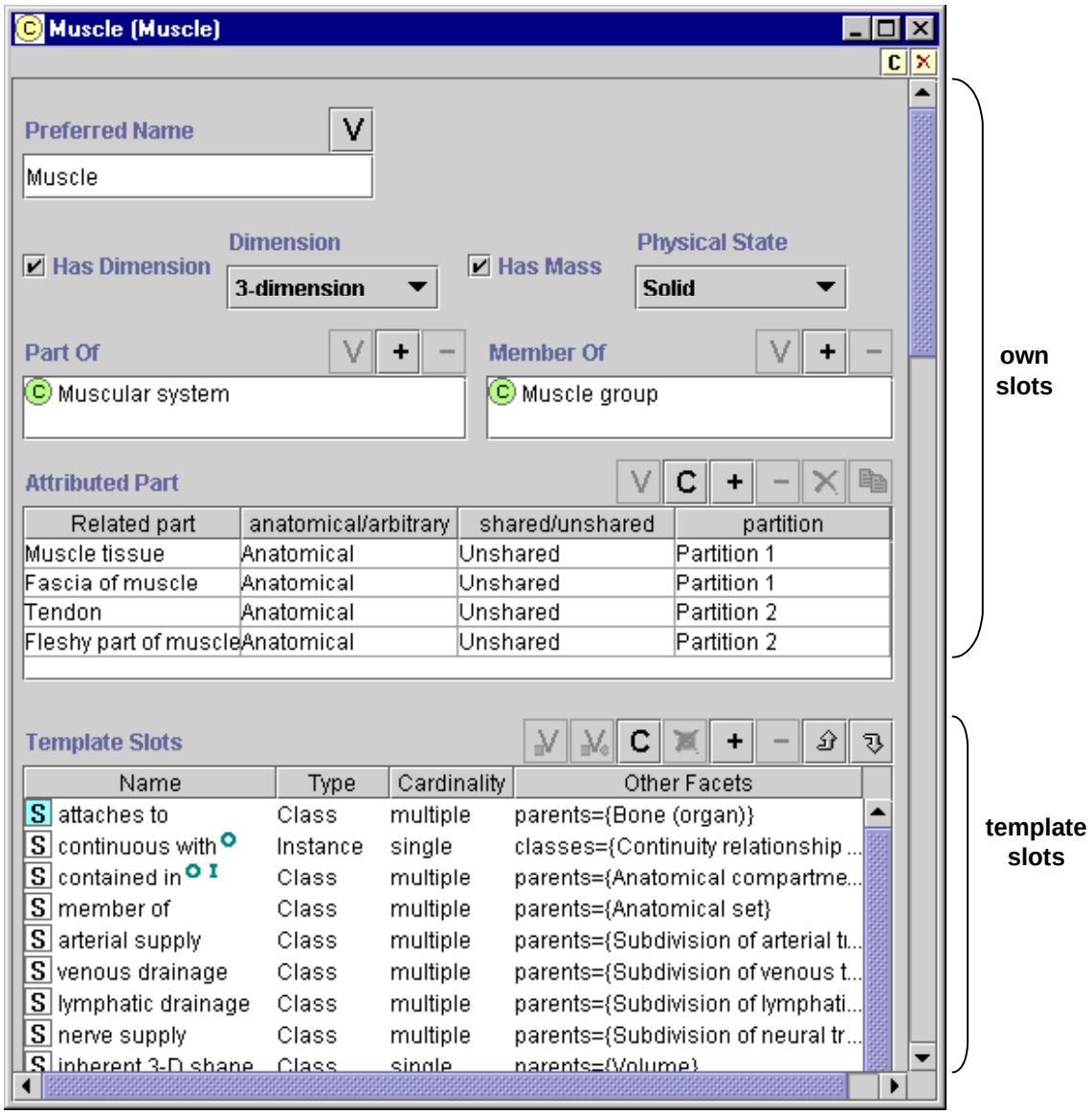

Figure 12. Definition of the `Muscle` class in Protégé-2000. Each line in the table for the `Attributed Part` slot is an instance of the reified `Part relation`. The own slots describe the `Muscle` class itself, which represents a set of different `Muscle` collections. The template slots describe the slots that instances of this collection (more specific collections of muscles) will have.

If we do not want to consider different kinds of esophagus, then esophagus itself should be not a subclass of `Cavitated organ` but rather only its instance.

6 Discussion of the Model

The model that we have presented addresses all the requirements that we have set forth in Section 2.2: representing attributed relations between classes, having classes inherit the values of these relations, enforcing slot-value restrictions, and representing different levels of granularity. It also has another major advantage: It is very flexible and extensible. We can use the same model to represent further details of the anatomy, as well as to represent other species, and other domains. For example, we can use this model if we need to model relations among parts of a car or elements in a building at the same level of detail that we have used for anatomy.

We have not solved all the representation problems however, and we discuss some of the remaining ones later in this section.

6.1 Extending the model further

The model that we have chosen, with reified relations and a metaclass hierarchy, lends itself easily to further extensions in the domain representation. For example, we had to describe different partitions of an organ. One way to divide an organ, such as a muscle, is using its anatomical parts, such as fascia, connective tissue, blood vessels and nerves. We can also define a different partition of muscle, or more specifically, a biceps, into long head, short head, belly and tendon. There can be other possible partitions as well. In other words, there is more than one way to "cut up" a whole into parts. In fact, this situation is quite common not only in anatomy but also in other domains. In addition, we may not know in advance how many different ways exist to partition an object.

We represented partitions simply as additional attributes of instances of reified relations for the `has-parts` slot: Fascia and connective tissue are anatomical unshared parts of muscle belonging to the same partition (Figure 12). Having represented different partitions declaratively, we can express additional axioms such as: "all parts in the same partition must have the same value for the slot that defines whether the partition is arbitrary or anatomical".

Similarly, some of the concepts and relations, and some of the partitions, exist only during certain stages of human embryological development. For example, an embryo has *limb buds*, which do not exist in later stages of development, but which give rise to all parts of the fully developed limbs. We can again use attributes for the relations to indicate that partitions involving a limb bud exist only at a particular developmental stage.

Likewise, the multiple levels of granularity in the model will help us in the task of extending the Foundational Model to model the anatomy of other species. For example, we can have additional own slots for classes indicating whether a class is present in all mammals or only in particular ones. We can reuse a metaclass hierarchy and change own slot values describing specific locations and attributes for different species.

6.2 We haven't solved all the problems

There are still some modeling challenges that we need to address. One such challenge is the propagation of own-slot values. Consider the example in Figure 8 again. Instances of reified relations are own-slot values. The own slot `has-parts` for class `Muscle` has two instances of `Part relation` as its values. One of those instances (fleshy part) is also a value for the `has-parts` slot for `Fusiform muscle`, a subclass of `Muscle`. However, in general, own-slot values in a class have no relation to the values of the same own slot in a superclass. Therefore, when we define the parts for `Fusiform muscle` and `Muscle sheet`, these classes do not inherit any of the specific `has-parts` information from the `Muscle` class and we need to enter them again.

Ideally, a subclass should by default get the same own-slot values for relation slots as its superclass has. A domain expert could then edit these values, making them more specific (e.g., replacing the `Tendon` class with the `Round tendon` class for the `Fusiform muscle`). However, there is nothing in the knowledge model that would allow such propagation. Protégé-2000 has a plugin mechanism which allows one to build plugin code to perform domain-specific knowledge acquisition [12]. We could build a plugin that would automatically propagate the values. Another plugin could monitor the change of values in the subclass to ensure that values are made only more specific and not changed completely.

A partial solution at the knowledge-model level is to use template-slot values. Recall that template-slot values are, in some sense, the opposite of own-slot values: They are inherited by subclasses and instances of a class and they cannot be changed or overridden. We can, however, add more values if the cardinality of a slot allows more values. Therefore, once we reach a part that will be the same in all subclasses of a class, we can make this part (in fact, the reified relation linking to this part) not only an own-slot value for the class but also a template-slot value. Then all subclasses of this class (since they are also instances of this class) will get this value as one of the values for the respective own (and, of course, template) slot.

7 Lessons Learned

We will concentrate on two lessons we have learned from this project: (1) expressivity requirements for knowledge-representation systems and (2) requirements for knowledge-representation tools.

7.1 Expressivity Requirements for Knowledge-Representation Systems

We started with recalling the paper by Haimowitz and colleagues [7] positing that "representing medical knowledge in a terminological language is difficult." As our discussion has shown, we wholeheartedly agree with this thesis. Every time when we thought we have come up with a representation that is capable of encoding all the required facts, we realized that there was some other aspect that this representation did not yet capture.

Knowledge-representation systems range in their expressivity from frame-based systems with limited expressivity but tractable inference (e.g., description logics) to FOL, which allows maximum expressivity but does not lend itself to tractable or even decidable reasoning procedures. Some researchers argue that there is no need for anything "in between": Once you give up some inference properties, you might as well go to full FOL. However, for many models it seems that the "right" expressivity is somewhere in the middle [2].

We cannot define everything we want in a traditional limited frame-based system. Moreover, we cannot satisfy many of the representation requirements that we have presented in a description-logic (DL) system. Medical terminologies such as GALEN and SNOMED-RT that rely on DL in their representation do not address the issue of attributed relations at all. It is unclear how this issue could be resolved adequately within the confines of a DL system. First, to preserve efficient reasoning, DLs strictly separate classes and

instances. A slot value or a restriction on a slot value for a class cannot contain an instance. Therefore, class definitions cannot contain instances of reified relations. Second, DLs usually do not have metaclasses. As a result, we cannot easily add such slots as synonyms, corresponding IDs from various terminologies, comments, and so on to class definitions.

However, we do not need to go to full FOL either. A knowledge-representation system that includes metaclasses and allows use of instances in class definitions (like many OKBC-compatible systems) is adequate for representing even the most complex anatomical relations. The representation may be not the most intuitive one and may at times seem awkward. At the same time, we still preserve some of the main advantages of frame-based representation: modularity, object-centered approach, and scalability.

Furthermore, the reuse of knowledge through inheritance and object-oriented organization that frames provide proved essential in developing a large-scale ontology. The sheer size of the Foundational Model (which has more than 140,000 frames already) imposes stringent scalability demands on a knowledge-representation system: It must be able to handle knowledge bases with tens and hundreds of thousands of concepts. In fact, we cannot imagine developing an ontology of such size in a knowledge-representation system that lacked these features.

In addition to being more scalable than full FOL, frame-based knowledge-representation systems have another advantage: They are much easier to understand for domain experts than FOL. In fact, many members of our team are anatomists with no formal training in artificial intelligence or knowledge representation. The anatomists can not only understand the model that we present here but also can use and extend it.

7.2 Requirements For Knowledge-Representation Tools

In this paper, we have discussed several alternative designs of the system. In practice, we implemented most of those designs before realizing their limitations. Therefore, we needed a knowledge-representation environment that allowed easy migration from one representation to another. Protégé-2000 proved to be such an environment. In Protégé, we can easily perform many changes by direct manipulation (e.g., drag and drop) and we can see the results immediately. Protégé's ease of use allowed for fast prototyping of alternative approaches, which proved invaluable: With the Foundational Model having more than 60,000

classes, we needed to test the approach on smaller parts of it before transforming the complete knowledge base to a new model.

For complex models such as the one presented here, another invaluable feature of a knowledge-representation environment is the ability to hide some of the complexity of the representation behind the user interface. Again, Protégé provides such ability. It has a plugin architecture that allows developers to write customized widgets, which are tailored for a particular domain or ontology. Consider the definition of `Muscle` in Figure 12 for example. In the model, the `has-parts` own slot is a list of instances of the class `Part relation`. Protégé-2000 has a custom-designed slot widget that enables us to view and edit, in a single table, all the instances of `Part relations`, which are values of the parts slot. Therefore, the widget hides from the user that the rows in the table are actually instances of the reified-relation class. This separation between the internal representation and the representation in the graphical user interface can hide several levels of complexity from the user.

In fact, we do not expect domain experts (anatomists in this case) to understand all the intricacies of the morel we presented in this paper. The user interface for creating and instantiating a hierarchy of metaclasses in Protégé-2000 looks almost identical to the interface for creating and instantiating regular classes. We have used the plugin mechanism to implement an extension that automatically assigns metaclasses according to the model we described. Slot widget hide the extra level of indirection introduced by reified relations. As a result, an anatomist can browse and extend the ontology while being oblivious to many set-theoretic explanations and implications.

8 Conclusions

We have tested the applicability of frame-based modeling for generating a complex domain model by developing a large-scale ontology of human anatomy. We have addressed the representation of attributed relations; their inheritance and the propagation of slot values; and enforcement of slot-value restrictions in such a model. Similar problems arise in detailed modeling of many other domains, from car manufacturing to the anatomy of other species. In order to represent all the richness of structural anatomical knowledge, we had to employ the full-power of OKBC-compatible frame formalisms such as metaclasses, own and template slots, and facets. We have shown that such formalisms are expressive enough to describe some of

the very intricate and complex elements of relations among anatomical entities, multiple views on partitioning an entity, and different levels of granularity in the representation. At the same time, these formalisms preserve core advantages of frame-based systems with set-theoretic semantics that make these systems usable for large-scale and multifaceted modeling projects: reuse of knowledge, cognitive simplicity, object-orientation, and scalability.

Acknowledgments

The project of this magnitude would not have been possible without help of many individuals. Augusto Agoncillo is an active developer of the Foundational Model and he is an active source of many challenging representation problems in anatomy. Ray Fergerson is the principal developer of the Protégé-2000 environment and he spent many days and weeks ensuring that Protégé-2000 can handle the requirements that the complexity and the size of the model posed. Peter Mork and Rada Chirkova provided very useful feedback on drafts of this paper helping make it more clear and understandable. This work was supported in part by contract LM13506 and grant LM06316, National Library of Medicine.

References

1. Amir, E., Object-Oriented First-Order Logic. *Electronic Transactions on Artificial Intelligence*. **3**(Section C): p. 63-84, 1999.
2. Bechhofer, S., Goble, C. and Horrocks, I. DAML+OIL is not Enough. In: *The First Semantic Web Working Symposium*. Stanford, CA, 2001.
3. Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P. OKBC: A programmatic foundation for knowledge base interoperability. In: *Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. Madison, Wisconsin: AAAI Press/The MIT Press, 1998.
4. Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P., *Open Knowledge Base Connectivity 2.0.3*., 1998.
5. FCAT, *Terminologia Anatomica*: Federative Comittee of Anatomical Terminology, 1998.

6. Grosso, W.E., Eriksson, H., Fergerson, R.W., Gennari, J.H., Tu, S.W. and Musen, M.A. Knowledge modeling at the millennium (the design and evolution of Protégé-2000). In: *Twelfth Banff Workshop on Knowledge Acquisition, Modeling, and Management*. Banff, Alberta, 1999.

7. Haimowitz, I.J., Patil, R.S. and Szolovits, P. Representing medical knowledge in a terminological language is difficult. In: *12th Symposium on Computer Applications in Medical Care*: IEEE Computer Society Press, 1988.

8. Karp, P.D., *The design space of frame knowledge representation systems*, SRI AI Center: Menlo Park, CA, 1993.

9. Lindberg, D.A.B., Humphreys, B.L. and McCray, A.T., The Unified Medical Language System. *Methods of Information in Medicine*. **32**(4): p. 281, 1993.

10. Michael, J., Mejino, J.L.V. and Rosse, C., The role of definitions in biomedical concept representation. *Journal of American Medical Informatics Association, AMIA'2001 Symposium Supplement*: p. 463-467, 2001.

11. Moser, M.C., *An overview of NIKL, the new implementation of KL-ONE*, Bolt, Barnek and Newman, Inc., 1983.

12. Musen, M.A., Fergerson, R.W., Grosso, W.E., Noy, N.F., Crubézy, M. and Gennari, J.H. Component-Based Support for Building Knowledge-Acquisition Systems. In: *Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress (WCC 2000)*. Beijing, China, 2000.

13. Noy, N.F., Fergerson, R.W. and Musen, M.A. The knowledge model of Protégé-2000: combining interoperability and flexibility. In: *12th International Conference on Knowledge Engineering and Knowledge Management (EKAW-2000)*. Juan-les-Pins, France: Springer-Verlag, 2000.

14. Odell, J., *Six different kinds of aggregation*, in *Advanced Object-Oriented Analysis & Design Using UML*. 1998, Cambridge University Press. p. 139-149.

15. Rector, A., Gangemi, A., Galeazzi, E., Glowinski, A. and Rossi-Mori, A. The GALEN CORE Model Schemata for Anatomy: Towards a Re-Usable Application-Independent Model of Medical Concepts. In: *Medical Informatics Europe, MIE'94*, 1994.

16. Rogers, J. and Rector, A. GALEN's Model of Parts and Wholes: Experience and Comparisons. In: *AMIA Annual Symposium*. Los Angeles, CA, 2000.

17. Rosse, C., Terminologia Anatomica: Considered from the Perspective of Next-Generation Knowledge Sources. *Clinical Anatomy*. **14**: p. 120-133, 2001.

18. Rosse, C. and Gaddum-Rosse, P., *Hollinshead's Textbook of Anatomy*. Philadelphia: J.P. Lippincott Publishers, 1997.

19. Rosse, C., Mejino, J.L., Modayur, B., Jakobovits, R., Hinshaw, K.P. and Brinkley, J.F., Motivation and Organizational Principles for Anatomical Knowledge Representation: The Digital Anatomist Symbolic Knowledge Base. *Journal of the American Medical Informatics Association*. **5**(1): p. 17-40, 1998.

20. Rosse, C., Shapiro, L.G. and Brinkley, J.F., The Digital Anatomist Foundational Model: Principles for Defining and Structuring its Concept Domain. *Journal of American Medical Informatics Association, AMIA'98 Symposium Supplement*: p. 820-824, 1998.

21. Schulz, E.M., Price, C. and Brown, P.J.B., Symbolic anatomic knowledge representation in the Read Codes Version 3: structure and application. *Journal of American Medical Informatics Association*. **4**: p. 38-48, 1997.

22. Spackman, K.A., ed. *SMOMED® RT: Systematized Nomenclature of Medicine, Reference Terminology*. College of American Pathologists: Northfield, IL, 2000.

23. Winston, M.E., Chaffin, R. and Hermann, D.J., A Taxonomy of Part-Whole Relations. *Cognitive Science*. **11**: p. 417-444, 1987.