Binding Ontologies & Coding systems to Electronic Health Records and Messages

AL Rector MD PhD[1], R Qamar MSc[1] and T Marley MSc[2]

[1]School of Computer Science, University of Manchester, Manchester M13 9PL, UK

[2]Salford Health Informatics Research, University of Salford, Salford, UK

Corresponding Author:

Alan L Rector

School of Computer Science, University of Manchester

Manchester M13 9PL, United Kingdom

TEL: +44 161 275 6149/6188

FAX: +44 161 275 6204

email: rector@cs.man.ac.uk

Citation:
***Journal of Applied Ontology Volume* (1) pp 51-69**
(ISSN1570-5838 (Print) 1875-8533 (Online) DOI: 10.3233/AO-2009-0063)

Abstract

A major use of medical ontologies is to support coding systems for use in electronic healthcare records and messages. A key task is to define which codes are to be used where – to bind the terminology to the model of the medical record or message. To achieve this formally, it is necessary to recognise that the model of codes and information models are at a meta-level with respect to the underlying ontology. A methodology for defining a Code Binding Interface in OWL is presented which illustrates this point. It generalises methodologies that have been used in a successful test of the binding of HL7 messages to SNOMED-CT codes

Keywords

Medical Terminology, Electronic Medical Records, OWL, Ontologies

1. Introduction

A major use of medical ontologies is to support medical terminologies and coding systems. A major use of medical terminology and coding systems is for electronic healthcare records (EHRs) and messages. Specifying the validation rules for how terminology and coding systems are to be used in electronic healthcare records and messages is, therefore, a key problem for medical ontologies.

We contend that electronic healthcare records and messages are data structures and refer to their models as "information models". By contrast, we contend that the model of meaning or "ontology" is a model of our conceptualisation of the world – of patients, their illnesses, treatments, etc. The function of the information models is to make it possible to specify and test the validity of data structures – *i.e.* whether or not they conform the their specifications – so that they can be exchanged and re-used in different information systems. The function of the model of meaning to represent accurately our understanding of the world so that we can reason soundly about the world in general or individual patients and their diseases in particular. Validity neither requires or guarantees accuracy and soundness, nor *vice versa*.

The interaction of information models and coding systems is a notorious source of ambiguity and errors in clinical systems. Typically, groups, usually from standards bodies, develop generic information model, *e.g.* the HL7 RIM[1] (Schadow *et al.*, 2000) or the OpenEHR reference model[2]. From these generic models they derive detailed models of messages, electronic healthcare record (EHR) entries, and related data structures. Other groups develop coding systems, with or without associated ontologies – *e.g.* SNOMED, ICD, CPT, MEDRA, etc. The problem is then how to use a given information model with a given coding system. Furthermore, it is often required that the same model be used with more than one coding system and the same coding system with more than one model. Hence, there is a need for a means to define formally the interface, or "binding," between an information model and a coding system, analogous to an "Application Programming Interface" or "API" between software modules. We call this a "Code Binding Interface" ("CBI").

One aspect of this problem is often expressed as defining "value sets" or "code sets" or just "subsets" for use with information models. For example, we might wish to specify which codes can be entered in the family history section of the record or the list of valid codes for "position" for a blood pressure measurement. For a coding system such as SNOMED-CT or GALEN that allow formal definitions by means of expressions, this includes the constraints on such expressions. The Archetype Definition Language (Beale, 2002), used by the CEN standard EN13606 and OpenEHR, specifies an "ontology section" similar in principle to what we here call a Code Binding Interface. However, it currently provides mechanisms to cover only the simplest cases, where there is a one-to-one mapping between the local codes within the archetype and specific codes in the coding system.

We contend that one issue in developing proper descriptions of the use of codes in information models is a frequent misunderstanding of the relationship between an "ontology" – a formal representation of our understanding of the meaning in terms of our understanding of the world – and coding systems – meta models of the relations between symbols ("codes") that represent those meanings. This is particularly important as coding systems based on ontologies – *e.g.* SNOMED-CT and the NCI Thesaurus, are coming into widespread use. We outline in this paper an account of the relation between information models, coding systems and ontologies and one method of implementing a Code Binding Interface

[1] http://www.hl7.org

[2] http://www.openehr.org

2. Ontologies, Codes, and Information Models

2.1 Overview

The process of using Electronic Health Records (EHRs) – or any similar data structures – can be seen as taking statements about our understanding of the world that we believe to be accurate, encoding them into data structures that must be valid, processing, storing or transmitting those data structures, and then re-interpreting those data structures as statements about the world. If the process is successful, the validity of the data structures ensures that the statements about the world are not corrupted by the intervening processing, storage and transmission of the data structures, *i.e.* that the processing has not compromised the accuracy of the reinterpretation of the data structures as statements about our understanding of the world. This is shown diagrammatically in Figure 1.

Figure 1:

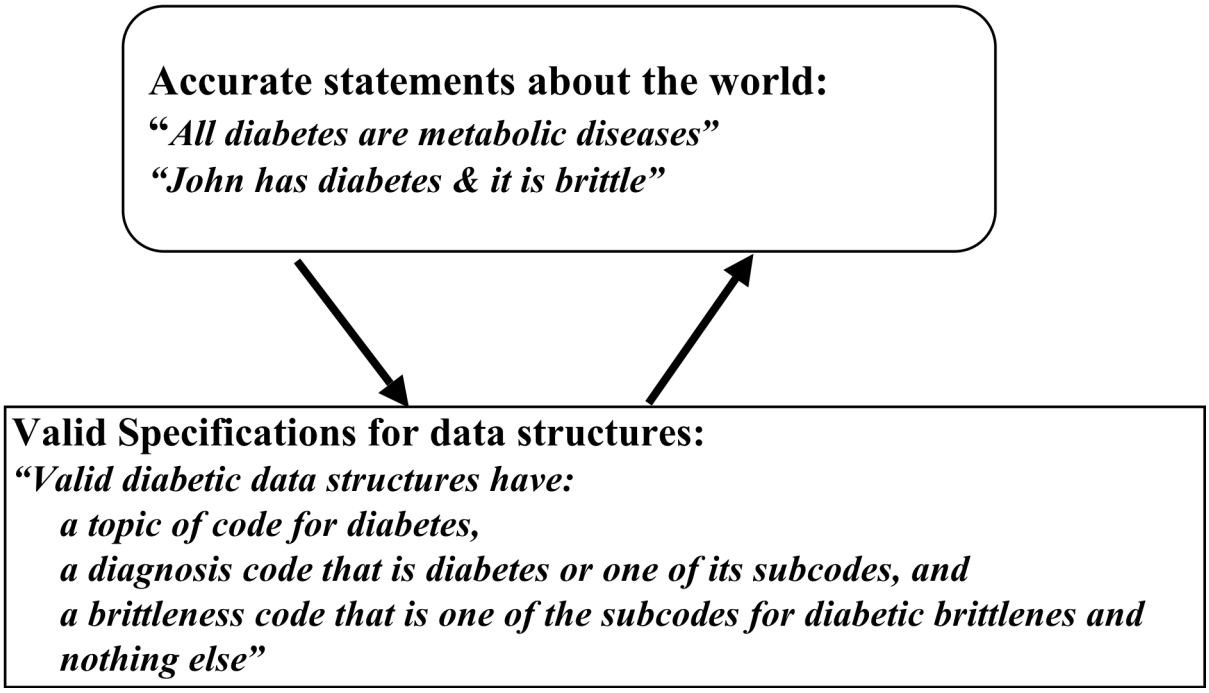

Figure 1: Encoding of accurate statements about the world in data structures and then their reinterpretation.

While the above statement may seem self evident or even pedantic, we suggest that much of the difficulty in dealing with issues of medical terminology and medical records follows from conflating the two levels and not making the processes of encoding and re-interpretation explicit. We contend that codes are also data structures – or more precisely symbols to be used in data structures – and that the model of codes is also at the level of the information model. We take the individuals in the model of meaning to represent patients and their illnesses. We take the individuals in the model of codes to be symbols representing classes of illnesses. (Typically, we term classes of illness "conditions" or "disorders".) In other words, the codes are the symbols in the information model that represent the entities in the model of meaning.

In an ideal system, the coding system would be derived from the ontology. The subcode/supercode relation amongst the individual codes in the coding system would mirror the necessary subsumptions amongst classes of entities the world inferred from the axioms in the ontology. Indeed the coding system may be thought of as a meta-model of the ontology. This idealised relation is shown in Figure 2a. The classes and subclasses of patients' disorders and other entities in the world – represented in the ontology – correspond to the subcode-supercode relationship between codes in the coding system, represented by the property is_subcode_of.

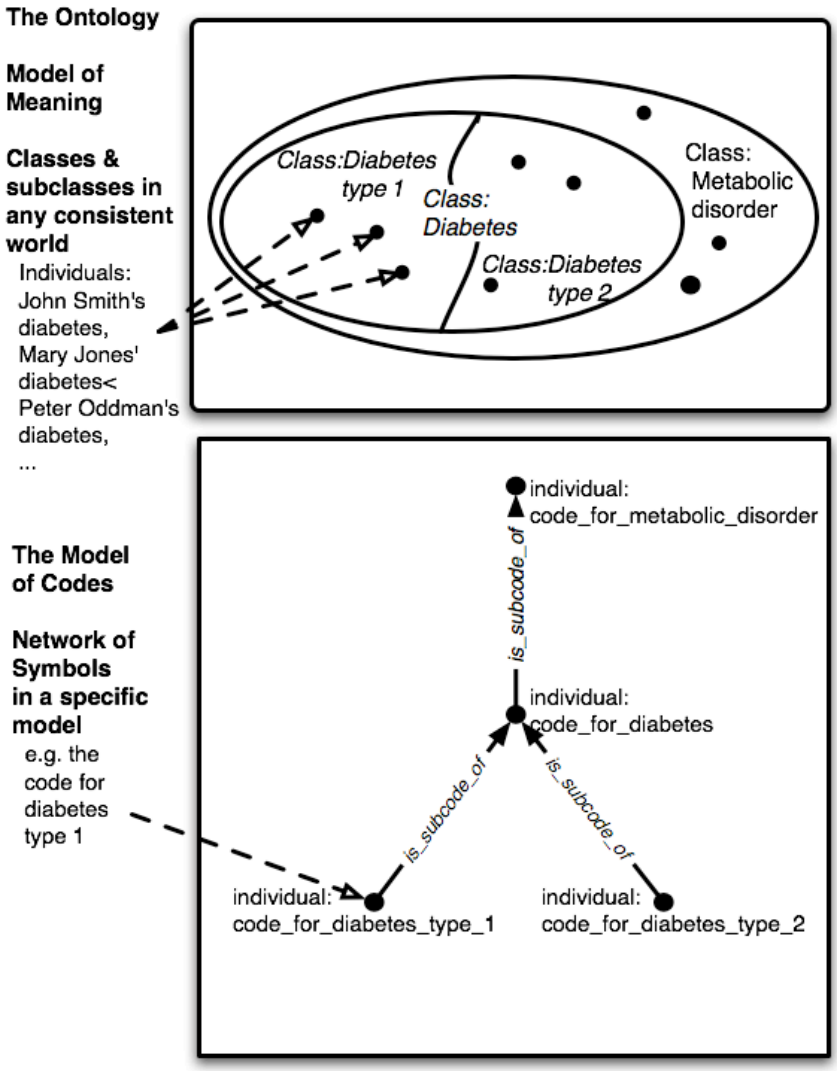

Figure 2a: The ontology – sets of entities in any consistent world – is mirrored by the model of codes – a network of symbols in a specific model

In Figure 2a, the is_subcode_of property precisely mirrors the subclass structure in the model of meaning and was derived from it by means of a systematic transformation. However, from the point of view of formal reasoning, each model is treated separately. Once the model of codes is generated, the inference that code_for_diabetes_type_1 is a member of the class Diabetes_and_its_subcodes is independent of the inference that, in the model of meaning, Diabetes_type_1 is a subclass of Diabetes. This separation avoids any possibility of the classic paradoxes of self-reference.

Why the apparent duplication? There are both theoretical and practical reasons.

- *Theoretically* – codes are not conditions and data structures are not patients. There are things that can be said of codes and data structures that are nonsense if said of conditions and patients, and *vice versa*. For example, both HL7 and OpenEHR have attributes in their data structures for "negation indicators". Clearly, data structures have negation indicators; patients do not. It makes sense to talk about whether a patient has, or does not have, diabetes. It makes sense to talk about whether a data structure has its negation indicator set to true, false or null.
- *Pragmatically* – existing coding systems and information models contain many idiosyncrasies and errors. Many coding systems are based on a flawed, or even no, model of meaning. Separating the information model and coding system from the model of meaning provides a degree of indirection that allows developers to compensate for these failings without compromising the underlying model of meaning.

2.2 Details and consequences for selecting codes

The information model and codes are about data structures and symbols. Instances of these models are data structures to be validated – messages, database entries, EHRs, etc. Model of data structures – HL7 messages, Archetypes, and other information models – are analogous to grammars rather than knowledge bases. This is most obvious where they contain many features that are clearly about their internal structure rather than the world. For example, mood codes in HL7 and tenses in natural languages are features of how the data structures or language utterances convey information about time and aspect. Events in the world, do not have moods or tenses; they occur in the past, the present or are proposed for the future. Likewise, notions such as "missing" and "null" make sense in terms of data structures, but not in terms of patients and their diseases. There is no such thing as a patient with a "missing blood pressure".

By contrast, the ontology itself is intended to represent the world, or our conceptualisation of it. The ontology is a form of knowledge base – the generic background knowledge needed to interpret specific information about patients, their diseases and care.

More precisely, the ontology represents all "worlds" – *i.e.* anything that could happen – that are consistent with its axioms. Ontologies may be diagrammed, as in Figure 2a, as sets and subsets, but the relationships in the ontology are is implied in such diagrams. To say in the ontology that diabetes is a metabolic disorder is not to say simply that all the cases of diabetes happen to fall in the set of metabolic disorders in the way that we might say, for example, that all the red-haired boys in the school are taking geometry. Rather, the ontology states that the basic axioms that represent our understanding of the world imply that a case of diabetes that was not a metabolic disorder would be a contradiction.

Individuals, at the level the ontology, represent things in the world – a case of diabetes, a patients, an act of giving a dose of insulin, etc. Statements about classes are really disguised statements about all possible instances of that class – *e.g.* "All patients are persons", "All diabetes involves disturbance of regulation of glucose metabolism", etc. Because they represent generic knowledge about all possible individuals, most ontologies consist mainly of classes. Information about their instances is usually inferred "on the fly" be interpreting databases or messages. The interpretation step is often not made explicit. This leads to confusion between information about the data structures – *e.g.* missing or null data – and information about patients and their care, confusion that can complicate attempts to draw inference about patient, *e.g.* in decision support systems.

Returning to Figure 1, we can determine relationships between meanings only at the level of the our understanding of the world – the level of the ontology. Codes and data structures do not "mean" anything on their own; they are merely valid or invalid. Hence, strictly at the level of data structures, it is impossible to say of two data structures and their associated codes are equivalent or not. However, if the two data structures are *interpreted* (See Fig 1) in terms of the meanings that they encode about the world – *i.e.* at the level of the ontology – then we can ask if they are logically equivalent – if they mean the same thing.

Conversely, we can specify valid sets of codes only at the level of the information model and coding system. When we specify an expression made up of codes, we are specifying classes of symbols, *e.g.* the class of all of the subcodes of the code for diabetes. We can form any arbitrary class of codes, including inclusions and exclusions, even if the corresponding descriptions of patients at the level of the ontology might be nonsensical or even inconsistent, for example the class of all codes for kinds of diabetes but not for diabetes itself, as shown in figure 2b. By contrast at the level of the ontology, the class of all cases of kinds of diabetes is, by definition, just the class of all cases of diabetes. There is simply no meaning to the notion of all kinds of diabetes except diabetes itself.

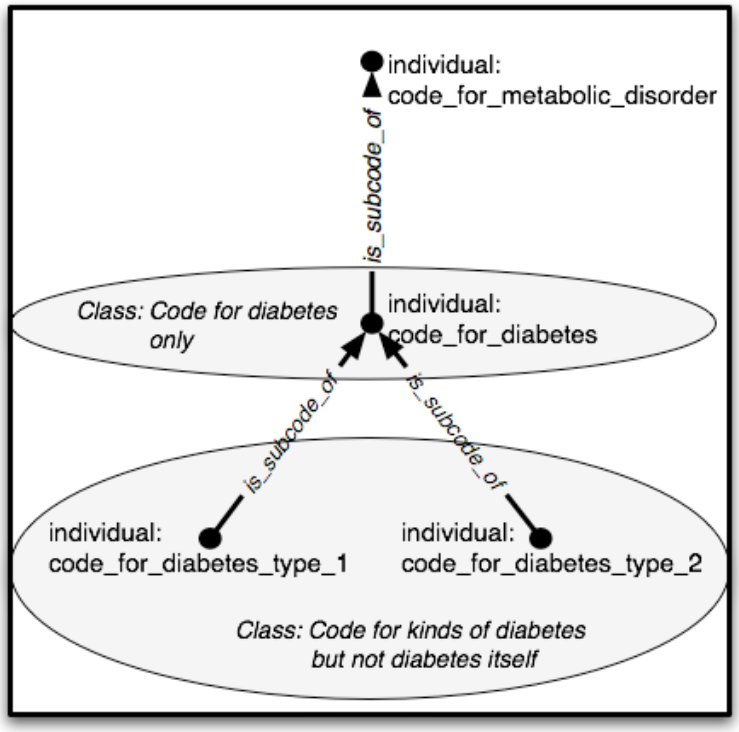

Figure 2b: Sets/Selections that can be formulated at the level of codes in the information model but not at the level of the ontology (upper half Fig 2a).

The relationship between ontologies and coding systems in Figures 1 and 2 is idealised, although to a first approximation, it corresponds to the structure of SNOMED-CT. The "stated form" of SNOMED-CT in its description logic Ontylog corresponds to the ontology proper; the file structures in the distribution correspond to the coding system derived from that ontology. However, even in SNOMED the correspondence is not perfect. In other coding systems, the relationship is much less clean. A coding systems may distort its ontology; there may be no explicit ontology behind a coding system, or idiosyncratic "work arounds" in the coding system may be required to compensate for limitations in the expressiveness of the ontology formalism, as in SNOMED's mechanisms for dealing with negation.

For all these reasons, it is equally fruitless to try to specify conditions for validity at the level of the ontology and to try to specify equivalence of meaning at the level of the information model. The binding of information to coding systems – the Ontology Binding Interface (OBI) – must be at

the level of the information system. Information systems can be linked to the ontologies only indirectly via meta-models or codes. In what follows, we illustrate a simple case using the Web Ontology Language, OWL.

3. Using codes in messages and Electronic Health Records

3.1 Basic requirements and tools

This work has been performed as part of a collaboration with practical users in the UK National Health Service. Its goal is to represent all of the constraints at the level of the information model – both those on the information model proper, on the coding system, and between the two. Our goal is to satisfy their requirements that:

1. There be a clear interface between the model of codes and the information model proper, a "Code Binding Interface" ("CBI");
2. The binding be expressive enough to capture a) enumerated lists of codes; b) all subcodes of a given code (with or without the root); c) all boolean combinations of a) and b).
3. The binding deal with both pre-coordinated and post coordinated expressions in SNOMED-CT, or other compositional coding systems.
4. The mutual constraints between the information and coding models be explicit and testable.
5. The constraints between information and coding models be part of a coherent methodology for expressing the constraints on the information model as a whole.
6. The models and interfaces be expressed in standard languages with well defined semantics and tools.

For the example implementation, we have chosen for the implementation language OWL-DL, the description logic variant of the new W3C standard Web Ontology Language. In practice, we have used some features from the new OWL 1.1 specification[3] that have already been widely implemented by tool builders. We use OWL here primarily as a standard syntax and toolset for description logics, a subset of first order logic. The use of OWL does not imply that the information models are 'ontologies' in any strong sense of that word.

Although for uniformity and clarity, OWL is used here to represent both the ontology and the information model, there is no reason that the ontology, information model and coding system need be represented in the same formalism. The principles would remain the same, although the implementation would require additional mechanisms to translate between formalisms.

3.2 Vocabulary and Notation

Because we take the model of codes and to be at the same level as the information model, a better vocabulary would be to speak of the "information model" as comprising the "model of data structures" – the "information model proper" –and the "model of codes", and then to distinguish both parts of the "information model" from the "model of meaning". However, general usage is to use the phrase "information model" for what we here describe as the "model of data structures". We shall follow that usage except where more precision is needed to avoid ambiguity.

For consistency with OWL's usage, we use the term "class" for what some others would prefer to call "types" or "universals". We refer to "individuals" where some might use the word "instances" and reserve the word "instance" for the relation between a class and an individual belonging to that class. We use the word "illness" to refer to an individual illness – *e.g.* "John Smith's diabetes" and the term "condition" to refer to a class of illnesses – *e.g.* "Diabetes". We use the term "property" to refer to relations between individuals. As a typographical convention, labels for classes begin with upper case; individuals and properties with lower case, and OWL keywords are in all upper case.

[3] http://www-db.research.bell-labs.com/user/pfps/owl/overview.html

All work reported was performed using the Protégé-OWL tools version 4[4]. Throughout we adopt the simplified Manchester syntax for OWL used in the Protégé-OWL tools (Horridge *et al.*, 2006), a summary of which is presented in Figure 3.

OWL abstract syntax	**Simplified Syntax**	**German DL Syntax**
someValuesFrom(C)	**SOME C** **ANY C** (in negative context)	∃.C
allValuesFrom(C)	**ONLY C**	∀.C
minCardinality(n C)	**MIN n C**	≤ n. C
maxCardinality(n C)	**MAX n C**	≥.n C
cardinality(n C)	**EXACTLY n C**	derived
value(c)	**VALUE c** or **HAS c**	c
intersectionOf(C D)	**C AND D** or **C & D** or **C THAT D**, or **C, D**	C ⊓ D
unionOf(C D)	**C OR D** or **C | D**	C ⊔ D
oneOf(…)	**{…}**	{…}
equivalentClasses	←→	C ≙ D
subclassOf	→	C ⊑ D
Type	∈	∈
allDifferent	**DIFFERENT**	
allDisjoint	**DISJOINT**	
owl:Thing	**THING**	⊤
owl:Nothing	**NOTHING**	⊥

Figure 3: Manchester simplified syntax for OWL 1.1

4. Binding the Models of Meaning, Codes, and Information Proper

As a simplified example, we wish to specify the binding between a fragment of the Electronic Health Record model conforming to the constraints expressed informally in Figure 4a. We show the relation of the models diagrammatically in Figure 4b.

[4] http://protégé.stanford.edu; http://www.co-ode.org

Field	Constraint
Topic	Exact code for diabetes mellitus, mandatory
Diagnosis	The code for diabetes or any kind of diabetes, optional
Brittleness	One of the subcodes of the code for "Diabetic Brittleness"

Figure 4a: Fields and constraints for example simplified information structure for Diabetes

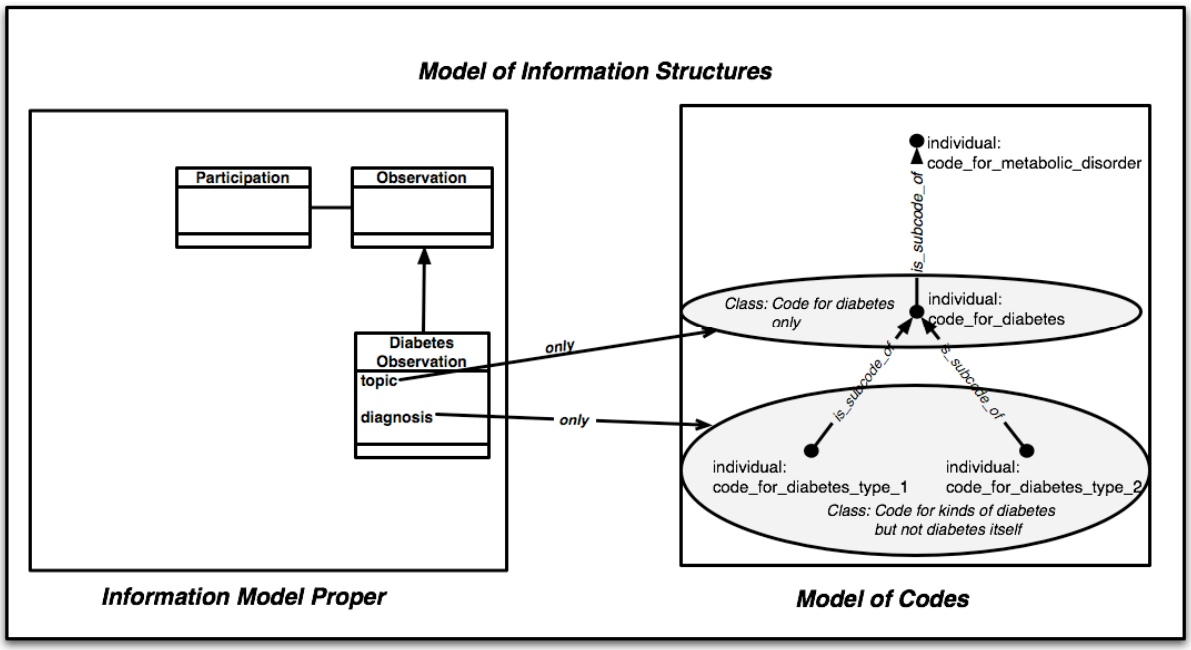

Figure 4b: Binding of attributes in the information proper to the model of codes in the Model of Data Structures for some of constraints in Fig 4a.

The two rectangles in Figure 4b represent the information model proper on the left and the models of codes or "coding system" on the right from Figure 2b. The two arrows between the attributes in the information proper and classes of codes in the model of codes represent the bindings, from "topic" to "Code for diabetes only" and from "diagnosis" to "Code for kinds of diabetes but not diabetes itself." (Formal representations of these links are given in Section 5 Figures 7-8.) Note that the model of meaning itself, the ontology, shown in the top half of Figure 2a, does not take part in the binding. The oval superimposed on the hierarchy in this model represents a class of codes, in this case the class of "the code for diabetes and all its subcodes". §

```
CLASS Diabetes →
   Metabolic_disorder,
   has_quality EXACTLY 1 Brittleness.

CLASS Diabetes_type_1 →
  Diabetes,
  is_caused_by SOME (Damage AND has_locus SOME Pancreatic_islet_cells).

CLASS Diabetes_type_2 →
  Diabetes,
  is_caused_by SOME
     ((Resistance AND has_locus SOME Insulin_metabolism)
      OR
     (Reduced_effectiveness AND has_locus SOME Insulin)).

CLASS Diabetic_brittleness ←→
    Brittleness,
    is_quality_of SOME Diabetes.

CLASS Diabetic_brittleness →
   has_state EXACTLY 1 Brittleness_state.

CLASS Diabetic_brittleness_state ←→
    Brittleness_state,
    is_state_of SOME Diabetic_brittleness.

CLASS Diabetic_brittleness_state →
    Brittle OR Well_controlled.
```

Figure 5: Fragment of simplified model of meaning ('ontology') for Diabetes.

5. Representing the Binding in OWL

5.1 The Model of Meaning – the "Ontology"

Figure 5 shows a fragment of a simplified ontology of conditions. The first line says that Diabetes is a kind of Metabolic_disorder and that it has a quality of Brittleness. The "EXACTLY"[5] keyword indicates that each illness of class Diabetes has one, and only one, Brittleness quality. The definition is not closed, so there is nothing in this representation to say that Diabetes cannot have other qualities.

The next two clauses give simplified definitions of type 1 and type 2 diabetes.

The following clause defines Diabetic_brittleness using the inverse of the quality relationship to say that any Brittleness that occurs in the context of being a quality of Diabetes is a Diabetic_brittleness. The next clause states that each Diabetic_brittleness quality has one, and only one Brittleness_state. The final two clauses define Diabetic_brittleness_state as any Brittleness_state in the context of Diabetic_brittleness, and then state that it includes only the two values: Brittle and Well_controlled.

[5] "EXACTLY" is an OWL 1.1 construct

```
CLASS Code → Coding_entity.

INDIVIDUAL code_for_diabetes ∈
   Code,
   is_subcode_of VALUE code_for_metabolic_disorder.

INDIVIDUAL code_for_diabetes_type_1 ∈
    Code,
     is_subcode_of VALUE code_for_diabetes.

INDIVIDUAL code_for_diabetes_type_2 ∈
     Code,
     is_subcode_of VALUE code_for_diabetes.

INDIVIDUAL code_for_diabetic_brittleness_value ∈
     Code.

INDIVIDUAL code_for_diabetic_brittle ∈
      Code,
      is_subcode_of VALUE code_for_diabetic_brittleness_value.

INDIVIDUAL code_for_diabetic_well_controlled ∈
      Code,
      is_subcode_of VALUE code_for_diabetic_brittleness.
```

Figure 6a: Individuals in the Model of Codes in the Information Model corresponding to Figure 5.

```
CLASS Code_for_diabetes_and_subcodes ←→
  {code_for_diabetes} OR
  is_subcode_of VALUE code_for_diabetes.

CLASS Subcode_of_code_for_diabetic_brittleness ←→
  is_subcode_of VALUE code_for_diabetic_brittleness_value.
```

Figure 6b: Classes of codes defined from code individuals. The first class corresponds to the shaded oval on the bottom right of Figure 4.

5.2 The model of codes – the coding system

Of the information in the ontology, only some is likely to be relevant to the coding system. For purposes of illustration, we shall concentrate only on qualities and omit causation. The information as to which properties are of interest is 'meta knowledge' that must be held in a "profile" specifying the required transformation.

Using an appropriate profile, the definitions of individual codes in Figure 6a might be generated from the ontology in Figure 5. We can then define classes of codes for use in the bindings as shown in Figure 6b. Since this model correctly mirrors a fragment of the ontology, the hierarchy the code classes will mirror the condition classes in the ontology. However, note that the additional constraints in the definitions are different in the ontology and coding system specified in the profile. For example, there is no axiom in the coding system that all diabetic codes must have a brittleness qualifier, although there is an axiom in the ontology that all (cases of) Diabetes have a quality Brittleness. This reflects a decision by the modellers, that in the data structures for diabetes, the attribute for brittleness will be optional. (As explained in section 2, the notion of "optional" makes no sense in the ontology.)

```
PROPERTY has_code FUNCTIONAL.

CLASS Topic → Coded_Attribute.
CLASS Diagnosis → Coded_Attribute.
CLASS Brittleness → Coded_Attribute.

CLASS Condition_data_structure →
  has_attr EXACTLY 1 Topic_attr,
  has_attr EXACTLY 1 Diagnosis_attr.

CLASS Diabetes_data_structure →
  Condition_data_structure,
  has_attr EXACTLY 1 Brittleness_attr.
```

Figure 7a: Basic data structures in the Information Model to OWL

```
CLASS Placeholder_cls_diabetes_only_code   → Code.

CLASS Placeholder_cls_diabetes_or_subcode → Code.

CLASS Placeholder_cls_diabetic_brittleness_subcode →Code.
```

Figure 7b: Placeholder code classes for use in Code Binding Interface (CBI)

```
CLASS Diabetes_data_structure →
  has_attr SOME (Topic_attr AND has_code SOME
    Placeholder_cls_diabetes_only_code),
  has_attr SOME (Diagnosis_attr AND has_code SOME
    Placeholder_cls_diabetes_or_subcode),
  has_attr SOME (Brittleness_attr AND has_code ONLY
    Placeholder_cls_diabetic_brittleness_subcode).
```

Figure 7c: Use of placeholder code classes and indication of whether codes are mandatory (*SOME*) or optional (*ONLY*).

5.3 The information model proper

A basic OWL model capturing the data structures or "information model proper" implied in Figure 4a is shown in Figures 7abc. We assume that we are modelling a class of diabetic data structures that have attributes for each item in Figure 4a: "topic", "diagnosis", and "brittleness". We map each attribute by a class linked to the data structure by the property has_attr. We define a special subclass of attributes that take codes as their values, Coded_Attribute. Each Coded_Attribute is linked to a maximum of one Code as the value by the has_code property.

We assume that there is a generic class Condition_data_structure all of whose instances have Topic and Diagnosis attributes, but that the Brittleness attribute is specific to the class Diabetes_data_structure. Because the class Diabetes_data_structure is a subclass of Condition_data_structure, it "inherits" all of the attributes of its superclass.

Although a representation in which attributes are mapped to properties (as is done in the mapping specified by OMG (IBM & Sandpiper Software Inc., 2005)) might seem simpler, mapping each attribute to its own class makes it easier to specify cardinality and closure at the level of detail required for HL7 and OpenEHR models.

5.4 Constraining the codes to placeholders

Given the basic information model defined in Figure 7a, we want to indicate that there are constraints on the codes to be used with each attribute. However, we do not wish to specify the coding system or the coding system specific constraints in the information model itself. Therefore, at this stage we state only that each attribute is constrained to a placeholder class of codes. These placeholder classes of codes are defined in Figure 7b.

The placeholder classes of codes can then be used to express generic constraints on the information model as shown in Figure 7c. In this example, we have stated the Topic and Diagnosis codes are mandatory, as indicated by the keyword "SOME". However, by using the keyword ONLY for Brittleness_code, we have said that it is optional (because stating that a property can have *only* codes from a particular class of codes does not imply that it has *any* such codes).

```
CLASS Placeholder_cls_diabetes_only_code ←→
    {code_for_diabetes}.

CLASS Placeholder_cls_diabetes_or_subcode ←→
  {code_for_diabetes} OR
    is_subcode_of VALUE code_for_diabetes.

CLASS Placeholder_cls_diabetic_brittleness_subcode ←→
     is_subcode_of VALUE code_for_diabetic_brittleness.
```

Figure 8: Code Binding Interface for Code System in Fig 6 and Information Model in Fig 7.

5.5 The Code Binding Interface

The model of the coding system in Figures 6ab and the information model in Figures 7abc might reside in separate modules. It now remains to define the Code Binding Interface between the two modules, which might reside in still a third module.

The Code Binding Interface (CBI) consists of logical equivalences between the placeholder classes defined in Figure 7b and formal definitions of classes of codes in terms of the individuals in the model of codes in Figures 6ab. A CBI consistent with the constraints in Figure 4a is shown in Figure 8. The first line indicates that the placeholder class consists of just those codes enumerated between the curly brackets, in this case just the code for diabetes. The second line indicates that the given placeholder can be either the code for diabetes or any of its subcodes. (The property is_subcode_of is transitive.) The third line indicates that the code for brittleness can be any of the subcodes of the code for diabetic brittleness but not the parent code itself. They can be combined using the boolean operators AND, OR, and NOT. These were the three specific cases to be covered in Requirement 2.

```
CLASS Qualifier_name_code → Code.

CLASS Qualifier_value_code → Code.

CLASS Qualifier →
      Coding_entity,
      has_qualifier_name_code EXACTLY 1 Qualifier_name_code,
      has_qualifier_value_code EXACTLY 1 Qualifier_value_code

INDIVIDUAL code_for_diabetic_brittleness_qualifier ∈
    Qualifier_name_code.

INDIVIDUAL code_for_diabetic_brittleness_value ∈
    Qualifier_value_code.

CLASS Code_for_diabetes_and_subcodes →
   has_qualifier MAX 1 (Qualifier AND
           has_qualifier_name_code VALUE code_for_diabetic_brittleness_qualifier).

CLASS (Qualifier AND
           has_qualifier_name_code VALUE  code_for_diabetic_brittleness_qualifier)
                                                                          →
    has_qualifier_value_code  EXACTLY 1 Subcode_of_code_for_diabetic_brittleness_value.
```

Figure 9a: Extension of Model of Codes to qualifiers.

```
CLASS Placeholder_diabetes_or_subcode_class ←→
  ({code_for_diabetes}
  OR
   is_subcode_of VALUE code_for_diabetes) AND
       NOT (has_qualifier ANY
                  (Qualifier AND
                    has_qualifier_name_code VALUE code_for_diabetic_brittleness_qualifier)).
```

Figure 9b: Extension of CBI in Figure 7 to exclude codes qualified by brittleness.

```
CLASS Placeholder_diabetes_or_subcode_class ←→
  ({code_for_diabetes}
  OR
   is_subcode_of VALUE code_for_diabetes).

CLASS Placeholder_cls_diabetic_brittleness_subcode ←→ NOTHING
```

Figure 9c: Alternative extension of CBI in Figure 7 to allow codes for brittleness but exclude attributes for brittleness – or at least any coded value for an attribute for brittleness.

5.6 Extension to compositional coding systems

The previous example was limited to simple coding systems without 'qualifiers'. However, the same principles can be extended to a coding system with qualifiers using suitably more complex constraints. In this case, since "brittleness" is to be explicitly catered for in the information model, we want to avoid any possibility of a contradiction between the value in the information structure and the qualifier in the terminology. Whether "brittleness" is represented in the coding system or in the information model by an attribute or association is arbitrary. Our requirement here is to be able to provide a rigorous specification, whichever choice is made.

To do so we first have to extend the representation in Figures 6ab to include qualifiers as shown in Figure 9a. We need a new class of data structures, Qualifier, which is a subclass of the class Coding_entity and a sibling of the class Code. Qualifiers are linked to the codes they qualify by the property has_qualifier. Each instance of Qualifier has two properties – has_qualifier_name_code and has_qualifier_value_code filled by individuals from the classes Qualifier_name_code and Qualifier_value_code respectively, both of which are subclasses of Code. In the penultimate line of Figure 9a, we assert that in general, in the coding system, Codes for diabetes may have a qualifier with the name Brittlenes_qualifier_name_code. Note that the qualifier is MAX 1 rather than SOME, indicating that the qualifier is not mandatory.

We first illustrate how to use these mechanisms to say that, in this binding, the brittleness will be represented in the information model rather than the coding system – *i.e.* that in this binding, no code in the class Placeholder_diabetes_or_subcode_class may have a qualifier with the qualifier name code_for_diabetic_brittleness.

Given the infrastructure in the coding model for describing qualifiers in Figure 9a, we can then extend the original code binding interface in Figure 8 as shown in Figure 9b. This asserts that, in this particular code binding interface, the code for diabetes – *i.e.* any member of the class, Placeholder_diabetes_or_subcode_class – cannot have a qualifier for diabetic brittleness.

A different group might develop a different information model that does not include brittleness as a separate attribute. It might, therefore, want to include brittleness with the diagnosis code. To do so, they need only change the Code Binding Interface to that shown in Figure 9c. In this case, we allow the brittleness qualifier of diabetes but make the placeholder class for codes for the brittleness attribute NOTHING, the class that can have no members (owl:Nothing, equivalent to the empty set of "bottom"). Alternatively, one might exclude the attribute "brittleness" from the information model, but this would involve changing the model itself rather than just the code binding interface.

Note that the same methodology can be used whether the code in question is predefined and named ("pre-coordinated") or left as an expression ("post-coordinated" also known as "code phrases" in HL7, the major standard for medical messages).

```
DISJOINT Diabetes_type_1, Diabetes_type_2.

DISJOINT Brittle, Well_controlled.
```

Figure 10a: Differentiating axioms for the model of meaning

```
DIFFERENT
 code_for_diabetes   code_for_diabetes_type_1,  code_for_diabetes_type_2,
 code_for_diabetic_brittleness, code_for_diabetic_brittle, code_for_diabetic_well_controlled).
```

Figure 10b: Differentiating axioms for the model of codes – the coding system

```
DISJOINT Data_structure, Attribute.

DISJOINT Topic, Diagnosis, Brittleness.
```

Figure 10c: Differentiating axioms for the information model.

5.7 Absence of the Unique Name Assumption and differentiating axioms

The above representations in OWL require a further addition. OWL does not make the "Unique name assumption". In most formalisms, if two entities have different names, then they are different. OWL, which was specifically designed to facilitate merging of Ontologies and allow equivalences of the type used in the "Code Binding Interface. To achieve this, OWL must allow for

the possibility that any two individuals or classes might overlap or be logically equivalent unless they are specifically declared to be distinct.

Therefore, to represent the intentions fully, we need a set of "differentiating axioms" examples of which are shown in Figures 10abc. If these axioms are omitted, the validation in the next section will be incomplete because the reasoner will never infer that a code is incorrect because the reasoner cannot infer that it is different from the correct code, even though it has a different name.

5.8 Validating information models

OWL-DL was chosen because it allows efficient reasoners. In principle, the task of using OWL-DL to represent and validate a set of information models and bindings to a coding system simply requires that the reasoner be used to determine if the combined models are consistent and the inferences drawn are as intended. Taking into account the previous discussion, the complete procedure consists of the following steps:

1. Transform the relevant parts of the model of meaning, *i.e.* the ontology, into a meta-level model of codes following the example in Figures 5 and 6ab.
2. Map the information model to an OWL model including the constraints on the terminology to be used as placeholders following the example in Figures 7abc.
3. Represent the bindings between the information model and the coding system model as a set of logical equivalences between the placeholders in the information model and class expressions in the coding system model to form the Code Binding Interface (CBI) module, following the example in Figure 8.
4. Import the three modules into a single OWL model.
5. Use the reasoner to classify the combined structure. Inconsistencies, inferred subclass relations, and inferred equivalencies will be flagged by the reasoner.
6. Examine the inferences and correct the errors.

Note that inferred subclass relations and equivalencies as well as inconsistencies may indicate errors. If an inferred subclass relation is not as intended, then either the superclass is under-constrained – *i.e.* too general – or the subclass is over-constrained – *i.e.* too specialised. If two classes that are intended to be different are inferred to be logically equivalent, then the distinguishing features have been omitted or an axiom with unexpected consequences included. (There are a host of subtle errors that can occur in OWL models that are beyond the scope of this paper – see (A. Rector *et al.*, 2004a)).

```
CLASS Diabetes_data_structure_complete →
  Diabetes_data_structure,
  has_attr ONLY (Topic OR Diagnosis OR Brittleness).
```

Figure 11a: "Completed" subclass of the Diabetes data structure class with closure axiom

```
INDIVIDUAL code_for_metabolic_disorcer ∈
   has_subcode ONLY {…code_for_diabetes…}.

INDIVIDUAL code_for_diabetes ∈
   has_subcode ONLY {code_for_diabetes_type_1
                     code_for_diabetes_type_2}.

INDIVIDUAL code_for_diabetes_type_1 ∈
   NOT (has_subcode ANY THING).

INDIVIDUAL code_for_diabetes_type_2 ∈
   NOT (has_subcode ANY THING).
```

Figure 11b: Closure axioms for code for diabetes

```
INDIVIDUAL diabetic_data_structure_123 ∈
  has_attr SOME  (Topic & has_code VALUE code_for_diabetes),
  has_attr SOME (Diagnosis & has_code  VALUE  code_for_diabetes_type_1),
  has_attr SOME (Brittleness  & has_code code_for_diabetic_brittle),
  has_attr ONLY (Topic OR Diagnosis OR Brittleness).
```

Figure 11c: The OWL mapping of a Diabetic data structure including closure axiom.

5.9 Validating individual data structures – the open and closed world assumptions

Before individual data structures can be validated, we must take into account a further feature of OWL's semantics. Databases, logic programs, and most related systems are based on a "closed world assumption" with "negation as failure" – *i.e.* anything that cannot be found in the database or proved true is treated as false. OWL is based on the "open world assumption" – *i.e.* things not proved true are treated as unknown; only things that can be proved false are treated as false. The open world assumption means that one can always add to an OWL model unless there is an explicit "closure axiom" to the contrary. Without the closure axiom, an OWL model or data structure means only "at least what is here". By contrast, most message and Electronic Health Record formalisms assume that a given data structure contains "what is here and only what is here". Without closure axioms, OWL will accept a data structure with missing items because, since the representation is open, the missing item could always be added.

Closure axioms are required in three places: a) in the information model to say that a particular class is complete, b) in the model of codes, to say that each code has only the subcodes explicitly asserted, and c) in each individual data structure to be validated, to say that it contains only what is explicitly present.

Step a: Before validating the model in Figures 7abc we need to create a new subclass of "complete diabetes data structures" with the added closure axioms. The new subclass definition is shown in Figure 11a. The second clause is the "closure axiom" that says that only these three attributes may occur.

Step b: The model of codes must similarly be closed, downwards by adding closure axioms to state that each node *only* has the subcodes listed and the terminal codes have no subcodes as indicated by the restriction "NOT has_subcode ANY THING".

Step c: An OWL mapping of a data structure that conforms to the model in Figures 6ab is shown in Figure 11c. The final line is the closure axiom. (The use of SOME and ONLY rather than VALUE avoids the need to define individuals for each individual data structure's attributes: Topic_attr, Diagnosis_attr and Brittleness_attr.)

Therefore, the steps to validate that a data structure conforms to the information model are:

1. Map the data structure to an OWL individual following the example in Figure 11c.
2. Add closure axiom as shown in Figure 11c.
3. Use the reasoner to check if the data structure is a valid instance of the intended class in the information model.

Recently, there has been much work in the OWL community on adding standard constructs to OWL to verify "constraints" expressed in information models (Motik *et al.*, 2007). Practical systems with these features are expected in early 2008 and are likely to form a part of the next owl specification by W3C. Such features will greatly simplify the use of OWL in such applications.

5.9.1 Limitations of OWL

OWL-DL is based on a subset of first order logic deliberately limited so that inference is computationally feasible. There are two main limitations relevant to the work reported here:

- *Limited support for data types.* Both HL7 and Archetypes have very elaborate structures of datatypes that go beyond the usual XML datatypes supported by OWL. This can be overcome by encapsulating datatypes in "holders". What OWL provides is a check on the constraints on which data types should be used where. The OWL specification provides for user defined data types. However, experience suggests that the more complex HL7 data types are best modelled explicitly. The most complex data types might require syntax checkers may be required to check the datatype formats themselves.
- *Lack of variables.* To preserve computational feasibility, OWL lacks auxiliary variables and expressions such as "same-as". For example, one can say that the left hand must be part of the left arm, but not that hands must be part of arms on the same side. Usually, it is possible to work around this limitation by having separate axioms for each case, *e.g.* for left-sided and right-sided rather than a single axiom for "same side". UML, and most other object-oriented formalisms, share this limitation. It has not proved a serious limitation in practice in the experience reported below or in related applications.

5.10 Experience: Representing HL7 message fragments developed by the NHS Connecting for Health

The methods in this paper are a refinement and generalisation of methods that were developed to represent the constraints in a set of message models developed by the UK NHS Connecting for Health Programme and their binding to SNOMED CT. The set of messages related to administration of medication were represented, a total of between twenty and thirty message formats (depending on how variants are counted). The methods were successful in representing all of the constraints identified, both in the HL7 models themselves and in the accompanying documentation, including the complex constraints on compositional forms required to maintain consistency between the SNOMED Context Model and the HL7 mood and status codes.

The representation, however, was tedious. Existing OWL tools are adapted to representing ontologies and models of meaning rather than data structures. Wider use of the methods presented here would benefit from the development of alternative tools, or at least alternative front-ends. In this respect, OWL is best viewed as an assembly language. A high level language adapted to the task of representing information systems and their binding to coding systems is required along with 'compilers' to transform it to OWL in a standard way.

6. Discussion

In previous papers (A. Rector *et al.*, 2004b; A. L. Rector, 2001; A. L. Rector *et al.*, 2001) we have identified the interface between "models of meaning" – the ontology – and "models of use" as critical to clinical systems. The "information model" is one form of "model of use". This paper clarifies the relation between the "model of meaning" and two parts of the "information model" – the

"model of codes", and the "model of data structures" (or "information model proper"). It contends that the "information models" – both the "model of data structures" and the "model of codes" – are formulated at a meta level with respect to the "model of meaning", *i.e.* the "ontology".

This paper illustrates a methodology for formulating a "Code Binding Interface" (CBI) to specify and constrain how codes are to be used in data structures. If the model of codes accurately reflects the model of meaning , then this will effectively bind the information model to the model of meaning. However, the binding is indirect. The task and methodology described in this paper is essentially "syntactic" – it binds symbols to data structures. It is concerned with whether the data structures are valid and can be processed reliably rather than with whether the information conveyed is accurate or correct. The structure of the information model is motivated by adequacy to convey meanings, but the constraints in the model are on the data structures to ensure validity rather than on the meaning to ensure accuracy. We suggest that the controversies around coding systems and standards such as HL7 arise, in part, from lack of clarity about this distinction between validity and accuracy. In particular, many of the comments of Smith and Ceusters (Smith & Ceusters, 2006) critique the information model as if it were intended to be an ontology rather than whether it can be effectively bound to a terminology. Likewise, "referents" in proposed mechanisms for "referent tracking" (Ceusters & Smith, 2006), should clearly refer to the level of the ontology – the model of our understanding of the world – rather than the information model.

The methodology has been used in practice and proved effective in supporting a range of independently formulated constraints.

This theoretical justification and practical experience is further supported by the observation that the requirements in the introduction cannot be met by a first order model of meaning directly linked to the information model. Requirement 2 includes being able to restrict the value of an attribute to a specific code at any level of abstraction – *e.g.* to "the specific code for diabetes" – or to any of the subcodes of a parent code but not the parent code itself – *e.g.* "to any subcode of brittleness". However, the semantics of the model of meaning are defined in terms of classes of illnesses. The class "all diabetic illnesses that are neither type 1 nor type 2" would be all those diabetic illnesses of some alternative type – a class that is quite probably empty. It would *not* be the parent class, Diabetes, as required. By contrast, if dealing with classes of codes at the meta-level, the required expressions, as shown in Figure 8, are straightforward. Implicitly, this is what most users of terminologies such as SNOMED actually do – they query the coding system in a "distribution form" which does not give access to the underlying semantics. However, without explicit recognition of the separation of the models of meaning and meta-level models of coding systems, these mechanisms remain *ad hoc* and cannot be specified formally.

This paper deals with only the first three steps in using patient information – formulating meanings, encoding the information in data structures and symbols to be stored or transmitted, and then re-interpreting those symbols and data structures as patient information. The fourth step – using the patient information for clinical decisions – requires a further model – a model of clinical action – to be discussed in a further paper.

The methodology given here meets the requirements given in the introduction for binding ontologies, coding and information models. There is great controversy over the flaws in both SNOMED and HL7. The indirection in this methodology can help provide rigorous specifications that allow systems to interoperate using valid message despite flaws in the model of data structures, model of codes, or both. However, even if the models were ideal, the ontology sound, the coding system a faithful meta model of the ontology, and the information model founded on a sound model of the information to be conveyed, a "Code Binding Interface" would still need to be specified to specify what constituted valid bindings of codes to the data structures. Any given message or record fragment will provide places for only a limited view on all possible meanings and hence all possible combinations of codes. Even in a near ideal world, if the information model and ontology are developed independently, there will still be overlaps and consequent need for mutual constraints between them.

Whether the methodology presented here is the best means to do so remains open to investigation. OWL has the technical advantage of being highly expressive, of supporting inverse properties which can be used to represent context, and of having available sound and complete reasoners. Its status as a standard brings the organisational advantage of a broad community developing tools and techniques. Current work on constraints and query languages promise to make the methods presented easier to implement. A principled layered version of OWL similar to that in this paper has also been suggested by others (Pan *et al.*, 2005). Current proposals on queries (Motik et al., 2007; Sirin & Parsia, 2007), constraints (Motik et al., 2007), and meta-modelling (Motik & Cuenco-Grau, 2007) for OWL promise to provide more choices in how the basic principles presented here might be implemented, and extensions to the OWL standard now under consideration are likely to make such approaches easier. We hope that the issues are presented here in sufficient detail to allow alternatives to be formulated and compared.

Acknowledgements

This work supported in part by the UK Department of Health "Connecting for Health" programme, the UK MRC CLEF project (G0100852), the JISC and UK EPSRC projects CO-ODE and HyOntUse (GR/S44686/1) and the EU Funded Semantic Mining Network of Excellence. The HL7 Terminfo working group stimulated and contributed to many of the ideas presented here.

References

Beale, T. (2002). Archetypes: Constraint-based domain models for future-proof information systems. Paper presented at the OOPSLA-2002 Workshop on behavioural semantics, available from http://www.oceaninformatics.biz/publications/archetypes_new.pdf.

Ceusters, W., & Smith, B. (2006). Strategies for referent tracking in electronic health records. Journal of Biomedical Informatics, 39(3), 362-378.

Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., & Wang, H. (2006). The manchester owl syntax. Paper presented at the OWL: Experiences and Directions (OWLED 06), Athens, Georgia, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-216/submission_219.pdf.

IBM, & Sandpiper Software Inc. (2005). Ontology definition metamodel: Third revised submission to omg: OMG.

Motik, B., & Cuenco-Grau, B. (2007). Making metalogical information in ontologies logical using metaviews. Paper presented at the International Semantic Web Conference (ISWC 2007), Busan, Korea, (in press).

Motik, B., Horrocks, I., & Sattler, U. (2007). Adding integrity constraints to owl. Paper presented at the Third OWL Experiences and Directions Workshop (OWLEd-2007), Salzburg, Austria.

Pan, J. Z., Horrocks, I., & Schreiber, G. (2005). Owl-fa: A metamodeling extension of owl dl. Paper presented at the Proc International workshop on OWL: Experiences and Directions (OWL-ED2005).

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., et al. (2004a, October, 2004). Owl pizzas: Common errors & common patterns from practical experience of teaching owl-dl. Paper presented at the European Knowledge Acquisition Workshop (EKAW-2004), Northampton, England, 63-81.

Rector, A., Taweel, A., & Rogers, J. (2004b). Models and inference methods for clinical systems: A principled approach. Paper presented at the Medinfo, San Francisco, 79-83.

Rector, A. L. (2001). The interface between information, terminology, and inference models. Paper presented at the Tenth World Conference on Medical and Health Informatics: Medinfo-2001, London, England, 246-250.

Rector, A. L., Johnson, P. D., Tu, S., Wroe, C., & Rogers, J. (2001). Interface of inference models with concept and medical record models. Paper presented at the Artificial Intelligence in Medicine Europe (AIME), Cascais, Portugal, 314-323.

Schadow, G., Russler, D. C., Mead, C. N., & McDonald, C. J. (2000). Integrating medical information and knowledge in the hl7 rim. Paper presented at the AMIA Fall Symposium, San Antonio, TX, 764-768.

Sirin, E., & Parsia, B. (2007). Sparql-dl: Sparql query language for owl-dl. Paper presented at the OWL Experiences and Directions (OWLEd 2007), Innsbruck, Austria.

Smith, B., & Ceusters, W. (2006). Hl7 rim: An incoherent standard. Paper presented at the Medical Informatics Europe (MIE 2006), Maastricht, Nl, 133-138.