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

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.*

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 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 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. The function of the information models is to make it possible to specify and test the validity of data structures so that they can be exchanged and re-used in different information systems. The function of the model of meaning is accuracy in representing our understanding of the world so that we can reason about the world in general or individual patients and their diseases in particular. Validity neither requires or guarantees accuracy, nor *vice versa*.

We contend that codes are also data structures and the model of codes is also at the level of data structures. Ideally the "model of codes" or "coding system" should be a meta model of the model of meaning. Hence, in the ideal case, 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 correspond to representations of classes of illnesses or "conditions".

Pragmatically, it is useful to decouple the coding system from the model of meaning so that reasoning about the model of meaning and model of coding system is always separated.

Using codes in messages and EHRs

Our goal is to assist software developers in specifying information systems and the use of codes from coding systems within them. We seek to have specifications that are sufficiently precise that separately implemented systems will work together. To achieve this we need to be able to validate that the models themselves are self-consistent and that individual messages conform to the models.

Typically, we want to start with a generic information model such as the HL7 RIM[1] or the OpenEHR reference model[2]. We then want to define progressively more specialised models in which each more specialised model is consistent with the next more generic model and ultimately the reference model. We want to use the models with separately developed coding systems – *e.g.* SNOMED, ICD, CPT, MEDRA, etc. Since we often want to use the same information model with more than one coding system, we want the "binding" between the information and coding system to be separate from both, analogous to an "Application Programming Interface" or "API" between software modules. We call this a "Code Binding Interface"

This problem is often expressed as defining "value sets" or "code sets" or just "subsets". 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 [1] used by the CEN standard EN13606 and OpenEHR specifies an "ontology section" similar in principle to what we here call a Code Binding Interface, but provides as yet only

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

OWL abstract syntax	Simplified Syntax	German DL Syntax
someValuesFrom(C)	**SOME C**	∃.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 **IS c**	c
intersectionOf(C D)	**C AND D** or **C & 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**	

Figure 1: Manchester simplified syntax for OWL

limited mechanisms for expressing semantic constraints.

Basic requirements and tools

This work has been performed as part of a collaboration with practical users in the UK National Health Service. Our goal is to satisfy their requirements that:

1. There be a clear interface between the model of meaning and the information model, 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. That it deal with expressions in SNOMED-CT, whether pre- or post-coordinated.
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 standard language we have chosen 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] which 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.

Vocabulary and Notation

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.

All work reported was performed using the Protégé-OWL tools[4]. Throughout we adopt the simplified Manchester syntax for OWL, a summary of which is presented in Figure 1.[5]

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

Figure 2: Some of the fields and constraints for a example simplified information structure for Diabetes

Binding the models of Meaning, Codes, and Information: Principles

As a simplified example, we wish to specify the binding between a fragment EHR model conforming to the constraints expressed informally in Figure 2.

We show the relation of the models diagrammatically in Figure 3. The upper (yellow) square represents the model of meaning or the ontology. Dots represent individual illnesses such as "john_smiths_diabetes".

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

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

[5] Extensive experience in tutorials and presentations indicate that this notation is more easily understood by those less familiar with OWL as well as being more compact than either of the official OWL syntaxes. Note that the syntax includes OWL 1.1 constructs for qualified cardinality (p MIN|MAX|EXACTLY n C) and allDisjoint.

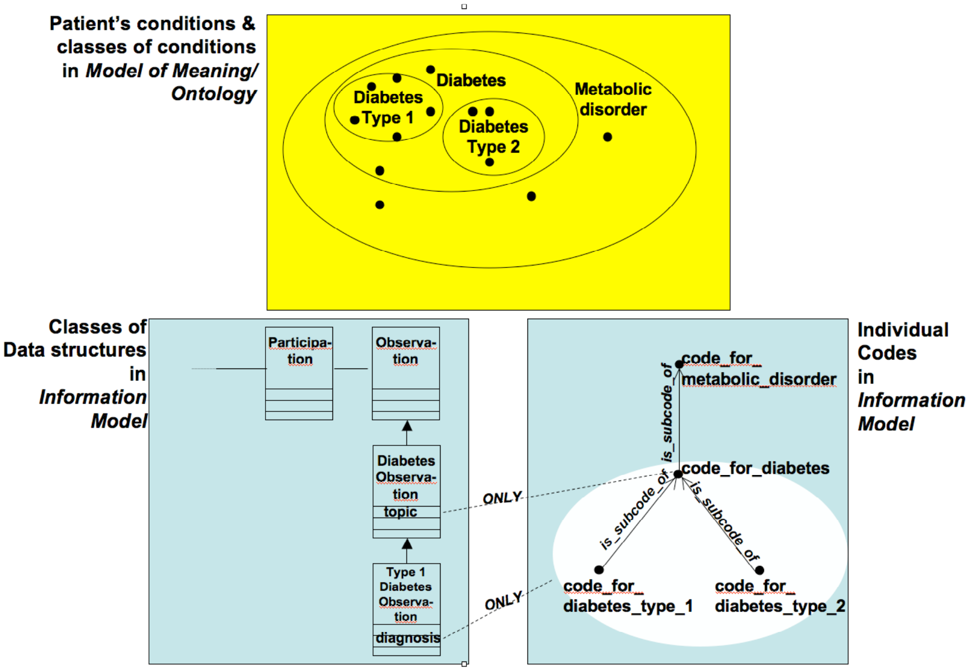

Figure 3: Relation of Model of Meaning to classes of data structures and model of individual codes in the Information Model

Ovals represent classes of illnesses or "conditions" such as "Diabetes_type_1".

The lower two (blue) squares represent the information model on the left and the models of codes or "coding system" derived from the model of meaning on the right. The class hierarchy on the left represents classes of data structures expressed in UML diagrams. The hierarchy on the lower right represents hierarchies of codes linked by the is_subcode_of property, which is transitive. 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 &
     has_locus SOME Insulin_metabolism) OR
    (Reduced_effectiveness &
     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_value_of SOME Diabetic_brittleness.
CLASS Diabetic_brittleness_state →
   Brittle OR Well_controlled.
```

Figure 4: Fragment of simplified condition model of meaning ('ontology') for Diabetes.

In this example, the is_subcode_of property precisely mirrors the inferred subclass relation in the model of meaning and was derived from it by a systematic transformation. However, from the point of view of formal reasoning, each model is treated separately. The inference that, in the model of codes, 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.

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

```
INDIVIDUAL code_for_diabetes ∈
   Code_entity,
   is_subcode_of VALUE code_for_metabolic_disorder.
INDIVIDUAL code_for_diabetes_type_1 ∈
    Code_entity,
    has_code VALUE code_for_diabetes.
INDIVIDUAL code_for_diabetes_type_2 ∈
     Code_entry,
     is_subcode_of VALUE code_for_diabetes.
INDIVIDUAL code_for_diabetic_brittleness ∈
     Code_entry,
     is_subcode_of VALUE code_for_qualifier.
INDIVIDUAL code_for_diabetic_brittle ∈
      Code_entry,
      is_subcode_of VALUE Code_for_diabetic_brittleness.
INDIVIDUAL code_for_diabetic_well_controlled ∈
      Code_entry,
      is_subcode_of VALUE Code_for_diabetic_brittleness.
```

Figure 5a: The code individuals corresponding to Figure 4.

```
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.
```

Figure 5b: Classes of codes defined from code individuals. The first class corresponds to the shaded oval on the bottom right of Figure 2.

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 no, or a flawed, 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.

Representing the Binding in OWL

The Model of Meaning – the "Ontology"

Figure 4 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"[6] keyword indicate that each illness of class Diabetes has one, and only one, Brittleness quality. The definition is not closed, so there is nothing in this limited 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.

```
CLASS Coded_Attribute →
  has_code MAX 1 Code.
CLASS Topic → Coded_Attribute.
CLASS Diagnosis → Coded_Attribute.
CLASS Brittleness → Coded_Attribute.
CLASS Condition_data_structure →
  has_attr EXACTLY 1 Topic,
  has_attr EXACTLY 1 Diagnosis.
CLASS Diabetes_data_structure →
  Condition_data_structure,
  has_attr EXACTLY 1 Brittleness.
```

Figure 6a: Basic mapping of data structure model to OWL

```
CLASS Placeholder_cls_diabetes_only_code   → Code.
CLASS Placeholder_cls_diabetes_or_subcode→ Code.
CLASS Placeholder_cls_for_diabetic_brittleness_subcode
                                                    →Code.
```

Figure 6b: Placeholder code classes for use in Code Binding Interface (CBI)

```
CLASS Diabetes_data_structure →
  has_attr ONLY (Topic & has_code SOME
    Placeholder_cls_diabetes_only_code),
  has_attr ONLY (Diagnosis & has_code SOME
    Placeholder_cls_diabetes_or_subcode),
  has_attr ONLY (Brittleness & has_code ONLY
    Placeholder_cls_diabetic_brittleness_subcode).
```

Figure 6c: Use of placeholder code classes and indication of whether codes are mandatory (*SOME*) or optional (*ONLY*).

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 transformation of the of the ontology to the coding system.

From the ontology fragment in Figure 4, a mirroring profile might specify a definitions of individual codes as shown in Figure 5a. Based on these definitions of individual codes, we can define classes of codes as shown in Figure 5b. 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. For example, there is

[6] "EXACTLY" is an OWL 1.1 construct

```
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 7: Code Binding Interface for Code System in Fig 5 and Information Model in Fig 6.

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 Diabetes have a quality Brittleness.

The basic information model

A basic OWL model capturing the structure implied in Figure 3 is shown in Figure 6. We assume that we are modelling a class of diabetic data structures which have attributes for each item in Figure 2: topic, diagnosis, and brittleness.

The basic OWL mapping is then shown in Figure 6. 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 of Condition_data_structures that all have Topic and Dagnosis attributes, but that the Brittleness attribute is specific to the class of 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) might seem simpler, mapping each attribute (and each association in the complete representation) to its own class makes it easier to specify cardinality and closure at the level of detail required for HL7 and OpenEHR models.

Constraining the codes to placeholders

Given the basic information model defined in Figure 6a, 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 6b.

Given the placeholder classes of codes, we can then use them in general constraints on the information model as shown in Figure 6c. 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 ONLY have particular codes does not imply that it need have any such codes).

```
CLASS Qualifier_name_code → Code.
INDIVIDUAL code_for_diabetic_brittleness_qualifier ∈
                                        Qualifier_name_code.
CLASS Code_for_diabetes_and_subcodes →
   has_qualifier ONLY
                    {code_for_diabetic_brittleness_qualifier}.
INDIVIDUAL code_for_diabetic_brittleness_qualifier ∈
   has_code EXACTLY 1
       Subcode_of_code_for_diabetic_brittleness.
```

Figure 8a: Extension of Model of Codes to qualifiers

```
CLASS Placeholder_diabetes_or_subcode_class ←→
  ({code_for_diabetes} OR
       is_subcode_of VALUE code_for_diabetes),
   NOT (has_qualifier VALUE
                    code_for_diabetic_brittleness_qualifier).
```

Figure 8b: Extension of CBI in Figure 7 to exclude codes qualified by brittleness

The Code Binding Interface

The model of the coding system in Figure 5 and the information model in Figure 6 might reside in separate modules. It now remains to define the Code Binding Interface between the two modules, which might likewise to reside in a third module.

The Code Binding Interface (CBI) consists of logical equivalences between the placeholder classes defined in Figure 6b and formal definitions of classes of codes in terms of the individuals in the model of codes in Figure 5. A CBI consistent with the constraints in Figure 3 is shown in Figure 7. The first line indicates that the placeholder class consists of just the 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. (Remember that 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.

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. The simplest way to do this is to exclude the use of codes including the Brittleness qualifier from use with the Diagnosis attribute. The constraints depend only on whether the coding system model contains the necessary definitions. The methodology is the same whether it is for named, predefined (pre-coordinated) or (post-coordinated) code expressions ("code phrases" in HL7).

To represent compositional coding systems in OWL, we need to extend the definitions of the coding system to say that any code for diabetes or its subcodes may be linked to a qualifier view by the property has_qualifier by at most one brittleness qualifier code which, if present, must be linked to a subcode of code_for_diabetic_brittleness. To do this we need a new class of codes, the Qualifier_name_code with an instance code_for_diabetic_brittleness_qualifier.

Using this scheme we extend Figure 5 as shown in Figure 8a. This is an extension of the coding system model, not of the information system model (nor of the model of meaning).

Given the definitions in Figure 8a, we can extend the Code Binding Interface in Figure 7 by extending the definition of the placeholder for the for the diabetes_or_subcode to exclude codes qualified by brittleness as shown in Figure 8b.

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.

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 they are different. In OWL, any two individuals might be the same unless declared different and any two classes might overlap unless declared disjoint.

Therefore, to represent the intentions fully, we need a set of "differentiating axioms" examples of which are shown in Figure 9abc. If these axioms are omitted, the validation in the next section will be incomplete because the reasoner will never infer that a code as incorrect because it cannot infer that it is different from the correct code, even though it has a different name.

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 as intended. Taking into account the previous discussion, the complete procedure consists of the following steps:

```
DISJOINT Diabetes_type_1, Diabetes_type_2.
DISJOINT Brittle, Well_controlled.
```

Figure 9a: 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 9b: Differentiating axioms for the model of codes – the coding system

```
DISJOINT Data_structure, Attribute.
DISJOINT Topic, Diagnosis, Brittleness.
```

Figure 9c: Differentiating axioms for the information model.

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 4 and 5.
2. Map the information model to and OWL model including the constraints on the terminology to be used as placeholders following the example in Figures 6.
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 7.
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 [2]).

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 which cannot be found in the data base 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 which 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 EHR formalisms assume that the 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 Figure 6 we need to create a new subclass of "complete diabetes data structures" with the added the closure axiom. The new subclass definition is shown in Figure 10a. The second clause is the "closure axiom" that says that only these three attributes may occur.

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

Figure 10a: "Complete" subclass of the Diabetes data structure class with closure axiom

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 (MAX 0) subcodes.

```
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 ∈
   has_subcode MAX 0.
INDIVIDUAL code_for_diabetes_type_2 ∈
   has_subcode MAX 0.
```

Figure 10b: Closure axioms for code for diabetes

Step c: An OWL mapping of a data structure that conforms to the model in Figures 6 is shown in Figure 10c. The final line is the closure axiom. (The use of SOME and ONLY rather than VALUE avoids the need to define individuals for each data structure's Topic, Diagnosis and Brittleness attributes.)

```
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 VALUE
                                 code_for_diabetic_brittle),
  has_attr ONLY (Topic OR Diagnosis OR Brittleness).
```

Figure 10c: The OWL mapping of a Diabetic data structure including closure axiom.

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 10c.
2. Add closure axiom as shown in Figure 10c.
3. Use the reasoner to check if the data structure is a valid instance of the intended class in the information model.

Limitations of OWL

OWL-DL is based on a subset of first order logic deliberately limited so that inference is computationally tractable. 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 datatype in "holders". What OWL provides is a check on the constraints on which data types should be used where. Separate datatype syntax checkers will be required to check the datatype formats themselves.
- *Lack of variables.* To preserve computational tractability, 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.

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.

Discussion

In previous papers [3-5] we have identified the interface between models of meaning – the ontology– and models of use as critical to clinical systems. This paper clarifies the relation between the model of meaning and one sort of model of use, the information model used for validating EHRs and messages. It contends that these information models are, in fact, models of data structures, and that they are formulated at a meta level with respect to the model of use, the ontology proper. It further contends that codes are likewise data structures and that the model of codes, or coding system, is likewise at a meta-level with respect to the model of meaning – the ontology.

The paper illustrates a methodology for formulating a "Code Binding Interface" (CBI) to specify and constrain how codes are to be used in data structures. This task is essentially "syntactic" – it is concerned with whether the data structures 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 rather than on the meaning itself. 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.

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 which 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 7, 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 two steps in using patient information – formulating meanings and storing or transmitting meanings in data structures. The third step – using the information for clinical decisions about individual patients – 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 allows systems to interoperate using valid message despite flaws in such models. 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. However, potential alternatives might include F-Logic [7], broader epistemic extensions to OWL and description logics [8] or other epistemic and or higher order logics. A principled layered version of OWL similar to that in this paper has also been suggested by others [6]. 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

1. Beale T. Archetypes: Constraint-based domain models for future-proof information systems. In: *OOPSLA-2002 Workshop on behavioural semantics*; 2002;
2. Rector A, Drummond N, Horridge M, Rogers J, Knublauch H, Stevens R, et al. OWL Pizzas: Common errors & common patterns from practical experience of teaching OWL-DL. In: *EKAW-2004*; October, 2004; Northampton, England: Springer; p. 63-81.
3. Rector AL. The Interface between Information, Terminology, and Inference Models. In: Patel V, (ed) *Proc Medinfo-2001*; London, England; p. 246-250.
4. Rector AL, Johnson PD, Tu S, Wroe C, Rogers J. Interface of inference models with concept and medical record models. In: Quaglini S, Barahona P, Andreassen S, editors. *Artificial Intelligence in Medicine Europe (AIME*); 2001; Cascais, Portugal: Springer. p. 314-323.
5. Rector A, Taweel A, Rogers J. Models and inference methods for clinical systems: A principled approach. In: *Proc Medinfo-2004*; San Francisco: North Holland; p. 79-83
6. Pan JZ, Horrocks I, Schreiber G. OWL FA: A metamodeling extensions of OWL DL. In: *Proc OWL-ED 2005*
7. Kifer, M, Lausen G. F-logic: a higher-order language for reasoning about objects, inheritance, and schemas. Portland, Oregon, United States: ACM Press; 1989.
8. Donini F, M L, Nardi D, Shaerf A, Nutt W. An epistemic operator for description logics. *Artificial Intelligence* 1998;100(1-2):225-274.