Mapping Archetypes to OWL *

Ozgur Kilic, Veli Bicer, Asuman Dogac

Software Research and Development Center
Middle East Technical University (METU)
06531 Ankara Turkiye
email: asuman@srdc.metu.edu.tr

ABSTRACT

In [6] and [8], the OWL representations of some archetypes are given. The work described in this paper is meant to complement those work.

1. ARCHETYPE DEFINITION LANGUAGE (ADL)

Archetypes are constraint-based models of domain entities and each archetype describes configurations of data instances whose classes conform to a reference information model. Having a small but generic reference information model helps the EHR system to handle many different medical concepts. Yet the small number of generic concepts in the reference information model is not enough to describe the semantics of the domain specific concepts: these are described through archetypes.

An archetype is composed of three parts: header section, definition section and ontology section. Header section contains a unique identifier for the archetype, a code identifying the clinical concept defined by the archetype. The header section also includes some descriptive information such as author, version and status. Definition section contains the restrictions in a tree like structure created from the reference information model. This structure constrains the cardinality and content of the information model instances complying with the archetype. Codes representing the meanings of nodes and constraints on text or terms, bindings to terminologies such as SNOMED [10] or LOINC [7], are stated in the ontology section of an archetype. A formal language for expressing archetypes is described in [1] known as Archetype Definition Language (ADL).

As an example to an archetype definition in ADL, a part of "Complete Blood Count" archetype definition is presented in Figure 1. The complete ADL definition can be found in [3]. Here the "OBSERVATION" class from the reference information model is restricted to create "Complete Blood Count" archetype, by restricting its CODED_TEXT value to "ac0001" term, (ac0001 term is defined to be "complete blood count" in the constraint_definitions part of the ADL, and declared to be equivalent to Loinc::700-0 term in the term bindings part), and by defining its content to be a list of "Haemoglobin", "Haematocrit" and "Platelet Count" test result elements.

```
OBSERVATION[at1000.1] matches {-- complete blood picture
 name matches {
     CODED_TEXT matches {
         code matches {[ac0001]}  -- complete blood count}}
 data matches {
     LIST_S[at1001] matches {-- battery
        items cardinality matches {0..*} \epsilon {
          ELEMENT[at1002.1] matches {-- haemaglobin
               name matches {
                 CODED_TEXT matches {
                     code matches {[ac0003]} -- haemaglobin}}
               value matches {
                  QUANTITY matches {
                     value matches {0..1000}
                     units matches {^g/l|g/dl|.+^}}}}
          ELEMENT[at1002.2] occurrences matches {0..1} matches
      {-- haematocrit
               name matches {
                  CODED_TEXT matches {
                      code matches {[ac0004]}-- haematocrit}}
               value matches {
                   QUANTITY matches {
                       value matches {0..100}
                       units matches {"%"}}}}
          ELEMENT[at1002.3] occurrences matches {0..1} matches
      {-- platelet count
               name matches {
                  CODED_TEXT matches {
                      code matches {[ac0005]} -- platelet count}}
               value matches {
                   QUANTITY matches {
                       value matches {0..100000}
                       units matches {"/cm^3"}
                      }}}}}}}
```

Figure 1: The ADL definition of "Complete Blood Count" Archetype

A template on the other hand is a directly, locally usable data creation/validation artefact which is semantically a constraint/choice on archetypes, and which often corresponds to a whole form or a screen. Templates in general have a 1:N relationship with underlying concepts, each of which is described by an archetype.

2. WEB ONTOLOGY LANGUAGE (OWL)

Web Ontology Language (OWL) [11] is a semantic

*

markup language developed by the World Wide Web consortium for publishing and sharing ontologies. OWL is based on Resource Description Framework (RDF) [9].

OWL describes the *structure* of a domain in terms of *classes* and *properties*. Classes can be names (URIs) or *expressions* and the following set of *constructors* are provided for building class expressions: *owl:intersectionOf, owl:unionOf, owl:complementOf, owl:oneOf, owl:allValuesFrom, owl:someValuesFrom, owl:hasValue*.

In OWL, properties can have multiple domains and multiple ranges. Multiple domain (range) expressions restrict the domain (range) of a property to the intersection of the class expressions.

Another aspect of the language is the axioms supported. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties [2]. The following are the set of OWL axioms: *rdfs:subClassOf, owl:sameClassAs, rdfs:subPropertyOf, owl:samePropertyAs, owl:disjointWith, owl:sameIndividualAs, owl:differentIndividualFrom, owl:inverseOf, owl:transitiveProperty, owl:functionalProperty, owl:inverseFunctionalProperty*.

3. HOW TO MAP ADL TO OWL?

In this section, we will present how archetypes are represented in OWL by mapping each ADL construct to its OWL counterpart. As mentioned in Section 1, ADL specializes the classes of the generic information model by constraining their attributes. The applicable constraints are as follows [1]:

- Constraints on the range of data-valued properties
- Constraints on the range of object-valued properties
- Constraints on the "existence" of a property indicating whether the property is optional or mandatory .
- Constraints on the "cardinaity" of a property indicating whether the property refers to a container type, the number of member items it must have and their optionality, and whether it has a "list" or a "set" structure.
- Constraints on a property with "occurrences" indicating how many times in runtime data an instance of a given class conforming to a particular constraint can occur. It only has significance for objects, which are children of a container property.

It is also possible to reuse previously defined archetypes and archetype fragments. There are two constructs for this purpose: The first one is the "use_node" construct, which is used to reference an archetype fragment by a path expression. The "use_node" references an archetype fragment within the archetype. The second one is the "allow_archetype" construct, which is used to reference other archetypes by defining a criteria for allowable archetypes.

As described in [1], the first step in representing archetypes in OWL is to construct the reference information model of the domain in OWL. A simple algorithm for object model to OWL mapping is given in [6]. First, each class in the reference information model is represented as an OWL class. Secondly, each relationship is represented as an ObjectProperty and each data-valued property is represented as DatatypeProperty in OWL. Finally, cardinalities of relationships and properties are represented by cardinality restrictions in OWL.

The next step is representing archetypes in OWL, based on the reference information model. In the following subsections, we describe how each ADL construct can be represented in OWL.

3.1 Specializing the Root Class

Basically, an archetype restricts the instances of the domain classes. It has a hierarchical structure and restriction starts with a root class. The root class of the archetype is the class of the instances which are validated against the archetype.

```
...
definition
  PERSON[at0000] matches {
    ...
  }
...
ontology
  primary_language = <"en">
  languages_available = <"en", ...>
  term_definitions("en") = <
    items("at0000") = <
      text = <"Doctor">
      description = <"Doctor of the patient">
   >
...
```

Figure 2: Archetype Representing the Doctor Concept

As an example, in Figure 2, "Person" represents such a root class. "Person" is restricted to the doctor concept in the domain. The "at0000" term binds the "Person" to the "Doctor" concept. This means that this archetype describes the "Person" instances who are doctors. In OWL, such a specialization can be represented by defining a new class for the root class of the archetype. Figure 3 depicts the corresponding OWL class of the archetype. The contents of the "Doctor" class is determined by applying restrictions on the properties inherited from the "Person" class. In other words, properties of the "Person" class of the reference information model have local restrictions inside the "Doctor" class and these local restrictions are derived from the archetype defined in the ADL document. How these local restrictions are handled in OWL is discussed in the following subsections. Note that, archetypes do not introduce new properties in addition to the properties of the reference model. They introduce new classes which define local restrictions on the properties of the imported reference model.

```
<owl:Class rdf:ID="Doctor">
    <rdfs:subClassOf rdf:resource=
     "http://www.sample.org/Domain.owl#Person"/>
    <rdfs:subClassOf rdf:resource=
     "http://www.sample.org/Archetype.owl#Archetype"/>
    ...
</owl:Class>
```

Figure 3: Representing the Doctor Concept in OWL

As a summary, the class "Person" is a reference information model class and the class "Doctor" is introduced to represent the doctor concept defined in the archetype. We

call such a class which are the root classes of archetypes as "Archetype" class in OWL. In Figure 3, "Doctor" is defined as a subclass of this "Archetype" class and hence it inherits attributes from the "Archetype" class in which archetype related properties are defined such as archetype id.

3.2 Constraints on the range of Data-valued Properties

As already mentioned, archetypes also constrain data-valued properties. However certain desired constraints on primitive types can not be directly expressed in OWL. In OWL, it is possible to compare the value of a data-valued property by "hasValue" construct and declare the type of such a property by "rdf:datatype". However, complex constraints such as defining allowable characters and sequences for a string-valued property or defining a maximum and a minimum value for an integer-valued property are not possible.

Yet OWL allows XML Schema to be used as a data type. Therefore, our solution for defining constraints on data-valued propertiesis to use "User-derived datatypes" in XML Schema. "User-derived datatypes" are those datatypes that are defined by individual schema designers [13]. In Figure 4, we present an example to a user-derived dataype which restricts adultAgeType to greater than 18.

```
<xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    targetNamespace="http://www.sample.org/sch/sample"
    xmlns="http://www.sample.org/sch/sample">
   <xsd:simpleType name="adultAgeType">
        <xsd:restriction base="xsd:nonNegativeInteger">
            <xsd:minInclusive value="18"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:schema>
```

Figure 4: An Example User-derived Datatype in XMLSchema

For each constraint on a data-valued property in an archetype, the corresponding user-derived datatype is declared in an XMLSchema. Data-valued properties of the reference information model are constrained by setting their types to user-derived types in the corresponding OWL class. If the data-valued property is restricted to have a value from a set of values, then "owl:oneOf" construct is used to describe the possible values of the property. Figure 5 shows how the user-derived datatypes are referenced from OWL.

```
...
<rdfs:subClassOf>
   <owl:Restriction>
     <owl:onProperty rdf:resource=
          "http://www.sample.org/Domain.owl#age"/>
     <owl:allValuesFrom rdf:resource=
      "http://www.sample.org/sch/sample#adultAgeType"/>
      </owl:Restriction>
</rdfs:subClassOf>
...
```

Figure 5: Referencing user-derived datatypes in OWL

3.3 Constraints on the range of Object-valued Properties

In an archetype hierarchy, nodes which refer to the properties of a class in the reference information model, may have object values, that is, refer to classes in the reference information model.

As an example, in Figure 6, "name" property of the "Person" is an object property. This property is restricted to be an instance of "Person_Name" class which is further restricted in the archetype. In OWL, this is expressed as follows: the range of the "name" Object property is restricted to "DoctorName" class which is a subclass of "Person_Name" class. Furthermore, the constraints defined on the "name" Object property in the archetype are defined as further restrictions in this newly introduced subclass, namely, "DoctorName".

```
PERSON[at0000] matches {
  name matches{
    PERSON_NAME[at0001] matches?{
      ...
    }
  }
}
ontology
  primary_language = <"en">
  term_definitions("en") = <
    items("at0000") = <
      text = <"Doctor">
      description = <" Doctor of the patient ">
    >
    items("at0001") = <
      text = <"DoctorName">
      description = <"Name of the doctor">
    >
...
```

Figure 6: Object-typed node in ADL

As a result, each constraint on an object-valued property introduces a new class in OWL. The range of such a property is defined to be the newly introduced subclass. Figure 7 depicts the corresponding OWL classes of the archetype defined in Figure 6.

```
<owl:Class rdf:ID="Doctor">
  <rdfs:subClassOf rdf:resource=
  "http://www.sample.org/Domain.owl#Person"/>
  <rdfs:subClassOf rdf:resource=
  "http://www.sample.org/Archetype.owl#Archetype"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource="#DoctorName"/>
      <owl:onProperty rdf:resource=
        "http://www.sample.org/Domain.owl#name"/>
    </owl:Restriction>
  </rdfs:subClassOf>
...
</owl:Class>

<owl:Class rdf:ID="DoctorName ">
  <rdfs:subClassOf rdf:resource=
  "http://www.sample.org/Domain.owl#PersonName"/>
...
</owl:Class>
```

Figure 7: Representing Object-valued Properties in OWL

It should be noted that, in our approach, a constraint on an object-valued property generates only a new class in OWL unlike the approach in [6] which also introduces

a new ObjectProperty. Introducing a new ObjectProperty may make it difficult to relate the object properties in the archetype with the object properties in the reference model.

3.4 Representing Existence Constraints

In ADL, an existence constraint on a property shows the optionality of the property. Optionality of a property in OWL can be specified by using minCardinality, maxCardinality and cardinality constructs. If a property has 'minCardinality = 0' and 'maxCardinality =1' then the property is optional. If a property has 'cardinality =1' then the property is required. Figure 8 gives an example of existence constraint in ADL.

```
PERSON[at0000] matches {
  specializedOn existence matches{1..1} matches{...}
  ...
}
...
```

Figure 8: An Example Existence constraint in ADL

In Figure 9, the OWL fragment corresponding to the existence constraint defined in Figure 8 is presented.

```
<owl:Class rdf:ID="Doctor">
   <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource=
           "http://www.sample.org/Domain.owl#specializedOn"/>
        <owl:cardinality rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
...
</owl:Class>
```

Figure 9: Defining Existence Constraints in OWL

3.5 Representing Cardinality Constraints

A cardinality constraint indicates that a property has a container type in ADL. Cardinality defines the number of valid elements by giving an upper and a lower bound to the element count for the container class. In Figure 10, "events" defined as having 'minCardinality = 1' and 'maxCardinality' unbounded. It is also stated that elements in events can be an instance of one of the two subclasses of "Event" class.

```
...
data matches {
  History[at0003] matches {
    events cardinality matches {1..*} matches {
      Event[at0004] matches {...}
      Event[at0005] matches {...}
    }
  }
}
...
```

Figure 10: Cardinality Constraints in ADL

Assume that "at0003", "at0004" and "at0005" concepts mean *OneMonthHistory*, *WeightGain*, *HeightGain* respectively. Then ADL fragment in Figure 10 can be represented in OWL as shown in Figure 11. In class *OneMonthHistory*, the minimum cardinality of "events" is restricted to one and the values of the "events" are restricted to be either *WeightGain* event or *HeightGain* event.

```
<owl:Class rdf:ID="OneMonthHistory">
  <rdfs:subClassOf rdf:resource=
  "http://www.sample.org/Domain.owl#History"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom>
        <owl:unionOf  rdf:parseType="Collection">
          <owl:Class rdf:about="#WeightGain "/>
          <owl:Class rdf:about="#HeightGain"/>
        </owl:unionOf>
      </owl:allValuesFrom>
      <owl:onProperty rdf:resource=
           "http://www.sample.org/Domain.owl#events"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource=
           "http://www.sample.org/Domain.owl#events"/>
      <owl:minCardinality rdf:datatype=
      "http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
...
</owl:Class>

<owl:Class rdf:ID="WeightGain">
  <rdfs:subClassOf rdf:resource=
  "http://www.sample.org/Domain.owl#Event"/>
  ...
</owl:Class>
<owl:Class rdf:ID="HeightGain">
  <rdfs:subClassOf rdf:resource=
  "http://www.sample.org/Domain.owl#Event"/>
  ...
</owl:Class>
```

Figure 11: Representing ADL's cardinality constraint in OWL

3.6 Representing Occurrence Constraints

In ADL, occurrence constraints are defined on the elements of a container class and they constrain the number of each type of element in the container. In Figure 12, an ADL fragment declares that "events"' should contain exactly one instance of "at0001", three instances of "at0003" and at most one instance of "at0002"' concepts.

```
events cardinality matches {*} matches {
  EVENT[at0001] occurrences matches {1} matches {...}
  EVENT[at0002] occurrences matches {0..1} matches {...}
  EVENT[at0003] occurrences matches {3} matches {...} }
```

Figure 12: Occurrence constraint in ADL

Occurrence constraints in ADL correspond to the qualified number restrictions of Description Logics (DLs). DAML+OIL [4] from which the OWL is derived supports the usage of qualified number restrictions with the following language elements: "daml:cardinalityQ", "daml:hasClassQ" ,"daml:maxCardinalityQ", "daml:minCardinalityQ".

OWL is developed as a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language [12]. Unfortunately OWL does not inherit qualified number restrictions from DAML+OIL. Therefore it is not possible to represent occurrence constraints in OWL.

3.7 Representing Invariant Constraints

An invariant constraint is an expression which should be satisfied by all instances of an archetype. Figure 13 gives an

invariant example in ADL, stating that the "value" property should exist when the "code" property exists.

```
Observation[at0000] matches {
    classCode cardinality matches {1} matches {...}
    moodCode cardinality matches {1} matches {...}
    id matches {...}
    code matches {...}
    confidentialityCode matches {...}
    uncertaintyCode matches {..}
    value matches  {...}

invariant:
  basic_validity: exists code implies exists value }
```

Figure 13: Invariant in ADL

Since representing invariants is not directly addressed in OWL, we propose to represent them outside OWL. There are several ways of representing invariants in XML documents. One of the solutions is to use XSLT/XPath stylesheets. We propose defining invariants through XSLT/XPath stylesheets and executing these invariant constraints over OWL instances using XSLT processors. Figure 14 depicts how the invariant constraint of Figure 13 can be expressed through XSLT.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
  <xsl:output method="text"/>
  <xsl:template match="/">
    <xsl:if test= "boolean(/Observation/code)">
      <xsl:if test= "not(boolean(/Observation/value))">
        <xsl:text>Error! code exists without value</xsl:text>
      </xsl:if>
    </xsl:if>
    <xsl:if test= "boolean(/Observation/code)">
      <xsl:if test= "boolean(/Observation/value)">
        <xsl:text>Instance document is valid</xsl:text>
      </xsl:if>
  </xsl:template>
</xsl:stylesheet>
```

Figure 14: Representing Invariant using XSLT

3.8 Representing Internal References

The aim of internal references in ADL is not to repeat a previously defined constraint in an archetype. Internal references are indicated by path expressions in ADL. Such references can be enforced by reusing the classes or data types in OWL.

3.9 Representing Archetype Reuse

In ADL "allow_archetype" construct is used to reference other archetypes by defining a criteria for allowable archetypes. An example of "allow_archetype" is presented in Figure 15. The elements of the "items" list described in Figure 15 are instances of "ENTRY" class of the reference information model and they are constrained by archetype classes whose id matches the indicated representation.

```
SECTION[at2000] occurrences matches {0..1} matches {
  items cardinality matches {0..*} matches {
    allow_archetype ENTRY occurrences matches  {0..1} matches {
      include
        id matches {/.*\.iso-ehr\.entry\..*\..*/}
    }
  }
}
```

Figure 15: Archetype reuse in ADL

In OWL the range of the "items" property is restricted in such a way that values of "items" are instances of "ENTRY" and also instances of an "Archetype" whose id has a pattern restriction. Such a range restriction can be defined by introducing a new class, and setting the range of the "items" to this class. This new class is subclass of "ENTRY" and restricts the "archetypeID" by setting its range to a user-derived type which defines the pattern restriction.

OWL class defined in Figure 16 can be used for this purpose to implement the "allow_archetype" constraint in OWL for the example in Figure 15.

```
<owl:Class rdf:ID="ReusedArchetype1">
  <rdfs:subClassOf rdf:resource=
     "http://www.sample.org/Domain.owl#Entry"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:allValuesFrom rdf:resource=
        "http://www.sample.org/ArchetypeSchema#entryPattern"/>
      <owl:onProperty rdf:resource=
        "http://www.sample.org/Archetype.owl#archtypeID"\>
    </owl:Restriction>
  </rdfs:subClassOf>
...
</owl:Class>
```

Figure 16: Implementing archetype reuse in OWL

4. REFERENCES

[1] Archetype Definition Language (ADL) 1.2 draft, http://www.openehr.org/drafts/ADL-1_2_draftF.pdf

[2] Baader, F. , Horrocks, I., Sattler, U., "Description Logics", Handbook on Ontologies, 2004.

[3] Complete Blood Count Archetype ADL Definition, http://www.openehr.org/repositories/archetype-dev/-adl_1.1/adl/archetypes/openehr/ehr/entry/openehr-ehr-observation.haematology-cbc.draft.adl.html

[4] DAML+OIL Reference Description. Dan Connolly, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. http://www.w3.org/TR/daml+oil-reference.

[5] Extending XML Schemas,

[6] HL7 Template and Archetype Architecture Version 3.0, http://www.hl7.org/library/committees/template/-HL7_Atlanta_10_20_04.doc

[7] Logical Observation Identifiers Names and Codes (LOINC), www.loinc.org

[8] openEHR Community, http://www.openehr.org/

[9] RDF Semantics, http://www.w3.org/TR/rdf-mt/

[10] SNOMED Clinical Terms, http://www.snomed.org/snomedct_txt.html

[11] Web Ontology Language OWL, http://www.w3.org/TR/owl-features/

[12] Web Ontology Language OWL Reference, Mike Dean and Guus Schreiber, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[13] XML Schema Part 2: Datatypes. Paul V. Biron