Archetype-Based Semantic Interoperability of Web Service Messages in the Health Care Domain

Veli Bicer, Middle East Technical University (METU), Turkey
Ozgur Kilic, Middle East Technical University (METU), Turkey
Asuman Dogac, Middle East Technical University (METU), Turkey
Gokce B. Laleci, Middle East Technical University (METU), Turkey

ABSTRACT

In this article, we describe an infrastructure enabling archetype-based semantic interoperability of Web Service messages exchanged in the health care domain. We annotate the Web Service messages with the OWL representation of the archetypes. Then, by providing the ontology mapping between the archetypes, we show that the interoperability of the Web Service message instances can be achieved automatically. An OWL mapping tool, called OWLmt, has been developed for this purpose. OWLmt uses OWL-QL engine, which enables the mapping tool to reason over the source archetype instances while generating the target archetype instances according to the mapping patterns defined through a GUI.

Keywords: Please provide

INTRODUCTION

Health care is one of the few domains where sharing information is the norm rather than the exception (Heard, Beale, Mori & Pishec, 2003). On the other hand, today there is no universally accepted standard for the digital representation of clinical data. There is a multitude of medical information systems storing clinical information in all kinds of proprietary formats.

We address this interoperability problem within the scope of the ARTEMIS project by wrapping and ex-

posing the existing health care applications as Web Services. However, given the complexity of the clinical domain, the Web Service messages exchanged have numerous segments of different types and options. To make any use of these messages at the receiving end, their semantics must be clearly defined.

In a previous effort described in Dogac et al. (in press), we annotated Web Services through the reference information models of Electronic Healthcare Record (EHR) standards. EHR standards define the interfaces for clinical content exchange. The prominent EHR standards include openEHR (openEHR Community, 2005), HL7 CDA (HL7 Clinical Document Architecture, 2004), and CEN TC/251 prEN 13606-1 (referred to as EHRcom) (CEN TC/251 prEN 13606-1, 2004). Although such an approach allowed us to achieve a certain degree of interoperability, there were further problems to be addressed as follows:

- The reference information models of EHRs contain generic classes rather than having a class for each specialized clinical concept. Therefore, given a class in source ontology, the corresponding class in the target ontology is not clear unless the context is known. For example, an instance of an ENTRY class in EHRcom corresponds to one of the instances of ACT or ORGANIZER or OBSERVATION or PROCEDURE classes in HL7 CDA.
- Another problem in mapping reference information models one into another is as follows: different reference information models structure their classes differently. As an example, both CEN EHRcom and HL7 CDA have a class name called SECTION, and sections can have nested sections. When the sections of a clinical document are organized differently, then generating the same hierarchy for the target domain as in the source domain would not be correct.

In this article, we address these problems by using archetypes to complement the work described in Dogac et al. (in press). An archetype is a reusable, formal expression of a distinct, domain-level concept such as *blood pressure*, *physical examination*, or *laboratory result*, expressed in the form of constraints on data whose instances conform to some reference information model (Beale & Heard, 2003). The reference information model can be CEN EHRcom (CEN TC/251 prEN 13606-1, 2004), openEHR (openEHR Architecture Specifications, 2005), or the HL7 CDA schema (HL7 Clinical Document Architecture, 2004).

We use the Web Ontology Language (OWL) (OWL, 2004) representation of the archetypes to semantically annotate the Web Service messages. We then provide the mapping between the OWL representations of archetypes through an OWL ontology mapping tool called OWLmt (OWLmt, 2005). The mapping definition produced by OWLmt is used by OWLmt engine to automatically transform the Web Service message instances one into other when two health care institutes conforming to different archetypes want to exchange messages.

RELATED WORK

Semantic heterogeneity occurs when there is a disagreement about the meaning, interpretation, or intended use of the same or related data (Sheth & Larsen, 1990). Since medical information systems today store clinical information about patients in all kinds of proprietary formats, there is a need to address the interoperability problem. For this purpose, several EHR standards that allow the structure of clinical content for the purpose of exchange are currently under development. A very detailed survey and analysis of electronic health care records is presented in Eichelberg, Aden, Dogac, and Laleci (2005).

However, since there are more than one electronic health care record standards, the semantic heterogeneity problem is still unavoidable among health care systems. HL7 and openEHR offer different reference information models for the health care domain. For example, an instance of an ENTRY class in openEHR corresponds to one of the instances of ACT, ORGANIZER, OBSERVATION, or PROCEDURE classes in HL7 CDA.

Two approaches are described in Kashyap and Sheth (1996) for providing interoperability based on ontologies. One is to build a common ontology; the other is reusing existing ontologies and combining them. Instead of building a common ontology, we resolve the semantic heterogeneity among health care standards by reusing existing ontologies and combining them through ontology mapping, which allows the exchange of information among health care information systems conforming to different standards.

ARCHETYPES AND REPRESENTING ARCHETYPES IN OWL

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, which are described through archetypes.

An archetype is composed of three parts: header section, definition section, and ontology section. The 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. The 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 (SNOMED Clinical Terms, 2005) or LOINC (LOINC, 2005), are stated in the ontology section of an archetype. A formal language for expressing archetypes (i.e., Archetype Definition Language [ADL]) is described in ADL (2003).

As already mentioned, ADL specializes the classes of the generic information model by constraining their attributes. The applicable constraints are as follows (ADL, 2003):

- 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 cardinality 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 criteria for allowable archetypes. 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 Complete Blood Count Archetype ADL Definition (2005). 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 as *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.

In ARTEMIS architecture, OWL representations of the archetypes are exploited. OWL describes the structure of a domain in terms of classes and properties. Classes can be names (URIs) or expressions. The following set of constructors is 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 (Baader, Horrocks, & Sattler, 2004). The following are the set of OWL axioms: rdfs:subClassOf, owl:sameClassAs, rdfs:subPropertyOf, owl:sameProperty As,

Figure 1. The ADL definition of complete blood count archetype

```
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"}
                                }}}}}}}
```

owl:disjointWith, owl:sameIndividualAs, owl:differentIndividualFrom, owl:inverseOf, owl:transitiveProperty, owl:functional Property, owl:inverse FunctionalProperty.

In HL7 Template and Archetype Architecture Version 3.0. (2003) and openEHR Community (2005), the OWL representations of reference information models of archetypes are given. The first step in representing archetypes in OWL is to construct the reference information model of the domain in OWL. A simple algorithm for mapping object model to OWL is given in HL7 Template and Archetype Architecture Version 3.0. (2003). First, each class in the reference information model is represented as an OWL class. Second, 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 as described in ADL (2003) and HL7 Template and Archetype Architecture Version 3.0. (2003). As stated in HL7 Template and Archetype Architecture Version 3.0. (2003), each ADL object node generates an OWL class declaration. Object-valued properties are restricted through these OWL classes.

ARTEMIS SEMANTIC INFRASTRUCTURE

The aim of the ARTEMIS project (ARTEMIS Consortium, 2004) is to allow health care organizations to keep their proprietary systems and yet expose the functionality of their applications through Web Services. ARTEMIS has a peer-to-peer infrastructure to facilitate the semantic discovery of Web Services and service registries (Dogac et.al., in press).

The full sharability of data and information requires two levels of interoperability:

- The functional (syntactic) interoperability, which is the ability of two or more systems to exchange information. This involves agreeing on the common network protocols, such as Internet or Value Added Networks; the transport binding such as HTTP, FTP, or SMTP and the message format like ASCII text, XML (Extensible Markup Language) or EDI (Electronic Data Interchange). Web Services provide functional interoperability through well-accepted standards like SOAP (2003) and WSDL (2005). However, note that in order to access and consume Web Services through programs, you must know their operational and message semantics in advance.
- Semantic interoperability is the ability for information shared by systems to be understood at the level of formally defined domain concepts so that the information is computer processable by the receiving system. In other words, semantic interoperability requires the semantics of data to be defined through formally defined domain-specific concepts in standard ontology languages (ISO TC/215, 2003).

To provide semantic interoperability in ARTEMIS, the Web Services are annotated with the following semantics:

- Operational semantics of Web Services. In order to facilitate the discovery of the Web Services, there is a need for semantics to describe what the service does; in other words, what the service functionality semantics is in the domain. For example, in the health care domain, when a user is looking for a service to admit a patient to a hospital, the user should be able to locate such a service through its meaning, independent of what the service is called and in which language it is in. Note that WSDL (2005) does not provide this information.
- In ARTEMIS, HL7 categorization of health care events are used to annotate Web Service functionality, since HL7 exposes the business logic in the health care domain. If further ontologies are developed for this purpose, they easily can

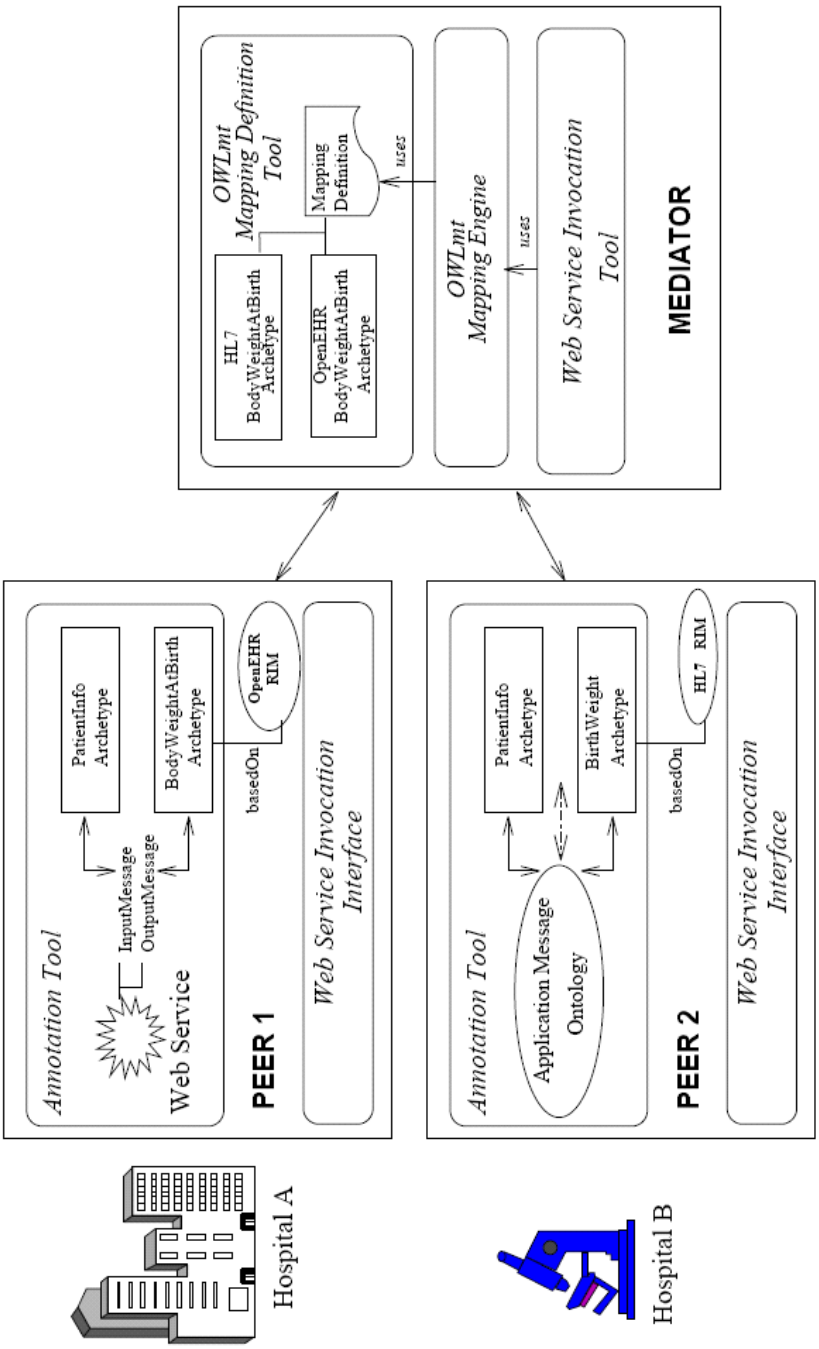
be accommodated in the ARTEMIS architecture through ontology mapping.

- Message semantics of Web Services. When invoking a Web Service, there is also a need to know the meaning associated with the messages or documents exchanged through the Web Service. In other words, service functionality semantics may suffice only when all the Web Services use the same message standards. For example, a GetClinicalInformation Web Service may include the messages to pass information on diagnosis, allergies, encounters, and observation results about a patient. Unless both the sending and the receiving ends of the message conform to the same EHR standard, interoperability cannot be achieved.

ARTEMIS proposes to semantically enrich the Web Service messages through archetypes. As depicted in Figure 2, through an annotation tool provided by the ARTEMIS infrastructure, the health care institutes can annotate the input and output messages of their Web Services with archetypes. For example, Hospital A in Figure 2 declares that its Web Service accepts a *PatientInfo Archetype Instance based on OpenEHR RIM* as an input and returns a *BodyWeightAtBirth Archetype Instance based on OpenEHR RIM* as an output. Note that the consumer application of the Web Service may be compliant with another standard. ARTEMIS enables the service consumers to speak their own language. For this purpose, the annotation tool enables the health care institutes to define their application message schemas in terms of archetypes. For example, Hospital B in Figure 2 declares that its messaging structure will provide and accept PatientInfo and BirthWeight information as archetype instances based on HL7 RIM while invoking the Web Services provided in the ARTEMIS network.

In the ARTEMIS architecture, the OWL representations of archetype definitions and instances are used. To interoperate the archetype instances based on different HER standards, the ARTEMIS mediator provides an OWL mapping tool called OWLmt. Through a graphical interface, OWLmt tool enables the user to define the mappings between archetype definitions, and the resulting mapping definitions are stored at the mediator. When a health care institute wants to join the ARTEMIS network, they advertise their Web Services to the mediator by semantically annotating them through archetypes. When one of the health care institutes wishes to invoke a Web Service provided by another institute in the ARTEMIS Network, the Web Service invocation request is delivered to the mediator. The health care institute provides the Web Service input to the mediator in terms of the archetype instances it conforms. Then the mediator invocation tool consults the OWLmt Mapping Engine to transform the archetype instances from one EHR reference information model standard to another, using the mapping definitions that previously have been generated through the OWLmt Mapping Definition Tool. Finally, the Web Service is invoked with the archetype instance to which the provider conforms. The output of the Web Service is processed in the same manner and presented to the requester as an archetype in-

Figure 2. Artemis semantic architecture



stance based on the EHR standard to which the requester conforms. In the following sections, the details of this process are elaborated through examples.

ARCHETYPE-BASED INTEROPERABILITY OF WEB SERVICE MESSAGES

Since there is more than one EHR standard such as openEHR (openEHR Community, 2005), HL7 CDA (HL7 Clinical Document Architecture, 2004), and CEN EN 13606 EHRcom (CEN TC/ 251 prEN 13606-1, 2004), each with different reference information models and archetypes, annotating Web Service messages with archetypes does not solve the interoperability problem.

Therefore, we need to transform archetypes of one standard into another through ontology mapping. For this purpose, we use the OWL representation of both the involved reference information models and the archetypes. Then, through an OWL ontology mapping tool that we developed, called OWLmt, we map the reference information models and the archetype schemas one into other. Once such a mapping is achieved, OWLmt automatically transforms a Web Service message annotated with an archetype in one standard into another.

In this section, we explain this process through a running example. For this purpose, we first generate the OWL descriptions of an archetype based on openEHR and another one based on HL7. We then present the OWL mapping tool and depict its functionality through the running example.

Example OpenEHR and HL7 Archetypes in OWL

Figure 3 depicts an archetype in ADL that represents the *body weight at birth* concept. This concept is described by restricting the OBSERVATION class in openEHR Reference Model.

The OWL representation of the archetype in Figure 3 is presented in openEHR Body Weight at Birth Archetype OWL Definition (2005). In brief, each restriction on an object-valued property introduces a new class, which is a subclass of the class on which the restriction is defined in the ADL document. For the example, in Figure 3, the data property of the OBSERVATION class is defined as having a type HISTORY, which is further restricted. In OWL, this restriction on history class is handled by introducing a subclass of history called *body weight at birth histor*y. On the other hand, each restriction on a data-valued property either introduces a user-derived datatype for further restricting datatype of the property or produces *owl:hasValue* or *owl:oneOf* restrictions on the property for restricting the value of the property to one value or set of values, respectively. Note that user-derived datatypes can be represented in XML schema and referenced from the OWL representation of the archetype.

Figure 4 depicts the *body weight at birth* ADL archetype based on the HL7 Version 3 Reference Information Model, whose OWL representation is presented in HL7 Body Weight at Birth Archetype OWL Definition (2005).

Figure 3. An example body weight at birth OpenEHR archetype in ADL

```
archetype
        openEHR-EHR-OBSERVATION.weight-birth.v1
specialize
        openEHR-EHR-OBSERVATION.weight.v1
concept
        [at0000.1] -- Body weight at birth
description
...
definition
 OBSERVATION[at0000.1] matches { -- Body weight at birth
  data matches {
   HISTORY[at0002] matches { -- history
    events cardinality matches {1..1; ordered} matches {
     EVENT[at0003] matches { -- Birth
      data matches {
       Simple[at0001] matches { -- Birth simple
        item matches {
         ELEMENT[at0004.1] matches { -- Birth weight
          value matches {
           C_QUANTITY
              property = <"mass">
              units = <"kg">
              magnitude = <|0.0..10.0|>
  } } } } } } } } }
  state matches {0..1} matches {
   List[at0008] matches { -- state structure
    items cardinality matches {1..1; ordered} matches {
     ELEMENT[at0009] occurrences matches {0..*} matches {
     -- Clothing
      value matches {
       CODED_TEXT matches {
          code matches {[local::
          at0010, -- Dressed
          at0011] -- Naked
       }
       assumed_value matches {"at0011"}
  } } } } } }
  other_participations matches {0..1} matches{
    List [at0014] matches { -- participation structure
     items cardinality matches {1..1; ordered} matches {
      PARTIPICATION [at0012] matches{ --Baby
       function matches {
        CODED_TEXT matches {
         code matches {
         [local::at0013] -- Patient
        }} } } } } } }      ...
```

Figure 4. An example body weight at birth HL7 archetype in ADL

```
archetype
        HL7-OBSERVATION.weight-birth.v1
specialize
        HL7-OBSERVATION.weight-birth.v1
concept
        [at0000.1] -- Body weight at birth
description
        author = <"Veli Bicer <veli@srdc.metu.edu.tr>">
        submission = <
                organisation = <"METU-SRDC">
                date = <2005-01-10>
        >
                version = <"version">
        status = <"draft">
        revision = <"1.0">
        description("en") = <
                purpose = <"Describe the observation for the body weight at birth">
                use = <"">
                misuse = <"">
        >
        adl_version = <"1.2">
        rights = <"">

definition
        Observation[at0000] matches { -- birth_weight
        classCode cardinality matches {1} matches {[hl7_ClassCode::OBS]}
        moodCode cardinality matches {1} matches {[hl7_ClassCode::EVN]}
        id matches {*}
                code cardinality matches {1} matches {[at0001],[at0002]}
        confidentialityCode cardinality matches {1..*} matches
        {[hl7_Confidentiality::N]}
        uncertaintyCode matches {[hl7_ActUncertainty::N]}
        value cardinality matches {1} matches {/.*kg[^ ]/}
        hasParticipation cardinality matches {1..*} matches{
          Participation matches{
            hasRole cardinality matches {1..*} matches{
              Patient{
                classCode cardinality matches {1} matches
                {[hl7_ClassCode::PAT]}
                player cardinality matches {1..*} matches{
                  Person matches{
                    classCode cardinality matches {1} matches
                    {[hl7_ClassCode::PSN]}
        }}}}}}
}
...
```

ONTOLOGY MAPPING

In the *Example OpenEHR and HL7 Archetypes in OWL* subsection, we present ADL descriptions of two archetypes. As depicted in Figures 3 and 4, the archetypes differ in terms of structure and format of the data they represent. The main cause of this difference is that the archetypes refer to different reference models (i.e., openEHR RIM and HL7 RIM). Thus, the interoperability between these archetypes becomes a difficult task, although they represent the same concept — *weight at birth*.

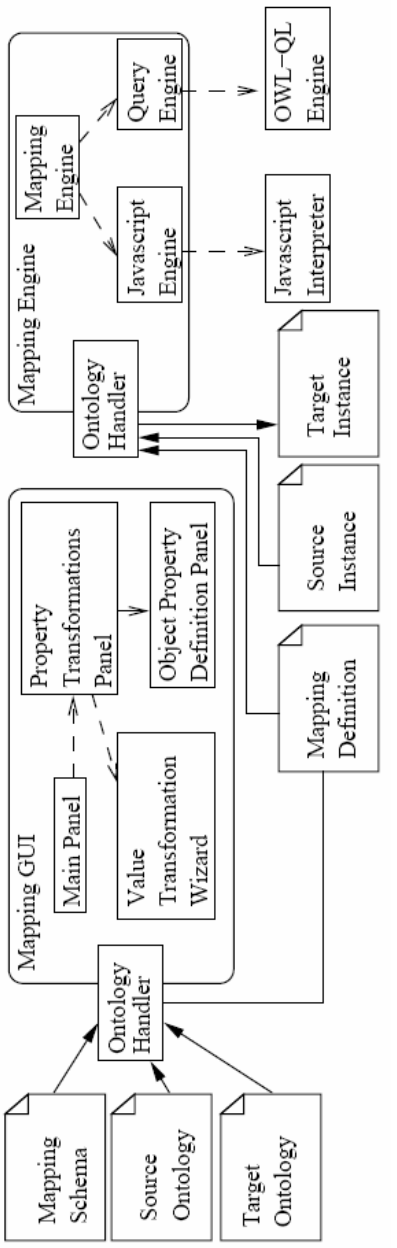
ARTEMIS mediator provides an ontology mapping tool — OWLmt — that enables us to define the mapping between different OWL schemas. In this section, we describe an ontology mapping process in OWL to achieve the interoperability between the archetypes based on different reference models. Once such a mapping definition is stored at the mediator, the mediator will interoperate the Web Service messages represented as archetypes between the health care institutes conforming to different EHR standards.

Ontology mapping is the process where two ontologies with an overlapping content are related at the conceptual level to produce a mapping definition. The source ontology instances then are automatically transformed into the target ontology instances according to the mapping definitions. The architecture of the OWLmt tool (see Figure 5) allows mapping patterns to be specified through a GUI. These patterns are stored in a document called *Mapping Definition*. The mapping engine uses the Mapping Definition to automatically transform source ontology instances into target ontology instances.

The OWLmt mapping tool has the following mapping capabilities:

- Matching the source ontology classes to the target ontology classes. We have developed the following four conceptual mapping patterns to represent the matching between the classes of the source and target ontology classes: EquivalentTo, SimilarTo, IntersectionOf, and UnionOf. The identical classes are mapped through EquivalentTo pattern. SimilarTo implies that the involved classes have overlapping content. As an example, the *body weight at birth* class which is a subclass of *observation* class in the openEHR archetype is similar to the *birth weight*, class which is inherited from the *observation* class in the HL7 archetype, since they both represent the *weight at birth* concept. The SimilarTo patterns in OWLmt are represented in OWL (see Figure 6). How similar classes are further related is described through property mapping patterns.
  The IntersectionOf pattern creates the corresponding instances of the target class as the intersection of the declared source class instances. Similarly, the UnionOf pattern implies the union of the source classes' instances to create the corresponding instances of the target class.
  In some cases, a class in a source ontology can be more general than a class in the target ontology. In this case, the

Figure 5. Architecture of OWLmt



instances of the source ontology that make up the instances of the target ontology are defined through Knowledge Interchange Format (KIF) (2005) conditions to be executed by the mapping engine. As an example, assume that a SimilarTo pattern is defined between the *body weight at birth* class of the openEHR archetype and the *birth weight* class of HL7 archetype. The

body weight at birth class in the openEHR archetype has a property *state* with the cardinality of zero or one (see Figure 3). On the other hand, the *code* property of the *birth weight* class of the HL7 archetype (see Figure 4) has either LOINC (2005) value of 8351-9 (weight at birth with clothes) or LOINC value of 8350-1 (weight at birth without clothes), depending on the value of the *code* value under the *state* property in the openEHR archetype. However, the *code* value is mandatory in the HL7 archetype, unlike the optionality of the *state* property in the openEHR. Therefore, we add a condition in KIF format to the SimilarTo pattern (see Figure 6) to ensure that there exists at least one *state* property of the *body weight at birth* instance in order to map it to an instance of *birth weight* class.

Matching the source ontology Object Properties to target ontology Object Properties. ObjectPropertyTransform pattern is used to define the matching from one or more object properties in the source ontology to one or more object properties in the target ontology. As an example, consider the openEHR archetype in the *Example OpenEHR and HL7 Archetypes in OWL* subsection. According to the openEHR specifications (openEHR Architecture Specifications, 2005), the *body weight at birth* class has an *other participations* object property inherited from the *observation* class, referring to a list of the *participation* class in order to represent the parties that participate in the *body weight at birth* observation. With the help of this object property, we have defined a path from *body weight at birth* class to the PARTY REF in order to state the patient who is involved in this particular observation. Likewise, in the HL7 archetype, there is also a path from the *birth weight* class to the *person* with a set of object properties such as hasParticipation, hasRole, and player. Although these two paths have different structures and involve different properties (e.g., other participations and hasRole) and classes (e.g., List in openEHR and Patient in HL7), they represent the same content; that is, patient of an observation (see Figure 7). Therefore, in the mapping process, an ObjectPropertyTransform pattern is

Figure 6. An example SimilarTo pattern

```
<SimilarTo rdf:about= "http://www.srdc.metu.edu.tr/Map#SimilarTo_1">
        <similarToInput rdf:resource= "http://www.sample.org/openEHRweight-
birth#Body_weight_at_birth"/>
        <similarToOutput rdf:resource= "http://www.sample.org/hl7weight-birth.owl#birth_weight"/>
        <operationName>SimilarTo_1</operationName>
        <Condition>(and (rdf:type ?x
                http://www.sample.org/openEHRweight-birth#Body_weight_at_birth)
                (state ?x ?y))
        </Condition>
...
</SimilarTo>
```

defined to match these paths to one another. These path expressions are stated as parameters in the ObjectPropertyTransform pattern in KIF format. For example, the path between the *body weight at birth* and PARTY REF can be represented in the source ontology through the following path: (rdf:type ?x Body weight at birth) (other participations ?x ?y) (rdf:type ?y List) (items ?y ?z) (rdf:type ?z PARTICIPATION) (performer ?z ?k) (rdf:type ?k PARTY REF).

This path corresponds to the following path in the target ontology: (rdf:type ?x birth weight) (hasParticipation ?x ?y) (rdf:type ?y Participation) (hasRole ?y ?z) (rdf:type ?z Patient) (player ?z ?k) (rdf:type ?k Person).

Through such patterns, the OWLmt constructs the specified paths among the instances of the target ontology in the execution step, based on the paths defined among the instances of the source ontology.

- Matching source ontology Data Properties to target ontology Data Properties. Through the DatatypePropertyTransform pattern, the data type properties of an instance in the source ontology are mapped to corresponding target ontology instance data type properties. OWLmt supports a set of basic XPath (XQuery 1.0 and XPath 2.0, 2004) functions and operators such as concat, split, and substring. In some cases, there is a further need for a programmatic approach in order to specify complex functions (e.g., need to use if-then-else, switch-case, or for-next). Therefore, we have introduced JavaScript support to OWLmt. By specifying the JavaScript to be used in the DatatypePropertyTransform pattern, the complex functions (enriched by the Java SDK libraries) can be applied in the value transformations.

As an example, the OWL representations of the archetypes (see Figures 3 and 4) include data type properties that involve the same kind of data. For in-

Figure 7. Mapping object properties

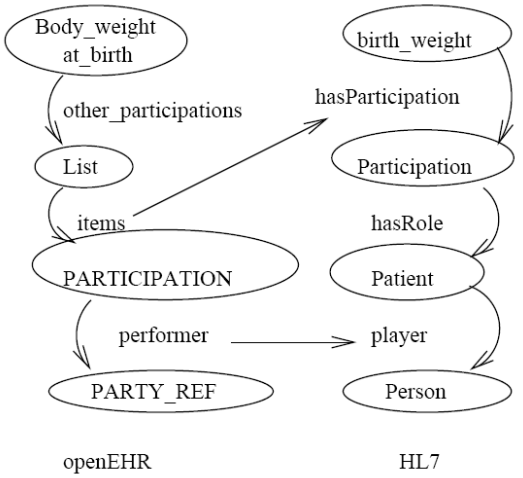


Figure 8. An example JavaScript

```
function copy_code(openEHR_code)
{
        if(openEHR_code.equals("Naked"))
        return "8351-9";
        else if(openEHR_code.equals(Dressed")
        return "8350-1";
}
```

stance, *units* and *magnitude* data type properties in openEHR archetype correspond to the *value* data type property in the HL7 archetype. To map the values stored in *units* and *magnitude* data type properties to the *value* data type property, we state a Datatype PropertyTransform pattern. This pattern takes the paths of the data type properties *units* and *magnitude* in KIF format as input parameters and relates them to the *value* data type property. The basic concat operation is sufficient to concatenate the values stored in the units and magnitude and to assign the result to the value through the mapping engine.

There is also a relation between the *code* property, which states the clothing status of a patient in openEHR archetype, and the *code* property of the *birth weight* in the HL7 archetype. Based on the value of the *code* (e.g., naked or dressed) in openEHR archetype instance, the *code* data type property in HL7 archetype has either the LOINC value 8351-9 or the LOINC value 8350-1. To achieve such a mapping, the JavaScript code (see Figure 8) can be used in the DatatypePropertyTransform pattern.

Once the mapping between two ontologies is specified by using the Mapping GUI, it can be serialized as a Mapping Definition in order to be used in the execution step as presented in the *Transforming the Archetypes Instances* subsection. The Mapping Definition itself is an OWL document whose structure is specified through Mapping Schema. In the mapping definition, the patterns are used to define the mappings among the classes and properties of the source and target ontology. The patterns are also specified as OWL class instances in the Mapping Specification. As an example, SimilarTo pattern is shown in Figure 6.

However, the use of OWL as a mapping definition language has some shortcomings, as stated in Brujin and Polleres (2004). One of the shortcomings of using OWL as a mapping definition language is its tight coupling between the source and the target ontologies. A mapping definition needs to import other related (source and target) ontologies with owl:import. This results in a tight coupling between ontologies, which is undesirable, because it makes one ontology dependent on another in the sense that axioms and definitions in one ontology use classes and properties from the other ontology. This can result in

such things as the necessity to use the other externally specified ontology in order to perform certain local reasoning tasks (Brujin & Polleres, 2004).

Therefore, rather than using pure OWL, we specify queries in our mapping definition in OWL-QL KIF syntax which are then executed by the mapping engine. Furthermore, the value transformations also should be expressed in the Mapping Definition. We specify the value transformations as JavaScript strings of the DatatypePropertyTransform pattern.

Transforming the Archetype Instances

Consider the example presented in Figure 2. Hospital B, using the archetype instances based on HL7, wishes to invoke the Web Service provided by Hospital A in order to receive the BirthWeight information for a patient. Through the Web Service Invocation Interface provided by the ARTEMIS peer, Hospital B provides the Web Service input as PatientInfo archetype instance based on HL7 RIM and wishes to receive the result as a BirthWeight archetype instance again based on HL7 RIM. Note that Hospital A has declared to the mediator that its Web Service exchanges are based on OpenEHR archetypes. When the mediator invokes this Web Service on behalf of Hospital B, the invocation tool in the mediator consults to the OWLmt Mapping Engine for transforming the archetype instances from source ontology to the target ontology. In this section, we detail how this instance transformation is achieved through the OWLmt mapping engine.

The OWLmt mapping engine creates the target archetype instances in OWL, using the mapping patterns in the Mapping Definition and the instances of the source archetype. It uses OWL Query Language (OWL-QL) (Fikes, Hayes, & Horrocks, 2003) to retrieve required data from the source ontology instances. While executing the class and property mapping patterns, the query strings defined through the mapping GUI are sent to the OWL-QL engine with the URL of the source ontology instances. The query engine executes the query strings and returns the query results.

During this process, OWL-QL uses the reasoning capabilities of Java Theorem Prover (JTP) (2005) to infer new facts from the source ontology and use them in order to construct the target ontology instance. To illustrate this, consider the archetypes introduced in the *Example OpenEHR and HL7 Archetypes in OWL* subsection. The range of the *state* object property of *body weight at birth* class in openEHR archetype is the state structure, which is a subclass of *list* and involves a restriction. *State structure* in OWL is depicted in Figure 9.

The *items* object property of *state structure* is involved in an owl:allValuesFrom restriction. Its range is stated to be the *clothing* class, as depicted in Figure 9.

The mapping engine uses the information in the source ontology to infer new knowledge at instance level. This new knowledge lets OWLmt obtain more accurate query results in the execution step. In the running example, the following rules are used to derive the fact that InferredInstance is an instance of the clothing: (rdf:type

Figure 9. The state_structure class

```
<owl:Class rdf:ID="state_structure">
        <rdfs:subClassOf rdf:resource= "openEHR:List"/>
        <rdfs:subClassOf>
                <owl:Restriction rdf:ID="RestrictionOnitems">
                        <owl:allValuesFrom>
                                <owl:Class rdf:ID="Clothing">
                                        <rdfs:subClassOf rdf:resource= "openEHR:ELEMENT"/>
....
                                </owl:Class>
                        </owl:allValuesFrom>
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:resource= "openEHR:items"/>
                        </owl:onProperty>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>``birth\_weight"
                        <owl:onProperty>
                                <owl:ObjectProperty rdf:resource="openEHR:items"/>
                        </owl:onProperty>
                        <owl:minCardinality rdf:datatype= http://www.w3.org/2001/XMLSchema#int">
                        0</owl:minCardinality>
                        <owl:maxCardinality rdf:datatype=http://www.w3.org/2001/XMLSchema#int">
                        1</owl:maxCardinality>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

MyStateStructure state_structure) (rdfs:subClassOf state_structure RestrictionOnitems) -> (rdf:type MyStateStructure RestrictionOnitems).

This rule derives the fact that MyStateStructure has rdf:type of RestrictionOnitems. With the derivation of this fact, all the predicates of the following rule become true: (owl:onProperty RestrictionOnitems items) (owl:all ValuesFrom RestrictionOnitemsClothing) (rdf:type MyStateStructureRestriction Onitems) (items MyStateStructure MyClothingType) -> (rdf:type MyClothingType Clothing).

According to the data obtained by querying the source ontology instance, the OWLmt mapping engine executes the conceptual mapping patterns to create the corresponding instances in the target ontology. The conditions specified for each conceptual mapping pattern also are applied to ensure the accuracy in the mapping process. In this step, the instances that do not satisfy a particular condition in the pattern are discarded.

Figure 10. The openEHR archetype instance

```
<state_structure rdf:ID="MyStateStructure">
        <openEHR:items rdf:resource="#InferredInstance"/>
</state_structure>
```

Figure 11. Source instance

```
<Body_weight_at_birth rdf:ID="Instance1">
        <openEHR:state>
                <state_structure rdf:ID="Instance2">
                        <openEHR:items rdf:resource="#Instance3"/>
                </state_structure>
        </openEHR:state>
...
</Body_weight_at_birth>
<Clothing rdf:ID="Instance3">
        <openEHR:value>
                <openEHR:CODED_TEXT rdf:ID="Instance4">
                        <openEHR:code>Naked</openEHR:code>
                </openEHR:CODED_TEXT>
        </openEHR:value>
</Clothing>
```

As an example, when the mapping engine executes the SimilarTo pattern (see Figure 6), the *body weight at birth* instance is obtained (see Figure 11) as a result of the query of the source instance. The mapping engine then creates the corresponding *birth weight* instance (see Figure 12) in the target ontology.

After the creation of the class instances in the target ontology, the property mapping patterns are applied to create the object and datatype properties for the instances in the target ontology. During the mapping of the datatype properties, the value transformations specified in the corresponding patterns are applied. OWLmt mapping engine uses JavaScript in order to transform the values from the source ontology to the target ontology programmatically. In order to achieve this, it sends the JavaScript specified in the corresponding property mapping pattern to the JavaScript Interpreter (RHINO, 2005) with the data obtained from the source ontology instance. The result from the execution of the JavaScript is set as the value of the datatype property in the target ontology. For example, as a result of executing the DatatypePropertyTransform pattern, which includes the JavaScript (see Figure 8), the *code* datatype property with LOINC code of 8351-9 (see Figure 12) is created in the target instance according to the *naked* value indicated in openEHR instance, as depicted in Figure 11.

As a result of these steps, the archetype instance based on the source RIM is transformed to the archetype instance, based on the target RIM, providing the interoperability of the Web Services exchanging such messages.

Figure 12. Target instance

```
<birth_weight rdf:ID="TInstance1">
        <HL7:code>8351-9</HL7:code>
...
</birth_weight>
```

CONCLUSION AND FUTURE WORK

Web Services have the capacity to bring many advantages to the health care domain, such as seamless integration of disparate health care applications conforming to different and, at times, competing standards. Also, Web Services will extend the life of the existing health care software by exposing previously proprietary functions as Web Services.

To the best of our knowledge, the ARTEMIS project is the first initiative to use semantically enriched Web Services in the health care domain. In fact, only very recently did Web Services start to appear in the medical domain. An important industry initiative to use Web Services is Integrating the Health care Enterprise (IHE) (IHE IT Infrastructure Integration Profiles, 2003). IHE has defined a few basic Web Services, such as Retrieve Information for Display Integration Profile (RID). Yet, since IHE does not address semantic issues, in order to use the IHE Web Services, it is necessary to conform to their exact specification by calling the Web Services with the names they have specified and by providing the messages as instructed in its specification.

However, given the complexity of the health care domain and the proliferation of standards and the terminologies to represent the same data, semantic annotation of the Web Service messages is essential.

In this article, we describe how interoperability among different health care systems conforming to different EHR standards can be achieved by semantically annotating the Web Service messages through archetypes. An archetype is a set of constraints on the generic EHR reference information model, which ensures that clinical concepts are correctly represented without actually storing them, since there are very many (more than 300,000) clinical concepts. The semantic differences among the archetypes are then handled through an OWL mapping tool that is developed.

As a future work, we plan to semantically annotate the IHE Web Services that currently are being integrated into the ARTEMIS infrastructure. How IHE Web Services are integrated to the ARTEMIS architecture is described in Aden and Eichelberg (2005).

ACKNOWLEDGMENT

This work is supported by the European Commission through IST-1-002103-STP ARTEMIS project and in part by the Scientific and Technical Research Council of Turkey (TÜB0TAK), Project No: EEEAG 104E013

REFERENCES

Aden, T., & Eichelberg, M. (2005). *Cross-enterprise search and access to clinical information based on IHE retrieve information for display*. Paper presented at EuroPACS-MIR 2005.

Archetype definition language 1.2 draft. (2003). Australia: Ocean Informatics, the OpenEHR Foundation.

ARTEMIS Consortium. (2004). The ARTEMIS Project. Retrieved from http://www.srdc.metu.edu.tr/webpage/projects/artemis

Baader, F., Horrocks, I., & Sattler, U. (2004). *Handbook on ontologies*. Description Logics.

Beale, T., & Heard, S. (2003). *Archetype definitions and principles* (Revision 0.5). Australia: Ocean Informatics, the OpenEHR Foundation.

Brujin, J., & Polleres, A. (2004). *Towards an ontology mapping specification language for the Semantic Web* (DERI Tech. Rep.).

Complete Blood Count Archetype ADL Definition. (2005). Retrieved from http://my.openehr.org/wsvn/knowledge/archetypes/dev/adl/openehr/ehr/entry/observation/openEHR-EHR-OBSERVATION.cbc.v1.adl?op=file&rev=0&sc=0

CEN TC/251 prEN 13606-1, Health informatics electronic health record communication. Part 1: Reference model, draft for CEN enquiry. (2004). Brussels, Belgium: CEN/TC 251 Health Informatics, European Committee for Standardization.

Dogac, A., et al. (in press). ARTEMIS: Deploying semantically enriched Web Services in the health care domain. *Information Systems Journal*.

Eichelberg, M., Aden, T., Dogac, A., & Laleci, G. B. (2005). *A survey and analysis of electronic health care record standards*. Unpublished manuscript.

Fikes, R., Hayes, P., & Horrocks, I. (2003). *OWL-QL: A language for deductive query answering on the Semantic Web* (Tech. Rep.). Stanford, CA: Stanford University, Knowledge Systems Laboratory.

Heard, S., Beale, T., Mori, A. R., & Pishec, O. (2003). *Templates and archetypes: How do we know what we are talking about* (Tech. Rep.).

HL7 Body Weight at Birth Archetype OWL Definition. (2005). Retrieved from http://www.srdc.metu.edu.tr/~veli/hl7_Weight_Birth_Archetype.owl

HL7 Clinical Document Architecture, Release 2.0 (HL7 v3 Standard). (2004). Ann Arbor, MI: Health Level Seven.

HL7 Template and Archetype Architecture Version 3.0 (Tech. Rep.). (2003). Health Level Seven Template Special Interest Group.

IHE IT Infrastructure Integration Profiles. (2003). *The Key to Integrated Systems: Integration Profiles*. Oak Brook, IL: Integrating the Health Care Enterprise.

ISO TC/215. (2003). *Health Informatics — Requirements for an Electronic Health Record Architectur*e (ISO Technical Specification 18308). International Organization for Standardization, Health Informatics.

JTP. (2005). *Java Theorem Prover*. Retrieved from http://www.ksl.stanford.edu/software/JTP/

Kashyap, V., & Sheth, A. P. (1996). Semantic and schematic similarities between database objects: A context-based approach. *VLDB Journal, 5*(4), 276-304.

KIF. (2005). *Knowldge interchange format*. Retrieved from http://logic.stanford.edu/kif/kif.html

LOINC. (2005). Logical observation identifiers names and codes (Version 2.15). Indianapolis, IN: Regenstrief Institute.

openEHR Architecture Specifications. (2005). Retrieved from http://www.openehr.org/getting_started/t_openehr_primer.htm

openEHR Body Weight at Birth Archetype OWL Definition. (2005). Retrieved from http://www.srdc.metu.edu.tr/~veli/openEHRweight-birth.owl

openEHR Community. (2005). Retrieved from http://www.openehr.org/

OWL. (2004). Web ontology language. In *Proceedings of the World Wide Web Consortium*.

OWLmt. (2005). *OWL mapping tool*. Retrieved from http://www.srdc.metu.edu.tr/artemis/owlmt/

RHINO. (2005). *JavaScript for Java*. Retrieved from http://www.mozilla.org/rhino/

Sheth, A. P., & Larsen, J. (1990). Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Surveys: Special Issue on Heterogeneous Databases, 22*(3), 183-236.

SNOMED Clinical Terms. (2005). Retrieved from http://www.snomed.org/snomedct/index.html

SOAP. (2003). Simple object access protocol (Version 1.2). In *Proceedings of the World Wide Web Consortium*.

WSDL. (2005). Web Service description language (Version 2.0). Part 0: Primer (W3C Working Draft). In *Proceedings of the World Wide Web Consortium*.

XQuery 1.0 and XPath 2.0. (2004). *Functions and operators* (W3C Working Draft). Retrieved from http://www.w3.org/TR/xpath-functions/

Asuman Dogac is a full professor at the Computer Engineering Dept., Middle East Technical University (METU), Ankara, Turkey and the founding director of the Software Research and Development Center (SRDC), METU. Her research interests include health care informatics, Web Services, and semantic interoperability.

Gokce B. Laleci is a PhD candidate at METU, Computer Engineering Dept., and a senior researcher at SRDC. Her research interests include Semantic Web, Web Service technology, and health care informatics.

Veli Bicer is an MS student at METU, Computer Engineering Dept., and a senior researcher at SRDC. His research interests include semantic interoperability, Web Service technology, and health care informatics.

Ozgur Kilic recevied a BS in computer engineering in 1998 from the Bilkent University. He recevied an MS in computer engineering in 2001 from the Middle East Technical University. He is currently a PhD student in computer engineering at the Middle East Technical University.