Exploiting ebXML Registry Semantic Constructs for Handling Archetype Metadata in Healthcare Informatics *

Asuman Dogac, Gokce B. Laleci, Yildiray Kabak, Seda Unal
Middle East Technical University, Turkey
Thomas Beale, Sam Heard, Ocean Informatics, Australia
Peter Elkin, Mayo Clinic, USA
Farrukh Najmi, Sun Micro Systems, USA
Carl Mattocks, OASIS ebXML Registry SCM SC, USA
David Webber, OASIS CAM TC, USA

ABSTRACT

Using archetypes is a promising approach in providing semantic interoperability among healthcare systems. To realize archetype based interoperability, the healthcare systems need to discover the existing archetypes based on their semantics; annotate their archetypes with ontologies; compose templates from archetypes and retrieve corresponding data from the underlying medical information systems.

In this paper, we describe how ebXML Registry semantic constructs can be used for annotating, storing, discovering and retrieving archetypes. For semantic annotation of archetypes, we present an example archetype metadata ontology and describe the techniques to access archetype semantics through ebXML query facilities. We present a GUI query facility and describe how the stored procedures we introduce, move the semantic support beyond what is currently available in ebXML registries.

We also address how archetype data can be retrieved from clinical information systems by using ebXML Web services. A comparison of Web service technology with ebXML messaging system is provided to justify the reasons for using Web services.

1. INTRODUCTION

Most of the health information systems today are proprietary and often only serve one specific department within a healthcare institute. To make the matters worse, a patient's health information may be spread out over a number of different institutes which do not interoperate. This makes it very difficult for clinicians to capture a complete clinical history of a patient.

A number of standardization efforts are progressing to provide the interoperability of healthcare systems such as CEN TC 251 prEN13606 [9], openEHR [36] and HL7 Version 3 [27]. However, exchanging machine processable electronic healthcare records have not yet been achieved. For example, although HL7 Version 2 Messaging Standard is the most widely implemented standard for healthcare information in the world today, being HL7 Version 2 compliant does not imply direct interoperability between healthcare systems. Version 2 messages contain many optional data fields. This optionality provides great flexibility, but necessitates detailed bilateral agreements among the healthcare systems to achieve interoperability. To remedy this problem, HL7 [24] has developed Version 3 which is based on an object-oriented data model, called Reference Information Model (RIM) [26].

Yet, given the large number of standards in the healthcare informatics domain, conforming to a single standard does not solve the interoperability problem.

In fact, the full shareability 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). One of the successful examples of functional interoperability is electronic mail: an email can be sent and received by any platform due to the well established standards for the transport binding which is SMTP running on TCP/IP and the message format which is ASCII or HTML. That is, SMTP and the fixed message format makes it possible for two systems exchange emails. However, note that the message content is only for human consumption.

- *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 [30]. In other words, semantic interoperability requires the semantics of data to be defined through formally

*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 (TÜBİTAK), Project No: EEEAG 104E013

International Journal of Metadata, Semantics and Ontologies (IJMSO)

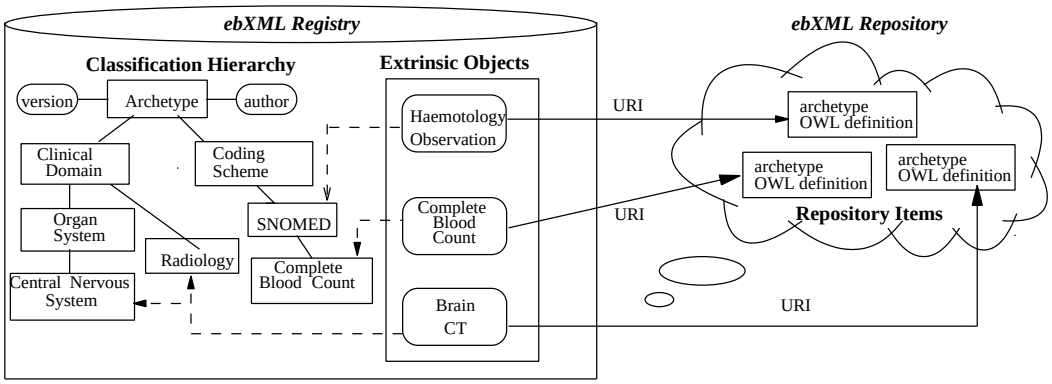

Figure 1: Handling Archetype Semantics in ebXML Registries

defined domain specific concepts. Semantic interoperability in the healthcare domain can be achieved by conforming to a single healthcare information standard, such as HL7 Version 3. However, it is not realistic to expect all the healthcare institutes to conform to a single standard. A promising approach in providing interoperability among different standards in the healthcare domain is the archetypes [6]. An "archetype" is a syntactically and semantically structured aggregation of vocabulary or other data that is the basic unit of clinical information [19]. When healthcare systems start exchanging information with well defined syntax and semantics as proposed by "archetypes", semantic interoperability among them will become a reality.

An important aspect of archetype based interoperability is providing the ability to the healthcare institutes and systems to share archetypes and their metadata among them. In this paper, we describe how ebXML registries can be used for handling templates and archetypes by using the semantic constructs of the ebXML registry. This work is carried out within the scope of Artemis project [3] which aims to provide interoperability in the healthcare domain through semantically enriched Web services.

Electronic Business XML (ebXML) [13] is a standard from OASIS[35] and United Nations Centre for Trade Facilitation and Electronic Business, UN/CEFACT [42]. ebXML specifies an infrastructure that allows enterprises to find each other's services, products, business processes and documents in a standard way and thus helps to facilitate conducting electronic business. One of the important characteristics of ebXML compliant registries is that they provide standard mechanisms both to define and to associate metadata with registry entries.

In order to achieve archetype based interoperability among healthcare institutes through ebXML registries, we propose the following phases:

- *Semantically annotating the archetypes*: For healthcare systems to exchange archetypes, they need to discover the archetypes of the institutes they wish to communicate with. This discovery must be based on the semantics of archetypes such as the purpose of the archetype, the clinical domains it is associated with, the types of clinical documents it is used in as constituents, where they fit into the slots of other archetypes as well as authorship, and the version of the archetype. Since ebXML registry allows metadata to be associated with the registry entries and provides mechanisms for semantic based discovery of registry entries, it provides a convenient medium for handling archetypes.

  As the first phase of archetype based interoperability, we show how ebXML registry semantic constructs can be used to annotate archetypes. For this purpose, we present an example archetype metadata ontology. It should be noted that our purpose is not to propose an archetype ontology but rather to show how it can be exploited once it is specified by standard bodies.

- *Retrieving archetypes from the registry through ebXML query facilities*: The archetypes need to be discovered in the registry according to their semantics. This semantic information serves the purpose of how or where the archetype can be used. For example, the metadata can be queried to find out the archetypes associated with a given clinical domain, or the coding schemes they are referring to, or the authorship.

  We show how the semantic information used in annotating the archetypes in an ebXML registry can be queried through ebXML standard query mechanisms such as Filter Query or SQL-92 query. In order to facilitate access to the system by novice users, we present ready to be used stored queries for discovering archetypes according to their metadata.

  We then describe how Web Ontology Language (OWL) can be used in ebXML registries to enhance the archetype semantics. Although the ebXML semantic mechanisms are useful as they stand, currently, semantics is becoming a much broader issue than it used to be through the use of ontologies and standard ontology definition languages [10, 20, 34]. An important standard ontology language is W3C's Web Ontology Language (OWL) [43]. There are several opportunities to be gained in extending ebXML semantic mechanisms to standard ontology languages, such as OWL. In this way, it becomes possible to exploit the richer semantic constructs of OWL as well as its reasoning capabilities.

  In our previous work, we describe how ebXML reg-

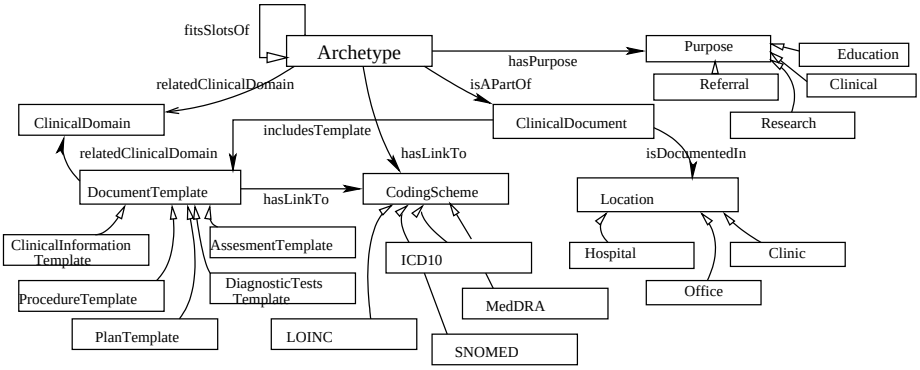

Figure 2: An Archetype Metadata Ontology

istries can be enriched with OWL semantics [11] and how this additional semantics can be processed with stored queries [12]. In this paper, we show how archetype and template semantics can benefit from OWL aware ebXML registries.

- *Retrieving archetype data from medical information systems*: After discovering archetypes by exploiting their metadata, it is also necessary to retrieve the associated information from the medical information systems conforming to the templates and archetypes.

  Although the templates and archetypes semantically and structurally define the data to be retrieved through formally defined domain concepts and hence provide for the semantic interoperability, we still need to provide the functional level interoperability. As mentioned previously functional interoperability involves agreeing on the common network protocols, the transport bindings and the message formats. Although ebXML messaging provides for this, we choose to use another functional interoperability standard, namely, Web services which is also supported by ebXML registries.

The paper is organized as follows: In Section 2, the objectives of archetypes and templates are summarized and how semantic interoperability can be achieved through archetypes is described. In Section 3, we show how archetype semantics can be handled in ebXML registries. Here we also describe enhancing ebXML registries with Web Ontology Language (OWL) semantics and explain how archetypes semantics can benefit from OWL aware ebXML registries. In Section 4, we present an example archetype metadata ontology to annotate archetypes and templates. Section 5 describes the techniques to access archetype semantics through ebXML query facilities. In this section we introduce a GUI query facility and show how stored procedures move the semantic support beyond what is currently available in ebXML registries. Section 6 addresses retrieving data from clinical information systems conforming to archetypes. In this section, we compare Web service technology with ebXML messaging system and provide the reasons for choosing Web services in accessing the underlying medical information systems. In Section 7, we describe how archetypes can be composed through Web service composition techniques. Finally, Section 8 concludes the paper.

2. WHY DO WE NEED TEMPLATES AND ARCHETYPES?

Healthcare is one of the few domains where sharing information is the norm, rather than the exception [21]. Archetypes and templates aim to solve the semantic interoperability problem in the healthcare domain by allowing modeling of domain concepts based on reference models, independent from the information systems. Archetypes have been adopted by a number of standards such as openEHR [36], CEN TC/251 [9] and is considered to be used by HL7 [24] as a basis for its templates specification.

An archetype is a reusable, formal expression of a distinct, domain-level concept such as "blood pressure", "physical examination", or "laboratory results", expressed in the form of constraints on data whose instances conform to some reference model [6]. The reference model refers to any model such as CEN TC 251 prEN13606 [9], openEHR [36], or the HL7 CDA schema [25]. The key feature of the archetype approach to computing is a complete separation of information models (such as object models of software, models of database schemas) from domain models [6].

A formal language for expressing archetypes has been introduced which is called Archetype Definition Language (ADL) [2]. In Figure 3, a part of "Complete Blood Count" archetype definition is presented in ADL. The complete ADL definition can be found in [8]. 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.

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.

Note that in order to address the interoperability problem through archetypes, a generic approach based on a "harmonised" information model needs to be adopted [7]. Alternatively, the reference models of these standardization bodies (openEHR, CEN TC/251, HL7) can be represented

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

Figure 3: The ADL definition of "Complete Blood Count" Archetype

through an ontology language like OWL. Then by defining the archetypes constraining these reference models also in OWL, and by providing the mapping between these reference models through ontology mapping, the interoperability of the archetype instances can be achieved automatically.

3. WHAT DOES EBXML REGISTRY PROVIDE FOR ARCHETYPES?

For healthcare systems to exchange information in an interoperable manner, they need to discover the archetypes and templates and their associated semantics. ebXML Registry, through its semantic constructs, provides an efficient medium to annotate, store, discover and reuse of archetypes.

In the following sections, we first briefly present ebXML Registry architecture and then demostrate how basic semantic constructs of ebXML Registry can be used for this purpose. Then, we present how ebXML registries can be enhanced with OWL semantics and how this knowledge can be exploited for handling arhetypes.

3.1 ebXML Specification

ebXML facilitates electronic business as follows:

- In order for enterprises to conduct electronic business with each other, they must first discover each other and the products and services they have to offer. ebXML provides a registry where such information can be published and discovered.

- An enterprise needs to determine which business processes and documents are necessary to communicate with a potential partner. *A Business Process Specification Schema (BPSS)* in ebXML, provides the definition of an XML document that describes how an organization conducts its business.

- After this phase, the enterprises need to determine how to exchange information. The Collaboration Protocol Agreement (CPA) specifies the details of how two organizations have agreed to conduct electronic business.

A registry can be established by an industry group or standards organization. A repository is a location (or a set of distributed locations) where a document pointed at by the registry reside and can be retrieved by conventional means (e.g., http or ftp).

It should be noted that within the scope of this paper, we address how to handle the archetypes and templates and not the healthcare business processes. There are other healthcare informatics initiatives complementary to our work such as IHE IT Infrastructure Integration Profiles [22] which define some of the business processes in the healthcare domain and IHE Cross-Enterprise Clinical Documents Sharing (XDS) [23] which specify how to use ebXML registries for healthcare document sharing. Note however that semantic issues are not yet addressed by these initiatives.

3.2 ebXML Registry Architecture and Information Model

ebXML registry provides a persistent store for registry content. The current registry implementations store registry data in relational databases. ebXML Registry Services Specification defines a set of Registry Service interfaces which provide access to registry content. There are a set of methods that must be supported by each interface. A registry client program utilizes the services of the registry by invoking methods on one of these interfaces. The Query Manager component also uses these methods to construct the objects by obtaining the required data from the relational database through SQL queries. In other words, when a client submits a request to the registry, registry objects are constructed by retrieving the related information from the database through SQL queries and are served to the user through the methods of these objects [18].

An ebXML registry [17] allows to define semantics basically through two mechanisms: first, it allows properties of registry objects to be defined through "slots" and, secondly, metadata can be stored in the registry through a "classification" mechanism. This information can then be used to discover the registry objects by exploiting the ebXML query mechanisms.

3.3 Exploiting ebXML Registries for Semantically Annotating Archetypes

As previously noted, ebXML Registry semantic constructs can be used to annotate registry objects. For example, as shown in Figure 1, the archetype "HaemotologyObservation" can be stored as an Extrinsic Object (which is also a Registry Object) whose link points to the repository item that holds the actual archetype definition. The metadata of the archetype, on the other hand, can be classified with as many *ClassificationNodes* as needed to describe its semantics. For example, in Figure 1, "HaemotologyObservation Archetype" Extrinsic Object is classified with *SNOMED* [38] clinical coding terms. Therefore when a user wishes

Table 1: Predefined Association Types in ebXML Registries

Name	*Description*
RelatedTo	Defines that source RegistryObject is related to target RegistryObject.
HasMember	Defines that the source RegistryPackage object has the target RegistryObject object as a member.
ExternallyLinks	Defines that the source ExternalLink object externally links the target RegistryObject object.
Contains	Defines that source RegistryObject contains the target RegistryObject.
EquivalentTo	Defines that source RegistryObject is equivalent to the target RegistryObject.
Extends	Defines that source RegistryObject inherits from or specializes the target RegistryObject.
Implements	Defines that source RegistryObject implements the functionality defined by the target RegistryObject.
InstanceOf	Defines that source RegistryObject is an Instance of target RegistryObject.
Supersedes	Defines that the source RegistryObject supersedes the target RegistryObject.
Uses	Defines that the source RegistryObject uses the target RegistryObject in some manner.
Replaces	Defines that the source RegistryObject replaces the target RegistryObject in some manner.
SubmitterOf	Defines that the source Organization is the submitter of the target RegistryObject.
ResponsibleFor	Defines that the source Organization is responsible for the ongoing maintainence of the target RegistryObject.
OffersService	Defines that the source Organization object offers the target Service object as a service.

to find all the archetypes in the registry annotated with SNOMED Clinical Coding terms; issuing an ebXML query will suffice. In other words, by relating a *ClassificationNode* (such as *SNOMED*) with a RegistryObject in ebXML (such as "HaemotologyObservation"), we make this object an implicit member of the *SNOMED* node and hence the object inherits the semantics associated with that node. The ebXML query to find all the archetypes coded with *SNOMED*, first finds the *SNOMED ClassificationNode* and the links from this node. These links are then used to retrieve the related ExtrinsicObjects and by issuing a "getContentQuery", the content of the archetypes are retrieved from the Repository. In fact such queries can be automatically generated and issued through Graphical User Interfaces as described in Section 5.

Through this example, we describe the simplest and most basic way of associating semantics with ebXML registry objects. ebXML classification hierarchies allow more complex semantics to be defined and queried both through the "slot" mechanism and through the predefined associations between registry objects. For example, through "slot" mechanism, it is possible to define the properties of classes (ClassificationNodes). "Slot" instances provide a dynamic way to add arbitrary attributes to "RegistryObject" instances. For the example shown in Figure 1, the archetype properties such as "version" and "authorship" are defined by using slots.

Furthermore, through predefined associations, it is possible to associate ClassificationNodes. There are a number of predefined *Association Types* that a registry must support to be ebXML compliant [17] as shown in Table 1.

Although ebXML semantic mechanisms are useful as they stand, currently, semantics is becoming a much broader issue than it used to be and the trend is to use ontologies [10, 20, 34]. One of the driving forces for ontologies is the Semantic Web initiative [5]. As a part of this initiative, W3C's Web Ontology Working Group defined Web Ontology Language (OWL) [43].

There are several opportunities to be gained in extending ebXML semantics mechanisms to make them OWL aware. In this way, it will become possible to represent ontologies defined in OWL in the ebXML registry and to exploit the richer semantic constructs of OWL as well as its reasoning capabilities.

In our previous work, we describe how ebXML registries can be enriched with OWL semantics [11] and how this additional semantics can be processed with stored queries [12]. Here, for the sake of completeness, we provide a brief summary.

Being OWL aware entails the following enhancements to the ebXML registry:

- *Representing OWL constructs through ebXML constructs*: ebXML provides a classification hierarchy made up of classification nodes and predefined type of associations between the registry objects. We represent *OWL* constructs by using combinations of these constructs and define additional types of associations when necessary. For example, OWL classes can be represented through "ClassificationNodes" and RDF properties that are used in OWL can be treated as "Associations". An "Association" instance represents an association between a "source RegistryObject" and a "target RegistryObject". Hence the target object of "rdfs:domain" property can be mapped to a "source RegistryObject" and the target object of "rdfs:range" can be mapped to a "target RegistryObject". "OWL ObjectProperty","DataTypeProperty" and "TransitiveProperty" are defined by introducing new association types such as "objectProperty". ebXML specification allows additional associations to be defined.

  When it comes to mapping OWL class hierarchies to ebXML class hierarchies, OWL relies on RDF Schema for building class hierarchies through the use of "rdfs:subClassOf" property and allows multiple inheritance. An ebXML Class hierarchy has a tree structure, and therefore is not readily available to express multiple inheritance, that is, there is a need for additional mechanisms to express multiple inheritance. We define a "subClassOf" property as an association for this purpose.

  As another example, in ebXML, the predefined "EquivalentTo" association (Table 1) expresses the fact that the source registry object is equivalent to target registry object. Therefore, "EquivalentTo" association is used to express "owl:equivalentClass", "owl:equivalentProperty" and "owl:sameAs" properties since classes, properties and instances are all

ebXML registry objects. The details of how the rest of the OWL constructs are represented in ebXML registries is available from [12].

- *Automatically generating ebXML constructs from the OWL descriptions and storing the resulting constructs into the ebXML registry*: We developed a tool to create an ebXML Classification Hierarchy from a given OWL ontology automatically by using the transformations described. The OWL file is parsed using Jena [32], the classes together with their property and restrictions are identified, and the "SubmitObjectsRequest" is prepared automatically. This request is then sent to ebXML registry which in turn creates necessary classes and associations between them. For example the OWL definition of Archetype Metadadata Ontology presented in Figure 2 is parsed and a Classification Hierarchy, a part of which is presented in Figure 1, is created automatically in the ebXML registry.

- *Querying the registry for enhanced semantics*: When various constructs of OWL are represented by ebXML classification hierarchies, although some of the OWL semantics stored in an ebXML registry can be retrieved from the registry through ebXML query facilities, further processing needs to be done by the application program to make use of the enhanced semantics.

  For example, two classification nodes can be declared as equivalent classes through the "EquivalentTo" predefined "association" in the ebXML registry. To make any use of this semantics, given a query requesting instances of a class, the application program must have the necessary code to find out and retrieve also the instances of classes which are declared to be equivalent to this class.

  To relieve the users from this burden, the code to process the OWL semantics can be stored in ebXML registry architecture through predefined procedures. As an example, the stored procedure given in Figure 4 retrieves all the archetype instances of a ClassificationNode (class) as well as the archetype instances of all classes equivalent to this class.

As an example to how such a stored query might help the users, assume that SNOMED "CompleteBloodCount" term is defined to be equivalent to the 'Full Blood Count" term in another coded term list, say, MedDRA (Medical Dictionary for Regulatory Activities) [33]. Then through the stored procedure given in Figure 4, when a user wishes to retrieve the archetype instances related with "CompleteBloodCount" term, it becomes possible to automatically obtain the archetype instances that are classified with SNOMED as well as those instances classified with MEdDRA "Full Blood Count" term.

4. DEFINING ARCHETYPE AND TEMPLATE METADATA

An archetype is a reusable, formal expression of a distinct, domain-level concept such as "blood pressure", "physical examination", "laborotory results", expressed in the form of constraints on data whose instances conform to some class model, known as a reference model [6]. An OpenEHR template on the other hand is a directly, locally usable data

```
CREATE PROCEDURE findEquivalentInstances($className)
BEGIN
SELECT N.value FROM ExtrinsicObject EO, Name_ N
    WHERE EO.id IN (
    SELECT classifiedObject
    FROM Classification
    WHERE classificationNode IN (
      SELECT id
      FROM ClassificationNode
      WHERE id IN (
          SELECT parent
          FROM name_
          WHERE value LIKE $className
          )
      UNION
      SELECT A.targetObject
      FROM Association A, Name_ N, ClassificationNode C
      WHERE A.associationType LIKE 'EquivalentTo' AND
            C.id = N.parent AND
            N.value LIKE $className AND
            A.sourceObject = C.id
          )
    ) AND EO.id=N.parent
END;
```

Figure 4: A Stored Procedure to Find Equivalent Instances

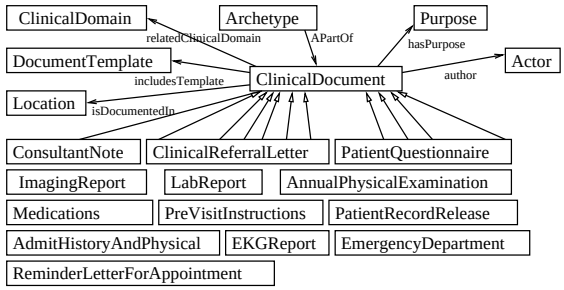

Figure 5: The Clinical Document Ontology

creation/validation artefact which is semantically a constraint/choice on archetypes, and which will often correspond to a whole form or screen [6]. In this section, we describe an example Archetype Metadata Ontology to annotate archetypes and templates. It should be noted that our purpose is not to propose an archetype ontology but rather to show how it can be exploited once it is specified by standard bodies.

4.1 An Archetype Ontology for Semantically Annotating Archetypes

We propose to classify archetypes on the basis of the following semantic information:

- The coding schemes it is referring to
- The purpose of the archetype
- The clinical domains it is associated with
- The types of clinical documents it is used in as constituents
- Other archetypes into whose slots it fits

The first level of the ontology covers this semantics as properties of the "Archetype" class as presented in Figure

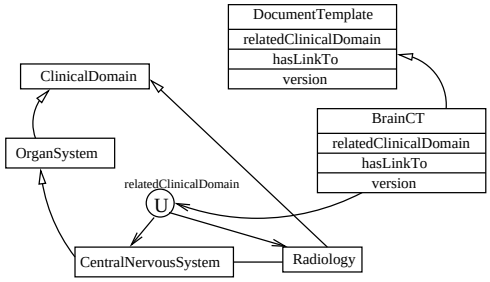

Figure 6: The BrainCT Template

2. For example, "Archetype" class has a property named "hasLinkTo" whose range is the class "CodingScheme". The "CodingScheme", "Purpose", "ClinicalDocument", "Clinical Domain" and "DocumentTemplate" classes are further detailed by defining their subclasses and properties.

The Clinical Document class in Figure 2 is organized on the basis of following axes as shown in Figure 5: author, location where the document is reported, purpose of the document, constituent templates, and clinical domains the document is related with. These metadata are represented as the properties of the "ClinicalDocument" class. A number of Clinical Document types are created as subclasses of this class.

Similarly "DocumentTemplate" class in Figure 2 is associated with "CodingScheme" class to indicate the coding schemes it is referring to, and it is associated with the "ClinicalDomain" class to indicate with which clinical domains the template is related.

As an example, "BrainCT" template can be created as a subclass of "DocumentTemplate" class, and the range of the "relatedClinicalDomain" property can be restricted to the "CentralNervousSystem" and "Radiology" classes (which are created as subclasses of "ClinicalDomain" class) as presented in Figure 6. In this way it is possible to query this template by referring to the "CentralNervousSystem" and "Radiology" domains.

Additionally generic template types "AssessmentTemplate", "ClinicalInformationTemplate", "PlanTemplate", "ProcedureTemplate" and "DiagnosticTestsTemplate" are added to the metadata ontology as subclasses of "DocumentTemplate" class as shown in Figure 2. Finally the Clinical Domain hierarchy is detailed in this metadata Ontology as depicted in Figure 7. "ClinicalDomain" class is defined to have subclasses such as "OrganSystems", "SystemicProcesses", "Temporal", "Medicine","Surgery" and "Radiology". Leaf level clinical domains are defined as subclasses of one or more of these classes. As an example, "Neurosurgery" domain is defined as a subclass of both "CentralNervousSystem" (which is a subclass of "OrganSystems" class), and "Medicine" classes.

5. HOW TO ACCESS ARCHETYPE METADATA THROUGH EBXML QUERY FACILITIES?

In this section we describe how archetypes can be represented and accessed in ebXML registries. Archetypes and their semantics are represented in an ebXML registry as follows:

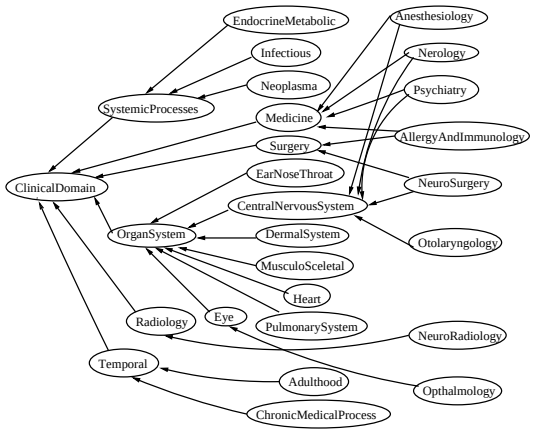

Figure 7: The Clinical Domain Ontology

- The "archetype metadata ontology" is stored in the ebXML registry to be used for querying the archetype definitions. For this purpose, a "SubmitObjectsQuery" is created by parsing the ontology and sent to the registry, which in turn creates a classification hierarchy as presented in Figure 8 (a). As described in Section 3.3, OWL classes are represented as "Registry Information Model (RIM) Classification Nodes" and OWL properties are represented as "RIM Associations".

- An "archetype" is represented in the Registry as a "RIM Extrinsic Object" as shown in Figure 8 (b). Note that "Extrinsic Objects" point to the Repository items where their contents are stored. An OWL definition of an archetype is created from its ADL (Archetype Definition Language) [2] definition and is stored in the Repository. This OWL document gives the content of the archetype describing the constraints over reference information model classes.

- In order to establish the relationship with archetype "Extrinsic Objects" and the "archetype metadata ontology", an OWL instance of the "Archetype Metadata Ontology" is created which specifies the property values of an archetype according to this ontology. As an example, Figure 9 shows an "Archetype Metadata Ontology" instance that defines the "Complete Blood Count (CBC)" archetype metadata by providing the property values in "Archetype Metadata Ontology". Through the tool developed, while storing the "ExtrinsicObject" representing, for example the CBC archetype to the registry, the "Archetype Metadata Ontology" instance in Figure 9 is parsed and the relations between the "Extrinsic Object" and the "Classification Nodes" are created automatically in the Registry through "Classification Objects". In RIM, any RegistryObject may be classified using ClassificationSchemes and ClassificationNodes through "Classification Objects". In Figure 9, the CBC archetype is related with SNOMED "Complete blood count" term with "hasLinkTo" property. This relation is represented with the "Classification Object" of ebXML

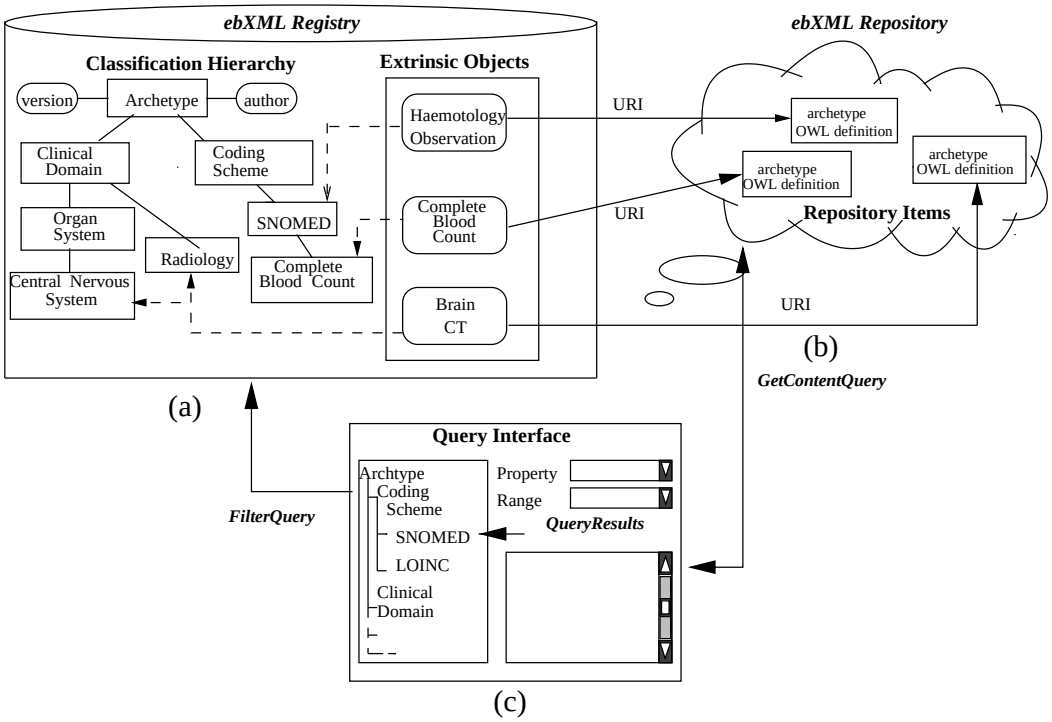

Figure 8: Querying Archetype Semantics in ebXML Registry

```
<LabReport rdf:ID="LabReport_Instance"/> <Staff
rdf:ID="GokceBanuLaleci"/> <CompleteBloodCount
 rdf:ID="Snomed_Instance"/>
<Hemotology rdf:ID="Hematology_Instance"/>
<ClinicalInformationTemplate rdf:ID=
"ClinicalInformationTemplate_Instance"/> <Archtype
rdf:ID="CBC_Archetype"> <relatedTemplate rdf:resource=
"#ClinicalInformationTemplate_Instance"/> <relatedClinicalDomain>
    <Medicine rdf:ID="Medicine_Instance"/>
</relatedClinicalDomain> <isAPartOf
rdf:resource="#LabReport_Instance"/> <relatedTemplate>
    <AssesmentTemplate rdf:ID="AssesmentTemplate_Instance"/>
</relatedTemplate> <hasLinkTo rdf:resource="#Snomed_Instance"/>
<relatedClinicalDomain rdf:resource="#Hematology_Instance"/>
<relatedTemplate>
    <DiagnosticTestsTemplate rdf:ID=
    "DiagnosticTestsTemplate_Instance"/>
</relatedTemplate> <author rdf:resource="#GokceBanuLaleci"/>
<version rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
>0.1</version>
<hasPurpose>
    <Clinical rdf:ID="Clinical_Instance"/>
</hasPurpose> <relatedClinicalDomain>
    <HemovascularSystem rdf:ID="HaemovascularSystem_Instance"/>
</relatedClinicalDomain> </Archtype>
```

Figure 9: "Complete Blood Count" Archetype Instance Definition

RIM as presented in Figure 10. Note that "CompleteBloodCount" class is a subclass of SNOMED.

After storing the archetype along with its semantic annotation to the ebXML Registry as presented in Figure 8(a), it becomes possible to query the archetypes according to their metadata. To facilitate the querying of the ebXML registry for novice users, we have developed a query tool with a GUI which on the left pane shows the classification hier-

```
<ExtrinsicObjectid="CBCarchetypeInstance" mimeType="text/xml">
        <Name>
            <LocalizedString lang="en_US" value =
            "Complete Blood Count"/>
        </Name>
</ExtrinsicObject>
<Classification
classificationNode="CompleteBloodCount"
classifiedObject="CBCarchetypeInstance">
        <Slot name = 'type'>
            <ValueList>
            <Value>hasLinkTo</Value>
            </ValueList>
        </Slot>
</Classification>
```

Figure 10: RIM Classification example

archy and allows users to formulate their queries by simply selecting the values automatically shown on the right pane according to the selections made on the left pane as shown in Figure 8(c). When a user selects values, Filter Queries are constructed for retrieving the ID's of the Extrinsic Objects, representing the Archetype Instances, according to the selected criteria. Then through these IDs ebXML "GetContent" queries are submitted for retrieving the repository items containing the archetype definition.

The ebXML semantic constructs can be used to query the archetypes through the GUI tool as follows:

- A user can search for all templates and archetypes that constrain to a particular code or coding scheme. For instance, a user can find all templates and archetypes that make reference to the SNOMED Complete Blood Count term, through the GUI tool as shown in Figure

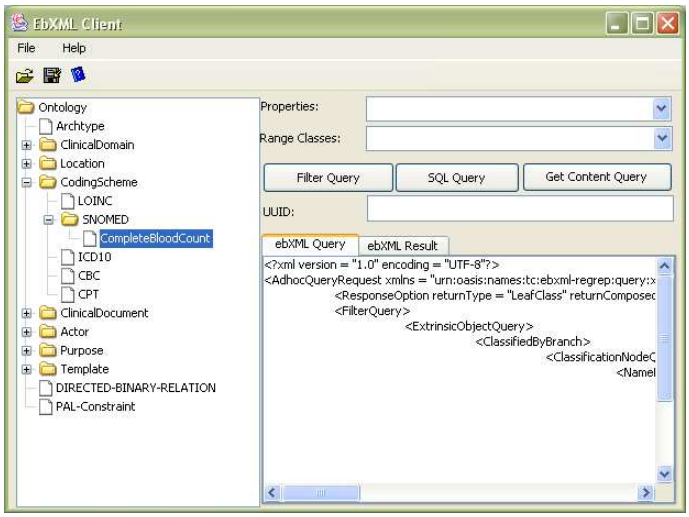

Figure 11: The GUI for retrieving the archetype Instances related with CompleteBloodCount Concept

```
<GetContentRequest>
    <rim:ObjectRefList>
        <rim:ObjectRef id="urn:uuid:368661c9-b733-4c14-96a3
        -eabbdf36ff5b"/>
    </rim:ObjectRefList>
</GetContentRequest>
```

Figure 12: ebXML GetContentRequest

```
<FilterQuery>
   <ExtrinsicObjectQuery>
    <ClassifiedByBranch>
     <ClassificationNodeQuery>
       <NameBranch>
          <LocalizedStringFilter>
           <Clause>
            <SimpleClause leftArgument = "value">
              <StringClause stringPredicate =
              "Equal" >CompleteBloodCount</StringClause>
            </SimpleClause>
           </Clause>
          </LocalizedStringFilter>
       </NameBranch>
     </ClassificationNodeQuery>
    </ClassifiedByBranch>
   </ExtrinsicObjectQuery>
</FilterQuery>
```

Figure 13: ebXML Filter Query for Extrinsic Objects classified with CompleteBloodCount node

11.

When the SNOMED term is selected on the left pane, the FilterQuery in Figure 13 is generated to retrieve the ExtrinsicObjects IDs classified with the "CompleteBloodCount" ClassificationNode. In order to get the content of these extrinsic objects, the user can press the "Get Content Query" button presented in Figure 11. This will automatically generate the "ebXML GetContentRequest" query given in Figure 12 to retrieve the OWL archetype definition from the Repository. The user can visualize the results of the query from the "ebXML Result" tab of the GUI given in Figure 11.

Note that there may be archetypes in the registry representing the same clinical concept but annotated with different clinical coding systems. Continuing with our example, assume that the "Complete Blood Count" archetype is annotated with term codes of other terminologies such as "Full Blood Count" of MedDRA [33] rather than with SNOMED "CompleteBloodCount" term. On the other hand, the user may be interested in a certain type of archetype independent of the coding system used to annotate it. In other words, we should be able to find the equivalency among the terms of different coding systems at run time. The US National Library of Medicine's Unified Medical Language System (UMLS) [41] provides a good resource for this purpose. UMLS is the official repository of many healthcare informatics' terminology standards. It contains information over one million biomedical concepts from more than one hundred controlled vocabularies and classifications (some in multiple languages) in the medical domain. It also provides the mapping between different terminologies. We have implemented a Web service, which takes a given coding term and its associated coding system and finds equivalent terms coded through other terminologies by using the UMLS.

Continuing with our example, we use this Web service for querying UMLS Metathesaurus for obtaining the synonyms of SNOMED "CompleteBloodCount" term automatically. This Web service returns:

- MedDRA - Full Blood Count
- Read Codes Full Blood Count

After obtaining the synonyms of SNOMED "CompleteBloodCount" term, the registry is queried for other ExtrinsicObjects, i.e. other archetypes, which are classified with the ClassificationNodes of these synonyms. These queries are expressed as shown in Figure 13.

After obtaining the ID's of the Extrinsic Objects, the archetype definitions can be retrieved from the Repository through "ebXML GetContentRequest" queries similar to the one depicted in Figure 12.

Note that, a healthcare institute may be using some local coding schemes that has not been defined in UMLS. In such a case, these schemas can be stored in the ebXML registry by defining the equivalences of their terms with other Clinical Coding Schema terms through the ebXML "EquivalentTo" association. Then by using the "findEquivalentInstances" stored procedure given in Section 3.3, it is possible to retrieve all the archetype instances of a Clinical coding scheme term as well as the archetype instances of all terms equivalent to this term.

- A user can search against any or all of the template and archetype metadata fields when looking for a template or an archetype.

  For example, it is possible to query the ebXML registry for "archetypes" that has been linked to "Clinical" class with "hasPurpose" property. As shown in Figure 14, when "Archetype" is selected on the left

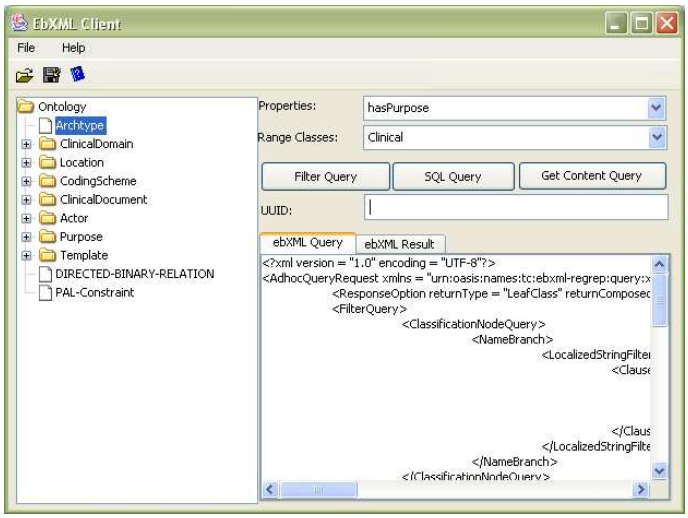

Figure 14: The GUI for discovering the archetypes classified with metadata ontology

pane, all its properties are shown to the user in the related combo box so that the user can make a choice. The range class of the selected property can again be chosen from the ontology presented in the left pane.

This GUI generates the necessary queries to the registry as follows: First the "ebXML IDs of Extrinsic Objects" of the archetypes annotated with the specified property are retrieved. In order to achieve this, two consecutive Filter queries (Figure 15) or one SQL-Query (Figure 16) are used by the system. In the first FilterQuery, the ID of the "Clinical" ClassificationNode is obtained and in the second one the Classifications that have a "type" Slot with value "hasPurpose" and that are bound to the "Clinical" ClassificationNode are retrieved. The result of the second query contains the ID's of the ExtrinsicObjects. These ID's can be used to retrieve the contents of the archetypes through "ebXML GetContentRequest" queries.

6. HOW TO RETRIEVE ARCHETYPE DATA FROM MEDICAL INFORMATION SYSTEMS?

When we want to retrieve the associated data out of individual patient records conforming to the templates and archetypes, the functional interoperability issue also needs to be addressed.

ebXML provides functional interoperability through its messaging system which gives the specification of a standard way to exchange messages between organizations [15]. It does not dictate any particular file transport mechanism, such as SMTP, HTTP, or FTP. It extends the base Simple Object Access Protocol (SOAP) [39] with MIME Attachments [40] for binding. All the interactions with the ebXML Registry as well as the interactions in a business process are specified to be handled through ebXML messages. There are implementations of ebXML messaging available both publicly and commercially.

```
<FilterQuery>
    <ClassificationNodeQuery>
        <NameBranch>
            <LocalizedStringFilter>
                <Clause>
                    <SimpleClause leftArgument = "value">
                        <StringClause stringPredicate =
                         "Equal">Clinical</StringClause>
                    </SimpleClause>
                </Clause>
            </LocalizedStringFilter>
        </NameBranch>
    </ClassificationNodeQuery>
</FilterQuery>

<FilterQuery>
    <ClassificationQuery>
        <SlotBranch>
            <SlotFilter>
                <Clause>
                    <SimpleClause leftArgument = "name">
                        <StringClause stringPredicate =
                        "Equal">type</StringClause>
                    </SimpleClause>
                </Clause>
            </SlotFilter>
            <SlotValueFilter>
                <Clause>
                    <SimpleClause leftArgument = "value">
                        <StringClause stringPredicate =
                         "Contains">hasPurpose</StringClause>
                    </SimpleClause>
                </Clause>
            </SlotValueFilter>
        </SlotBranch>
        <ClassificationFilter>
            <Clause>
                <SimpleClause leftArgument = "classificationnode">
                    <StringClause stringPredicate =
                    "Equal">urn:uuid:ef039f8f-0170-42a6-a329
                    -bfb40c8fe3a9</StringClause>
                </SimpleClause>
            </Clause>
        </ClassificationFilter>
    </ClassificationQuery>
</FilterQuery>
```

Figure 15: Two consecutive ebXML Filter Query

```
<SQLQuery>
    SELECT * FROM extrinsicobject E, classification CL1, slot S
    WHERE S.name LIKE 'type' AND S.value LIKE 'hasPurpose' AND
     S.parent = CL1.id AND CL1.classifiedobject = E.id
     AND CL1.classificationnode IN (
        SELECT C.id FROM Name N, ClassificationNode C
        WHERE C.id = N.parent AND N.value LIKE 'Clinical')
</SQLQuery>
```

Figure 16: ebXML SQL Query

Later, however ebXML also started providing registry support for Web services [16] which is another standard to provide functional interoperability. Web services are a set of related application functions that can be programmatically invoked over the Internet. The information that an application must have in order to programmatically invoke a Web service is given by a Web Services Description Language (WSDL) [48] document. WSDL of a Web service, which defines its interface, is its established public contract with the outside world. The network protocol used is usually HTTP and binding is SOAP [39]. It should be noted that Web services not only provide synchronous invocation but also enable message exchange through asynchronous invocation.

We use Web services for the functional interoperability

layer for the following reasons:

- Web services describe their interfaces through WSDL which gives a machine processable interface definition. Furthermore, WSDL has become a stable definition and there are several tools to automate the construction of an interface defined in WSDL such as IBM Web Services Toolkit [29], or Java Web Services Developer Pack [31].
- Functional interoperability standards, like Web services, need to improve various aspects of the usage scenarios. Several standardization initiatives are under way for providing security [46], privacy [45], transaction support [47], and reliability of Web services [44]. Although ebXML Messaging Services Specification Version 3 [14], also considers some of these issues, it has not been finalized yet.
- Finally, all the application servers support Web services such as BEA WebLogic [4], IBM WebSphere [28] and Oracle Application Server [37].

Web services retrieving data out of individual patient records conforming to the templates and archetypes, can be stored to ebXML registry as "Service" objects. It is possible to annotate such web services with the "Archetype Metadata Ontology" stored in ebXML registy. In this way, the discovery of these Web services through archetype semantics are facilitated.

When retrieving the associated information out of individual patient records conforming to the templates and archetypes through Web services, deciding on the granularity of Web service is important since this effects the service reusability and interoperability with other healthcare standards.

Among the archetype definitions, the ones that contain "elementary" information should be retrieved by "elementary" Web services, whereas composite archetypes should be retrieved by composing elementary Web services through workflow technology. Note that there could be composition relationship between archetypes when at some node in an archetype, a new archetype would occur, rather than a continuation of the constraints in the current archetype. This composition on the other hand can be handled with Web service composition tools as explained in Section 7.

7. COMPOSITE WEB SERVICES

Templates composed of elementary archetypes can be constructed through Web service composition tools. Template designers can refer to the "Archetype Metadata Ontology" in order to indicate what kind of elementary archetypes constitutes such a template. The related elementary archetypes instances can be dynamically discovered and bound to the templates at runtime by using the ebXML registry as described in this paper. Additionally if the user wishes to retrieve the data specified in a template from a specific Medical Information System, the Web services accessing the archetypes can also be discovered from the ebXML registry and can be bound to the template definition.

Consider the "Clinical Information Template" presented in Figure 17. The template designer may annotate the components of this template with Archetype Metadata Ontology rather than explicitly indicating the sub-units of those components. For example the "Allergies" component can be

Figure 17: The Clinical Information Template

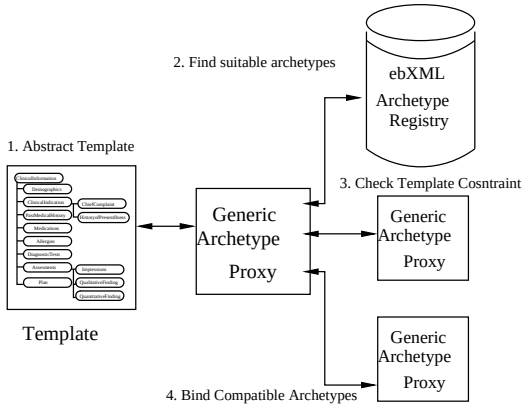

Figure 18: Dynamic Archetype Binding

annotated with "Clinical Information Template" node presented in Figure 2 and "Allergies and Immunology" Clinical Domain presented in Figure 7, and "Medications" components can be annotated with "Medication" Document Type presented in Figure 5.

In order to compose this template, these archetype instances should be discovered from the ebXML Registry. We have provided a "Generic Archetype Proxy" for this purpose. The steps necessary to retrieve archetype instance are as follows (Figure 18):

- The "Generic Archetype Proxy" parses the semantic annotations presented in the Template definition.
- Exploiting these annotations, the "Generic Archetype Proxy" constructs the relevant ebXML Filter queries to retrieve the Extrinsic objects in ebXML registry representing the archetypes.
- Then through "GetContent" queries the content of these archetypes can be retrieved from Repository.
- If further constraints have been defined in the Template definition (such as it should use "SNOMED" Coding Scheme), the "Constraint Checker" component

can assure the suitability of the archetype instance with this template definition.

8. CONCLUSIONS AND FUTURE WORK

Interoperability is a challenging problem in the healthcare domain, and archetypes are a promising approach for tackling this problem. However for archetypes to be used as a shared model of domain concepts, there is a strong need for mechanisms for sharing archetypes through an archetype server, discovering archetypes through their semantics, facilitating template composition from archetypes and retrieval of corresponding data from the underlying medical information systems.

In this paper, we provide guidelines on how ebXML Registries, through their semantic constructs, can be exploited as an efficient medium for annotating, storing, discovering and retrieving archetypes. We also present how archetype data can be retrieved from proprietary clinical information systems by using ebXML Web services. This work is carried out within the scope of Artemis project [3] which aims to provide interoperability in the healthcare domain through semantically enriched Web services.

As already mentioned archetype based interoperability necessitates "harmonised" information model to be adopted. However, as an alternate solution, the reference models of the standardization bodies like openEHR, CEN TC/251, and HL7 can be represented through an ontology language like OWL. As a future work, we plan to define the archetypes constraining these reference models in OWL, and provide the mapping between these reference models through ontology mapping.

9. REFERENCES

[1] Aden, T, Eichelberg, M., Artemis Deliverable D3.1.1.4: Review of the State-of-the-Art- Healthcare standards: HL7, GEHR and CEN TC251 ENV 13606, CEN TC251 prEN 13606-1, http://www.srdc.metu.edu.tr/webpage-/projects/artemis/calendar.php.

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

[3] Artemis Project, http://www.srdc.metu.edu.tr/webpage/projects/artemis

[4] BEA WebLogic Platform, http://e-docs.bea.com/platform/docs81/index.html

[5] Berners-Lee, T., Hendler, J., Lassila, O., "The Semantic Web", Scientific American, May 2001.

[6] Beale, T., Heard, S.,"Archetype Definitions and Principles", http://www.openehr.org/repositories/spec-dev/latest/publishing/architecture/archetypes/-principles/REV_HIST.html

[7] Beale, T., Heard, S.,"The OpenEHR archetype system", http://www.openehr.org/repositories/spec-dev/latest/publishing/architecture/archetypes/system/-REV_HIST.html

[8] 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

[9] CEN TC/251 (European Standardization of Health Informatics) prEN13606, Electronic Health Record Communication, http://www.centc251.org/

[10] Dogac, A., Laleci, G. B., Kabak, Y., Cingil, I., "Exploiting Web Service Semantics: Taxonomies vs. Ontologies", IEEE Data Engineering Bulletin, Vol. 25, No. 4, December 2002.

[11] Dogac, A., Kabak, Y., Laleci, G., "Enriching ebXML Registries with OWL Ontologies for Efficient Service Discovery", in Proc. of RIDE'04, Boston, March 2004.

[12] A. Dogac, Y. Kabak, G. Laleci, C. Mattocks, F. Najmi, J. Pollock, "Enhancing ebXML Registries to Make them OWL Aware", Submitted to the Distributed and Parallel Databases Journal, Kluwer Academic Publishers. http://www.srdc.metu.edu.tr/webpage/-publications/2004/_DAPD_ebXML-OWL.pdf.

[13] ebXML, http://www.ebxml.org/

[14] ebXML version 3, http://www.oasis-open.org/-committees/download.php/4383/ebMSv3FeaturePreview

[15] ebXML Message Service Specification v1.0, May 2001, http://www.ebxml.org/specs/ebMS.pdf.

[16] ebXML Registry Information Model v2.0, http://www.ebxml.org/specs/ebrim2.pdf

[17] ebXML Registry Information Model v2.5, http://www.oasis-open.org/committees/regrep/-documents/2.5/specs/ebRIM.pdf

[18] ebXML Registry Services Specification v2.5, http://www.oasis-open.org/committees/regrep/-documents/2.5/specs/ebRIM.pdf

[19] Elkin, P., Kernberg, M., "HL7 Template and Archetype Architecture", HL7 Template Special Interest Group, Jan. 2004.

[20] Fensel, D., Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.

[21] Heard, S., Beale, T.,Mori, A.R., Pishec, O.,"Templates and Archetypes: how do we know what we are talking about" http://www.openehr.org/-downloads/archetypes/templates_and_archetypes.pdf

[22] IHE IT Infrastructure Integration Profiles, http://www.rsna.org/IHE/tf/ihe_tf_index.shtml

[23] IHE IT Infrastructure Technical Framework, Cross-Enterprise Clinical Documents Sharing (XDS), by the IHE ITI Technical Committee, Version 3.0, April 26, 2004.

[24] Health Level 7 (HL7), http://www.hl7.org

[25] HL7 Clinical Document Architecture, Release 2.0. PDF format. Version: August 30, 2004, http:/xml.coverpages.org/CDA-20040830v3.pdf

[26] HL7 Reference Information Model, http://www.hl7.org/library/data-model/RIM/modelpage_non.htm

[27] HL7 Version 3 Message Development Framework, http://www.hl7.org/library/mdf99/mdf99.pdf

[28] IBM WebSphere Application Server Technology for Developers V6, http://www-106.ibm.com/developerworks/websphere/-downloads/WASTD6Support.html

[29] IBM Web Services Toolkit, http://www.alphaworks.-ibm.com/tech/webservicestoolkit

[30] ISO TC/215, International Organization for Standardization, Health Informatics, ISO TS 18308:2003, secure.cihi.ca/cihiweb/en/downloads/-infostand_ihisd_isowg1_mtg_denoct_contextdraft.pdf

[31] Java Web Services Developer Pack (Java WSDP),

http://java.sun.com/webservices/jwsdp/index.jsp
[32] Jena2 Semantic Web Toolkit, http://www.hpl.hp.com/semweb/jena2.htm
[33] MedDRA: Medical Dictionary for Regulatory Activities, http://www.fda.gov/medwatch/report/meddra.htm
[34] Staab, S., Studer, R., Handbook on Ontologies, Springer, 2004.
[35] OASIS: Organization for the Advancement of Structured Information Standards, 1998, http://www.oasis-open.org/cover/siteIndex.html.
[36] openEHR Community, http://www.openehr.org/
[37] Oracle Application Server Web Services Tutorials, http://www.oracle.com/technology/tech/webservices/-htdocs/series/index.html
[38] SNOMED Clinical Terms, http://www.snomed.org/snomedct_txt.html
[39] SOAP: Simple Object Access Protocol, http://www.w3.org/TR/SOAP/.
[40] SOAP Messages with Attachments, http://www.w3.org/TR/SOAP-attachments.
[41] Unified Medical Language System (UMLS), http://www.nlm.nih.gov/research/umls/
[42] UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/index.htm.
[43] Web Ontology Language OWL, http://www.w3.org/TR/owl-features/
[44] Web Services Reliability (WS-Reliability), http://developers.sun.com/sw/platform/technologies/-ws-reliability.html
[45] Handling Privacy In WSDL 2.0, http://www.w3.org/-TeamSubmission/2004/SUBM-p3p-wsdl-20040213/
[46] Web Services Security (WS-Security), http://www-106.ibm.com/developerworks/webservices/-library/ws-secure/
[47] Web Services Transaction (WS-Transaction), http://www-106.ibm.com/developerworks/webservices/-library/ws-transpec/
[48] WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl.