Published in final edited form as:
Int J Semant Comput. 2013 September ; 7(3): 237–255. doi:10.1142/S1793351X13500037.

Semantically Interoperable XML Data

Cristobal Vergara-Niedermayr[*],
Oracle, California, USA

Fusheng Wang,
Center for Comprehensive Informatics, Department of Biomedical Informatics, Emory University, Atlanta, Georgia, USA

Tony Pan,
Center for Comprehensive Informatics, Department of Biomedical Informatics, Emory University, Atlanta, Georgia, USA

Tahsin Kurc, and
Center for Comprehensive Informatics, Department of Biomedical Informatics, Emory University, Atlanta, Georgia, USA

Joel Saltz
Center for Comprehensive Informatics, Department of Biomedical Informatics, Emory University, Atlanta, Georgia, USA

Cristobal Vergara-Niedermayr: cristobal.vergara.niedermayr@gmail.com; Fusheng Wang: fusheng.wang@emory.edu; Tony Pan: tony.pan@emory.edu; Tahsin Kurc: tkurc@emory.edu; Joel Saltz: jhsalz@emory.edu

Abstract

XML is ubiquitously used as an information exchange platform for web-based applications in healthcare, life sciences, and many other domains. Proliferating XML data are now managed through latest native XML database technologies. XML data sources conforming to common XML schemas could be shared and integrated with syntactic interoperability. Semantic interoperability can be achieved through semantic annotations of data models using common data elements linked to concepts from ontologies. In this paper, we present a framework and software system to support the development of semantic interoperable XML based data sources that can be shared through a Grid infrastructure. We also present our work on supporting semantic validated XML data through semantic annotations for XML Schema, semantic validation and semantic authoring of XML data. We demonstrate the use of the system for a biomedical database of medical image annotations and markups.

Keywords

Biomedical Data Management; XML Database; Data Integration; Semantic Interoperability

[*]Work done at Center for Comprehensive Informatics, Emory University

1. Introduction

Paucity of interoperable data sources is one of the major obstacles to efficient sharing and use of biomedical datasets. Autonomously developed and maintained data sources often manage data in different formats and in a variety of database systems. Different schemas may be used to represent the same data types; different semantics may be associated with data elements and data content. Such differences make it difficult for biomedical researchers to query, retrieve, and integrate information from disparate data sources.

XML has become the de facto standard for exchanging data and information in the Web. XML database technologies [1; 2; 3; 4; 5] also are rapidly maturing, allowing efficient persistence of data as XML documents. Health and biomedical sciences domains have developed data models and standards based on XML [6]. For example, the latest HL7 standard Clinical Document Architecture is based on XML. Standard organizations such as IHE [7], ASTM, and CDISC develop their standards partially based on XML. Many XML based data models have been developed in different fields of biomedicine, such as the Bioinformatic Sequence Markup Language (BSML) to represent sequence data, the Microarray gene expression (MAGE) standard to represent gene expression data, the Annotation and Image Markup (AIM) model [8] for Radiology images, and the Pathology Analytical Imaging Standards data model [9; 10; 11] to represent markups and annotations on Pathology images.

When data is exchanged between two end-points in a Grid environment as XML documents conforming to a published XML schema (i.e., W3C XML Schema), consumers of the XML documents can parse them correctly. That is, the structure of the data is well defined and can be consumed programmatically, enabling *syntactic interoperability* among resources. However, the XML schema by itself is not sufficient to enable semantically interoperable resources. Semantic Web technologies provide a comprehensive framework for supporting semantic interoperability. Most real world data, however, are in relational or XML databases. This limits the applicability of the framework. Common data elements, controlled vocabularies, and rich metadata models play an important role in achieving syntactic and semantic inter-operability. They are core building blocks in software systems based on the model driven architecture (MDA), an architecture approach to support implementation of interoperable systems. In the MDA, the information structure of a data source is expressed as object-oriented information models using the Unified Modeling Language (UML). The classes and attributes in these models draw from common data elements, which are published and registered in the environment and annotated by concepts (terms) from controlled vocabularies. In this way, both the structure of data and its semantics are well-defined and published, enabling clients to discover, query, and integrate data sources that use these models to represent their data.

In this paper, we first present a framework that employs the concepts of MDA, common data elements, and controlled vocabularies, in order to create interoperable Grid data services, which encapsulate XML databases as backend data management systems. In our framework, the development of a data service starts with an XML schema and results in a Grid data

service that exposes the backend XML database through an object-oriented, semantically annotated information model.

In addition to support of semantic harmonization at the level of data models and their data elements, we propose a framework for enforcing the validity of XML data. Semantic annotations in the information model can then be applied to the original XML schema, which can then be transformed into a form for validating XML data submitted into XML databases, to guarantee the semantic validity of XML data sources.

We leverage technologies developed and employed by the cancer Biomedical Informatics Grid (caBIG) [12], such as caDSR [13], EVS [14], and caGrid in our implementation. caBIG is a national-scale effort funded by the National Cancer Institute. The caBIG program has developed a suite of interoperable applications, informatics standards and best practices, and a common Grid infrastructure to address the increasing informatics requirements of cancer research, promote and enable efficient data and tools sharing among cancer research groups, and facilitate large scale collaborative efforts.

This paper is organized as follows. We will first introduce related work, and present an overview of caBIG technologies and processes as an example approach for syntactic and semantic interoperability of resources. We then describe our work on the semantic extension for xService to support XML data in Section 4. In Section 5, we discuss XML Schema annotation derived from the semantic annotated information model, and how we can apply it to semantically validate XML documents for XML databases (Section 6). A real world use case is discussed in Section 7, followed by conclusion.

2. Related Work

XML based approach for managing biomedical data becomes increasingly popular. Existing applications in biomedical domains include XNAT [15], which provides an XML view based approach for managing neuroimaging and related data. There is also work done to provide unique XQuery based front end for relational based biomedical data sources [16]. Project Mobius [17] provides generic mapping of XML Schema elements to relational tables, although XML query support is limited. SciPort [18; 19; 20] provides an XML based approach for modeling, managing and integrating scientific experiments and result data. XML is also used to integrate heterogeneous bio-molecular data [21].

XML based standards are commonly used for data representation and data exchange for biomedical and healthcare applications. HL7 [22] version 3 is partially defined in XML Schemas. AIM is a caBIG In Vivo Imaging Workspace project [8] to provide standardization for image annotation and markup for radiology images, which is implemented in XML – XML Schema based model representation and XML based data services [23; 24]. Pathology Analytical Imaging Standards (PAIS) [11] is a standard for representing image analysis results and human annotations for pathology and microscopy images. PAIS uses an XML based representation to represent and share data. Both AIM and PAIS are or will be harmonized with caBIG semantic infrastructure.

Semantic annotations have been used for Web Services through the OWL-S standard [25] for automatic service discovery, invocation and composition. Work on OWL based data modeling and harmonization has been proposed in [26]. Open MDR [27] is a suite of software that provides caGrid-compatible semantic metadata management capabilities. Our work can also be easily extended to use Open MDR for semantic annotations for XML data models. SciPort [28] provides semantic based authoring through automatically discovering related ontology terms from a user typed keywords. Data element level semantic annotations or constraints are not provided.

There has been much research on data mapping from UML to XML Schema [29]. A survey of work on XML Schema to UML is presented in [30]. There are a couple of goals for our mapping: i) comprehensiveness – support all concepts in XML Schema, ii) round-trip mapping – the transformed UML model can be transformed back to its original XML Schema, iii) semantic equivalence – the UML model is semantically meaningful for users to make annotations. The first approach is presented in [31] based on an early version of XML Schema, by introducing stereotype based extensions to UML to support mapping of elements and attributes, model groups, and enumerations. The work in [32] uses a UML profile based approach to address major XML schema concepts. However, model groups such as sequence and choice are not considered. The weakness in [32] is addressed in [33] to incorporate support of enumerations, list and union type constructors, with a proprietary approach. In [34], a platform independent logical schema is used to provide direct mapping using a UML profile. The coverage of XML Schema concepts is broad, but several mappings do not conform to UML. Our approach provides a very comprehensive coverage of XML Schema concepts, including entity/attribute mapping, relationship mapping, inheritance mapping, and query mapping [35]. Query mapping is through metadata annotations in the UML model using tagged values to guarantee same query semantics for CQL and XPath. We also allow users to modify mapped UML class names where the semantics is preserved using annotations. Through automated annotation during XML Schema to UML mapping, the mapping is also round-trip: annotated UML model can be mapped back into annotated XML Schema.

3. caBIG Interoperability Infrastructure

The caBIG project developed a process and tools to assist with the implementation of interoperable data models and data services. The process consists of the following main steps: i) Creation of an object-oriented model of the backend data format or schema – queries to the corresponding data service are composed on the object-oriented model; ii) Registration and harmonization of the classes and attributes of the object-oriented model as common data elements. A *Common Data Element* (*CDE*) is the smallest entity with a semantic annotation. If there is a common data element already registered in the caBIG environment to represent a data type or attribute, this data element is reused in the model. Common data elements and reuse enable interoperability at the data element level. iii) Annotation of the new common data elements with terms from a controlled vocabulary. The semantic annotation is key to correctly understanding and interpreting the meanings of the data elements in the model and being able to consume them; and iv) Harmonization of the object-oriented model as an annotated information model using the registered common data

elements. This step ensures that the model and its data elements are interoperable with other models, which use registered common data elements, and that data represented by the model can be accessed programmatically and interpreted correctly.

Once the object-oriented model has been harmonized and registered for use, a service developer can create caGrid data services to share data in the caBIG Grid environment. caGrid is a Grid middleware infrastructure with a service oriented architecture. It facilitates the implementation and deployment of data and analytical resources as services, provides Grid-wide security, and supports federated queries across caGrid data services. A client application can use caGrid tools to compose queries using the Common/caGrid Query Language (CQL) [36]. CQL queries are object-oriented and translated into SQL queries through a middleware-level mapping layer and executed on databases of the data sources. Query results are returned as sets of objects (transferred as serialized XML documents over the wire) to the client application.

caBIG leveraged two infrastructure-level components to support the process of implementing interoperable data models and services. NCI Enterprise Vocabulary Service (EVS) [14] provides access to controlled biomedical terminologies which are administered by the National Cancer Institute (NCI). These terminologies are built on standard terminologies as well as those created by the community. They are stored in a database, called the NCI Thesaurus. Each concept in a terminology is uniquely identified by an ID in this database and linked to a human readable definition – an *EVS concept* (*also referred to as a data element concept*) is a tuple consisting of one concept and its definition. The concepts are used by model creators to describe data elements to be shared and by users who want to search for relevant datasets. Cancer Data Standards Registry and Repository (caDSR) [13] stores UML models, common data elements (CDE), and semantic metadata associated with them. At this level, associations and inheritance relationships among classes are defined. The relationships define rules on how data can be queried. For example, a query containing a join operation between two classes of objects is possible only if the classes are associated in the model. Model creators annotate their object-oriented models and the model's classes and attributes with concepts from the EVS service using a tool called SIW [37], thus creating common data elements. A common data element is also associated with a value domain, which specifies a representation of said data element using concrete data types (e.g., float, integer) and permissible values (e.g., a data element can take values between −1 and +5, or can take a value from a set of discrete values).

Figure 1 illustrates an example of a common data element *Protein Type*, which is used to annotate a property in a UML model. In this example, a property "type" in a UML model is annotated as a CDE. This CDE is composed of a concept and a value domain. All these entities reside in caDSR and are linked to controlled vocabulary definitions in the NCI Thesaurus provided by the EVS. Permissible values for the value domain are also defined in the EVS.

4. Developing Interoperable XML-based Data Services

To support sharing of existing XML data from existing XML schemas, we developed caGrid XML Data Service Framework (xService) [24]. xService is an extension of caGrid for querying and retrieving XML documents managed in XML databases. xService provides query mapping from CQL to XML query language (XPath), and generic XML database interfaces for uploading, updating, querying and retrieving of XML data from diverse XML data sources such as Berkeley DB XML, IBM DB2, and eXist XML databases. xService also provides an extension to Introduce Toolkit for users to flexibly and rapidly create their own caGrid XML data services based on a predefined XML Schema. In the rest of this section, we will present the extensions to xService that allow for creation and semantic annotation of object-oriented models.

To take advantage of existing caBIG semantic infrastructure and tools for supporting annotations – which relies on UML for the modeling and semantic annotations, our approach is to create a structurally and semantically equivalent UML model (xUML) from an existing XML based model specified in an XML Schema document. We then can take advantage of all existing semantic harmonization tools to enable the semantics for XML models. The xUML follows exactly the same process for caBIG semantic harmonization as other models. The final semantically annotated model (aUML) is loaded into the XML data service creation wizard provided by xService to create a caBIG compatible data service. This process is illustrated in Figure 2 (top part). Next we explain the details of each step.

4.1. Transformation of XML Schema into UML Model

The goal of transforming XML Schema into UML model is to create an equivalent UML based model (xUML) that can rely on existing caBIG semantic annotation and harmonization tools. This is challenging as XML schema model is graph based and UML model is object-based. We develop a comprehensive data model mapping tool XSD2UML [35]. XSD2UML is written in XSLT [38], a W3C standard based pattern and rules oriented XML transformation language. XSD2UML transforms XML Schema document (in XML format) to XMI (XML based representation of UML model).

Mappings from XML Schema Constructs into UML Constructs—We have the following principles in mind when mapping a XML Schema to a UML model: i) Represent all the entities and their attributes in the UML model; ii) Represent all relationships between entities in the model; iii) Represent inheritance relationships between entities, and iv) Preserve necessary information for mapping CQL queries to XPath queries at runtime. Next we will first give an overview of the fundamental model concepts in XML Schema and UML, and how these principles are followed during the mapping process. (The query mapping will be discussed in Section 4.3.)

XML Schema provides data type definitions and data element definitions. Data types can be primitive types (e.g., String, Integer, etc.), simple types, and complex types. Simple types are derived from primitive types or other simple types by restricting the domain of another derived or primitive type, or by defining a union of the domains of derived and primitive types. A complex type defines data types composed of other complex types or primitive

types. The *element* construct defines an XML element in an XML document. An element can contain attributes, nested elements (defined by complex type) or text based value of a primitive or simple type. Nested elements naturally generate parent/child relationships. Inheritance between complex types can be defined by either extending the parent type with new components or omiting components from the parent type. UML model is object-oriented, and provides constructs for representing classes, attributes, associations, and generalizations.

We defined the following rules in our mappings:

1. *Entity/attribute mapping*. Each complex type in an XML Schema is mapped to a class. A complex type with simple content maps to a UML class with an attribute to store the value of the text. Attributes, elements based on a simple or primitive type in an XML Schema, and element with an attribute “type” that has as value either a primitive type or the name of a simple type or an attribute “ref” referencing another element with the same characteristics are mapped to UML attributes.
2. *Relationship mapping*. Elements based on a complex type are mapped to associations in UML. The cases include element with a complex type child, element with a “type” attribute for a named complex type definition, and element with a “ref” attribute referencing an element of complex type.
3. *Inheritance mapping*. There are four different inheritance mapping cases:
    - Inheritance from complex types using extension. A child class is created with the additional components defined. Additionaly, a generalization relationship between child and parent is created.
    - Inheritance from complex types using restriction. A child class with the omitted components is created for consistency in associations. No generalization relationship is created, since there is no equivalent concept for this kind of inheritance in UML.
    - Inheritance from simple types using restriction. The type of the primitive type in the top of the inheritance hierarchy is used as the type for the UML attribute.
    - Inheritance from simple types using union. The most general type that can represent all the types in the union is selected. For example, for a union of integer and float, float is selected for the UML attribute type, since an integer can also be represented as float. The most general type is string. When none of the types are eligible, string is used. For example, for a union of integer and date, string will be used in the UML attribute.
4. *Query mapping*. To guarantee proper query mapping, for those necessary information that can not be directly represented in UML, we create metadata annotations in the UML model using tagged values. These metadata can be summarized as follows:

a. For each class in UML, we store the top level elements in the XML Schema allowing the appearance of the complex type that was mapped to the class. (Top level elements are elements that can be used as root elements of XML instance documents). A CQL query searching for objects of a specific type, shall return only documents having a root element of the complex type mapped to the class targeted by the query. If no top level elements in the schema refer the complex type mapped to the class targeted by the CQL query, an empty result set shall be returned immediately.

b. For each attribute in the UML model we store a tag specifying if the attribute was defined in the schema as an attribute, an element of primitive type, or a simple type. This information allows the CQL query processor to create the appropriate XPath query.

c. For each destination class in associations in the UML model, we keep the absolute XPath query of the element. This allows the CQL query processor to create more efficient XPath queries that allow the database to instantly locate the destination element in the instance instead of searching for it through the whole document.

Figure 3 illustrates an example of association mappings described above (detailed explanation in [35]).

XSD2UML tool provides comprehensive mappings for complex XML Schema constructs [35]. XSD2UML will generate the xUML document based on the mapping rules. The generated model could also be reviewed and adjusted (e.g., naming changes) (Figure 2) if necessary through UML modeling tools, such as Enterprise Architect and Argo UML. The xUML can then be used as the starting point for semantic harmonization, as discussed in the next section. The caBIG infrastructure supports only annotated UML models in XMI format, without support for annotating XML Schemas. By transforming XML Schema into an equivalent UML model, we could take advantage of the suite of tools already developed and the curation/harmonization workflow provided by the working group. XSD2UML is not only integrated into xService, but also available as a standalone tool for general XML Schema to UML model transformation. It can be downloaded from its svn repository [39].

4.2. Annotating XSD Equivalent UML Model

One of the means of the caBIG workflow to reach semantic integration is the harmonization process conducted by the caDSR team that reviews the UML models submitted by the caBIG model creators. The reviewers use the SIW tool for curating the UML models that are stored in caDSR and referenced by the Indexing Service. In an interactive process involving the model creators and caDSR reviewers, new UML models with potential new CDEs are loaded in caDSR. The harmonization process assures that only well defined and strongly typed objects are transferred between services and applications, managed in a central repository and available for the caBIG community [13; 40]. With semantically harmonized UML model (aUML), we can then create XML data service, as discussed next.

4.3. Generating Semantic Enabled XML Data Service in xService

xService [24] provides a framework to create caGrid data service for XML based data sources. xService provides a mapping layer between caGrid and XML for both data model mapping (XML Schema to UML model, and UML model to caGrid domain model) and query mapping (CQL to XPath), thus caGrid client applications can view XML data sources virtually as UML models and use CQL to query them. The data model mapping is generated at the service creation time, and it generates a domain model, an XML based representation of the aUML model defined by caGrid for querying purpose.

The CQL to XPath mapping is also created at service generation time, and executed when a client queries the service through invoking the CQL query processor. The CQL query processor transforms CQL queries to the XML query language of the backend XML data sources.

The xService creation wizard is an extension of the caGrid Introduce toolkit [41], which will take an annotated UML (aUML) (or XML Schema without annotations), and automatically generate the service APIs, domain model file, WSDL files, and other artifacts. An overview of the whole workflow is illustrated in figure 2.

The generated XML data service can then be deployed to an application server and provides all the functionalities to support a caGrid node based on the XML database.

5. Semantic Annotation for XML Schema

The semantic infrastructure of caBIG provides a mechanism for different applications and data sources to reuse, create and share data elements from common data element repositories linked to ontologies. Although the framework provides harmonization at the level of data models and their associated data elements, caBIG does not provide tools for applications to enforce the semantic validity of the data.

Data semantics has to be enabled at data authoring time, and validated before they are loaded into the database. In many cases, data management and authoring applications are isolated from common data element repositories or ontologies, and there is a lack of tools for semantic enforcement. Next, we will discuss our work on providing generic semantic annotations for XML Schemas to provide XML data validation, based on the semantically annotated information model discussed above. The workflow is illustrated in Figure 2.

5.1. Types of Semantic Annotations for XML

The purpose of semantic annotation for XML Schema is to define constraints related to data element or data value semantics thus XML data can be semantically validated before they are loaded into XML databases. XML Schema has its own constraints such as identity, referential, cardinality, and data type constraints. XML Schema can also constrain data values through enumerations. These constraints are, however, at the structural level and isolated from any semantic infrastructure. Valid XML documents based on the structural constraints will not guarantee its semantic validity for semantic interoperability through

semantic infrastructures such as caBIG. Semantic constraints have to be provided through its own approach, as we discuss next.

In general, there are two types of semantic constraints – data element constraints and data value constraints. *Data element constraint* defines a data element is linked and mapped to a common data element in a common data element repository (or an ontology concept from an ontology). The type of annotation discussed in previous section falls into this category. As a CDE has its value domain, such constraint will eventually enforce the possible values of a data element in a local XML database, i.e., through the permissive values of the value domain. For example, a *protein Type* will be constrained by the four permissive values defined by the value domain: Glycoprotein, Lipoprotein, Immunoglobulin Family Protein, Metalloprotein, Phosphoprotein.

Value constraint specifies the possible values of data elements, and it can vary between applications even for the same data element. For example, for image annotations [11], users need to provide observations of the characteristics of images. The type of observations for brain tumor can be very different from those for prostate cancer. For example, a study will use the following observations: “Section Location” (CDE id 2673929), “Glioblastoma Slide Proliferating Cell Number” (CDE id 2673930), and “Percent positive of marker ULN” (CDE id 2627046). (Here we take three of 15 observations for illustration purpose.) When data are collected, only a predefined set of observations are allowed. The difference of value constraint from data element constraint is that, a data element constraint maps a data element to a common data element thus the data element follows the CDE constraint (i.e., permissible values of a vale domain), while a value constraint defines a “template” that defines possible data elements (e.g., the three CDEs above) that can be used as values for a data element (e.g., Observation).

5.2. Specifying Semantic Annotations in XML Schema

Semantic annotation in XML Schema can be specified through the annotation capability provided by XML Schema syntax. XML Schema provides an *xsd:annotation* tag that is specifically used for adding application specific annotations. We can add annotation elements under the *xsd:appinfo* element. Two types of annotation elements are provided: *Data Element Annotation* for data element semantic constraint, and *Value Annotation* for value based semantic constraint.

Data Element Annotation (Figure 4) has three attributes: *coding Scheme Designator*, *code Value*, and *coding Scheme Version*. coding Scheme Designator defines the common data element repository or the ontology used; code Value defines the code of the CDE (or ontology concept) mapped to this data element, and coding Scheme Version defines the version of the repository. All semantic annotations added through UML models can be represented in this approach.

Value Annotation (Figure 5) has a list of constrained *Value* elements. Each Value element includes the same attributes as Data Element Annotation: coding Scheme Designator, code Value, and coding Scheme Version.

Benefits of Semantics Annotation in XML Schema—One significant advantage of XML Schema based semantic annotation over UML based semantic annotation is the usability. XML has rich set of tools for visualizing and processing, and is defined with human readable tagged text. It is much easier to process and transform an XML Schema document than a UML document – the later has very complex syntax and often depends on proprietary tools which are difficult to extend. Another advantage is that the semantic annotation is unified within a single XML Schema document, and it can be easily exchanged and shared.

we develop a tool on taking the semantic annotations from the UML model (aUML) back to the original XML Schema. From there, we can also add additional value constraints that further enrich the semantics specifications. Another immediate benefit is that with annotated XML Schema (aXSD), we are able to perform semantic validation for XML documents based on the aXSD, and provide constraints for authoring documents.

6. Semantically Validating and Authoring of XML Documents

The annotated XML Schema provides semantic constraints which can be used to support XML data validation and semantic enabled authoring, as discussed next.

6.1. Semantic Validation of XML Documents

The benefit of semantic validation of data is to enable semantic interoperable data at the data instance level. For example, while the data element *protein Type* can be semantically discovered across different data sources through its CDE ID, the values of this data element can be arbitrary and at authors' will, if they are not semantically enforced. Such potentially mismatched data element values will eventually counteract the effort of semantic infrastructures such as caBIG.

We provide an approach to semantically validate an XML document through validating the document against a semantically transformed XML Schema (vXSD). This is achieved by transforming the semantic constraints as native XML Schema constraints such as enumeration or constrained data types. The steps are as follows.

- Retrieving constrained values from CDEs with their value domains. For example, caDSR repository provides APIs to retrieve permissible values for a data element. For value based constraints, we can retrieve the "Preferred Question Texts" for all constrained data elements if desired. The approach can be extended to use permissible values from ontologies as well. The retrieved enumerated values are initially stored in a constraint XML document.
- Transforming XML Schema document by adding constraints. The constraint values retrieved from CDE or ontology repositories can then be used to transform the XML Schema into a Schema with enumeration based constraints. XML Schema standard provides inheritance of primitive types, where new types called simple types can be created. Simple types are atomic, but can have values belonging to a domain formed by the union of multiple domains of other simple or primitive types, or a subset of the domain of a simple or primitive type. Creating a new type

by unifying two domains of existing simple types is accomplished by using the union construct, and creating a new type by restricting the domain of an existing type can be defined using the restriction element. For example, in Figure 6, the *Protein* type is transformed from a *String* type into a *Protein Type* simple Type, which contains an enumeration of the data element's permissive values retrieved from caDSR.

We develop an XSLT based tool to transform aXSD to vXSD. The tool takes the aXSD XML Schema document and the constraint XML document and transforms them into a vXSD XML Schema document.

- Validating XML documents against vXSD XML Schema document. Once we have the vXSD schema document ready, we are able to use standard XML document validation tools such as javax.xml. validation API, Xerces Java Parser to validate XML documents. Such applications can be customized to display exact what data element is not valid and for what reason.

6.2. Semantically Enabled Authoring of XML Data

The approach of semantic validation can be similarly applied to enabling semantics for data authoring. The validation vXSD XML Schema document indeed can be used to restrict the choice of values for users while authoring XML data. The authoring application can automatically look for constraints through the enumerations defined in vXSD document for each field to be authored, thus create a constrained dropdown list for users to choose (the approach is application specific.) For example, Figure 6.2 shows a possible form generated from the vXSD document to generate a drop down list for the protein type.

7. Use Case: NCI TCGA AIM Data Services

The xService system we develop is open source, and is available for download at xService wiki [24]. XSD2UML is also available as a standalone tool for XML Schema to UML model transformation [39]. xService works on both Windows and Linux/Unix systems. xService supports caGrid 1.2 and caGrid 1.3. Currently three XML databases are supported: Berkeley DB XML database [1], eXist [4] and DB2 pure XML [3]. xService has been deployed for supporting HL7 based XML data (aECG) and AIM data.

AIM is a caBIG project developed by Northwestern University and Stanford University [8]. AIM provides standardized representation for medical image annotation and markup, especially for clinical trials. The model includes dozens of classes such as patient, observer, equipment, image, anatomic entities, image observations and image observation characteristics, geometric shapes, text annotations, and calculations. The classes Anatomic Entity, Imaging Observation, and Imaging Observation Characteristic classes represent essential features for an annotation. The representation of AIM data is an XML based format. AIM data management and sharing for NCI TCGA Enterprise Use Case [42] is through xService (AIME [43]), which supports semantically enabled XML data services for sharing the data via caGrid. Figure 8 shows an example screenshot of making semantic annotations for the UML model generated from XML Schema. Figure 9 shows a screenshot of the generated AIM data service in the Introduce Toolkit of xService. In future work, we

can implement semantic validation of XML documents into XML databases to provide automated validation during data injection process through user-defined functions.

8. Conclusion

Semantic discovery of data services and data harmonization is among the major goals to provide interoperabable data sharing among biomedical research communities. In this work, we develop a generic framework and tool set xService to support the development of semantically interoperable Grid data services and semantically validated XML data sources. We take advantage of existing caBIG semantic infrastructure to support XML data models through developing additional data mapping, query mapping and XML service data service authoring tools. Besides supporting XML data model and common data element level harmonization for data sharing infrastructure, we develop a method for semantic annotation for XML Schema, thus XML data can be semantically validated or semantically enabled through data authoring before they are loaded into XML data sources. With our work, we are able to not only provide semantically valid XML data sources, but also support semantically sharing of the data from these data sources.

Acknowledgments

This research is supported in part by PHS Grant UL1RR025008 from the CTSA program, by R24HL085343 from the NHLBI, by Grant Number R01LM009239 from the NLM, by NCI Contract No. N01-CO-12400 and 94995NBS23 and HHSN 261200800001E, by NSF CNS 0615155, 79077CBS10, and CNS-0403342, and P20 EB000591 by the BISTI program.

References

1. cabig. https://cabig.nci.nih.gov/
2. caGrid XML Data Service Framework (xService). https://web.cci.emory.edu/confluence/display/xmlds/
3. Cancer data standards registry and repository (cadsr). https://cabig.nci.nih.gov/concepts/caDSR/
4. CQL: Common/caGrid Query Language. http://cagrid.org/display/dataservices/CQL
5. DB2 Express-C Overview. http://www-306.ibm.com/software/data/db2/express/
6. Enterprise vocabulary services (evs). https://cabig.nci.nih.gov/concepts/EVS/
7. eXist XML Database. http://www.exist-db.org/
8. Health Level Seven International (HL7).
9. Integrating the health care enterprise (ihe). http://wiki.ihe.net/
10. Introduce: The cagrid service creation toolkit. http://cagrid.org/display/introduce/Home
11. The mobius project at osu bmi. http://www.projectmobius.org
12. Open MDR. http://cagrid.org/display/MDR/
13. Oracle Berkeley DB XML. http://www.oracle.com/database/berkeley-db/xml/
14. Oracle XML DB. http://www.oracle.com/technology/tech/xml/xmldb
15. owl-s: Semantic markup for web services. http://www.w3.org/Submission/OWL-S/
16. Semantic integration workbench (siw). https://cabig-kc.nci.nih.gov/Vocab/KC/index.php/Semantic_Integration_Workbench_%28SIW%29
17. Tamino: The XML database. www.softwareag.com/us/products/wm/tamino/
18. uml model browser. https://wiki.nci.nih.gov/display/caDSR/UML+Model+Browser
19. Xsl transformations (xslt). http://www.w3.org/TR/xslt
20. Semantic integration of xservice for the creation of cabig compatible xml data services. 2010. https://web.cci.emory.edu/confluence/download/attachments/1015923/CristobalThesis.pdf

21. Aim data service at emory university. 2011. https://web.cci.emory.edu/confluence/display/IVI/AIME
22. Pathology analytical imaging standards. 2011. https://web.cci.emory.edu/confluence/display/PAIS
23. Tcga enterprise use case. 2011. https://wiki.nci.nih.gov/display/Imaging/TCGA+Enterprise+Use+Case
24. Xsd2uml svn repository. 2011. https://scm.cci.emory.edu/svn/xservice/branches/xService_1.3.5_caBIGSemanticIntegration/projects/caBIGSemanticIntegration
25. Bales N, Brinkley J, Lee ES, Mathur S, Re C, Suciu D. A framework for xml-based integration of data, visualization and analysis in a biomedical domain. XSym. 2005:207–221.
26. Bernauer, M.; Kappel, G.; Kramler, G. Representing xml schema in uml a comparison of approaches. In: Koch, N.; Fraternali, P.; Wirsing, M., editors. Web Engineering, volume 3140 of *Lecture Notes in Computer Science*. Springer; Berlin Heidelberg: 2004. p. 440-444.
27. Booch G, Christerson M, Fuchs M, Koistinen J. Uml for xml schema mapping specification. White paper, Rational. 1999
28. Carlson, D. Modeling XML Applications with UML: Practical e-Business Applications. Addison-Wesley; 2001.
29. Channin D, Mongkolwat P, Kleper V, Sepukar K, Rubin D. The caBIG Annotation and Image Markup Project. Journal of Digital Imaging. 2009
30. Domínguez E, Lloret J, Pérez B, Rodríguez A, Rubio A, Zapata M. A survey of uml models to xml schemas transformations. Web Information Systems Engineering. 2007:184–195.
31. Foran DJ, Yang L, Chen W, Hu J, Goodell LA, Reiss M, Wang F, Kurç TM, Pan T, Sharma A, Saltz JH. Imageminer: a software system for comparative analysis of tissue microarrays using content-based image retrieval, high-performance computing, and grid technology. JAMIA. 2011; 18(4):403–415. [PubMed: 21606133]
32. Marcus D, Olsen TR, Ramaratnam1 M, Buckner RL. The extensible neuroimaging archive toolkit: An informatics platform for managing, exploring, and sharing neuroimaging data. Neuroinformatics. Mar.2007 :11–33. [PubMed: 17426351]
33. McCarthy, J.; Warzel, D.; Kendall, E.; Bargmeyer, B.; Solbrig, H.; Keck, K.; Grey, F. Data modeling and harmonization with owl: Opportunities and lessons learned. Fifth International Workshop on Semantic Web Enabled Software Engineering; 2009.
34. Mesiti M, Jimnez-Ruiz E, Sanz I, Berlanga-Llavori R, Perlasca P, Valentini G, Manset D. Xml-based approaches for the integration of heterogeneous bio-molecular data. BMC Bioinformatics. 2009; 10(Suppl 12 S7)
35. Provost, W. Ml for w3c xml schema design. 2002.
36. Routledge N, Bird L, Goodchild A. Uml and xml schema. ADC. 2002:157–166.
37. Shabo A, Rabinovici-Cohen S, Vortman P. Revolutionary impact of xml on biomedical information interoperability. IBM Syst J. 2006; 45(2):361–372.
38. Wang F, Bourgué P-E, Hackenberg G, Wang M, Kaltschmidt D, Liu P. Sciport: an adaptable scientific data integration platform for collaborative scientific research. VLDB. 2007:1310–1313.
39. Wang F, Kong J, Cooper L, Pan T, Tahsin K, Chen W, Sharma A, Niedermayr C, Oh TW, Brat D, Farris AB, Foran D, Saltz J. Data model and database for high-resolution pathology analytical image informatics. Journal of Pathology Informatics. 2011; 2(32)
40. Wang, F.; Liu, P.; Pearson, J. chapter SciPort: An Extensible Data Management Platform for Biomedical Research. World Scientific; Database Technology for Life Science and Medicine. Forthcoming
41. Wang F, Liu P, Pearson J, Azar F, Madlmayr G. Experiment management with metadata based integration for collaborative scientific research. ICDE. 2006
42. Wang F, Pan T, Sharma A, Saltz J. Managing and querying image annotation and markup in xml. SPIE Medical Imaging. 2010:762805–11.
43. Wang F, Thiel F, Furrer D, Vergara-Niedermayr C, Qin C, Hackenberg G, Bourgué P, Kaltschmidt D, Wang M. An adaptable xml based approach for scientific data management and integration. SPIE Medical Imaging. 2008; 6919

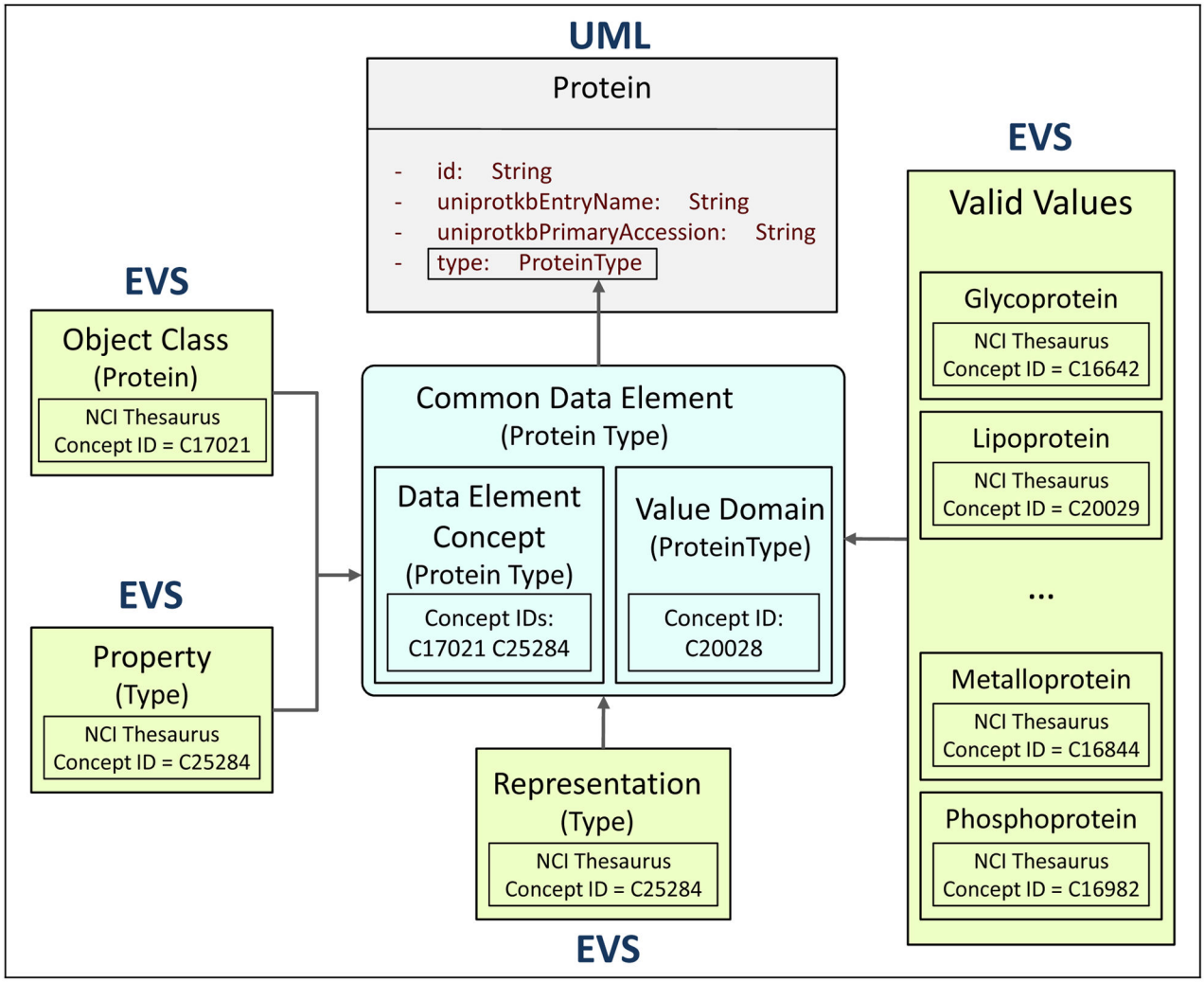

Fig. 1.
An Example Common Data Element

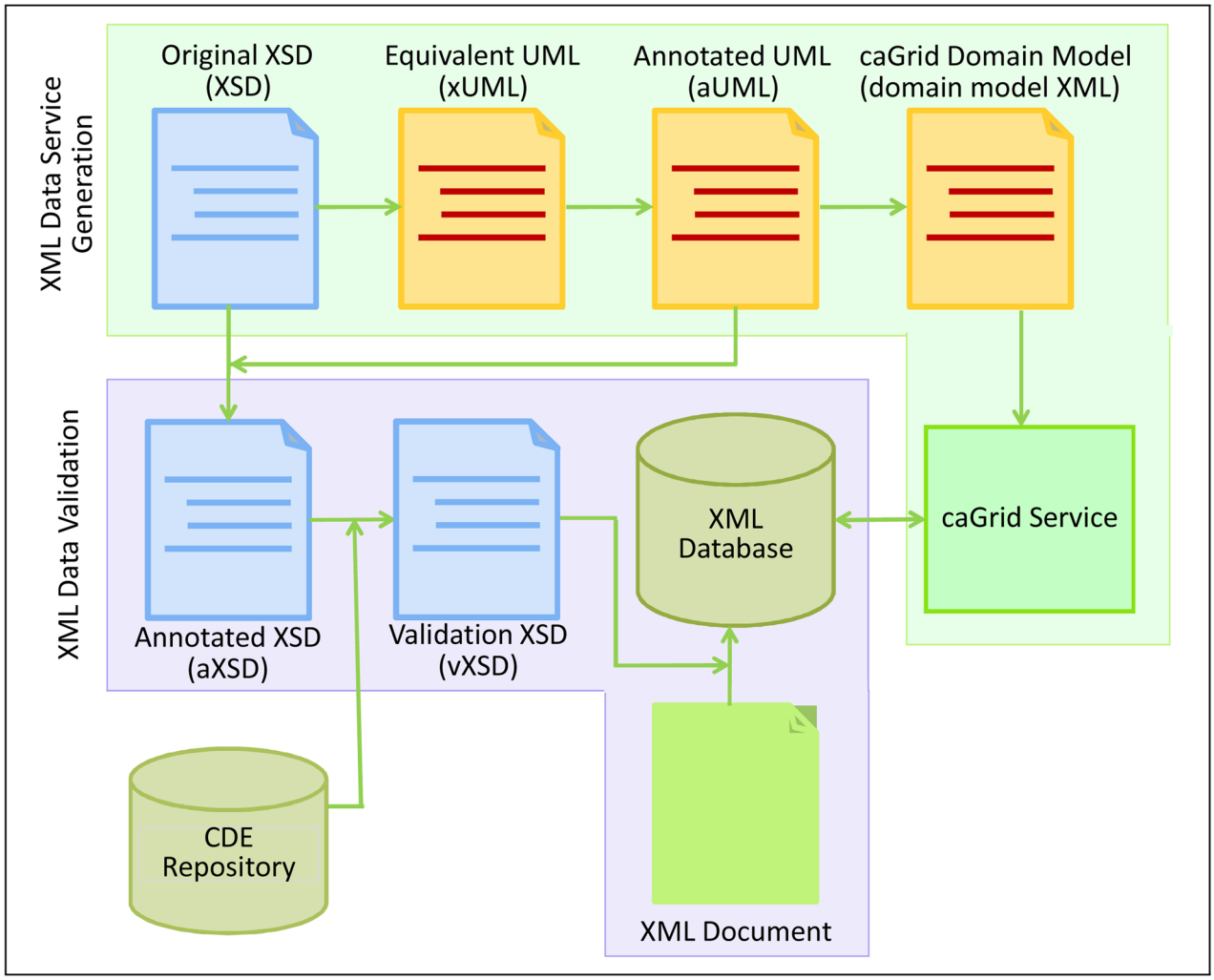

Fig. 2.
Overview of XML Data Service Generation and Data Validation

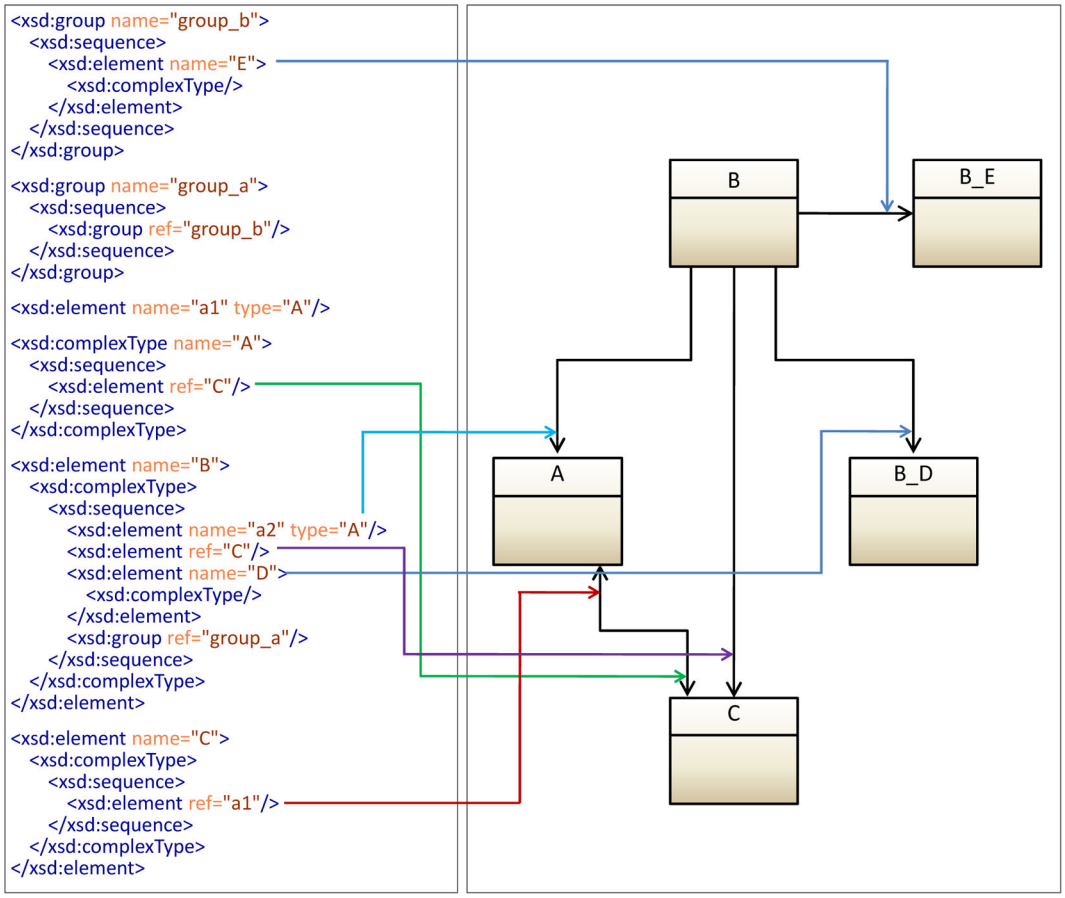

Fig. 3.
An Example Mapping of Relationships to Associations

```
<xsd:complexType name="Protein">
  <!-- ... -->
  <xsd:attribute name="type" type="xsd:string">
    <xsd:annotation><xsd:appinfo>
        <sa:DataElementAnnotation codingSchemeDesignator="caDSR"
          codeValue="2863117" codingSchemeVersion="1.0"/>
    </xsd:appinfo> </xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
```

Fig. 4.
An Example of an XML Schema with Data Element Annotation

```
<xsd:complexType name="Observation">
  <xsd:attribute name="name" type="xsd:string">
    <xsd:annotation><xsd:appinfo>
      <sa:ValueAnnotation>
        <sa:Value codingSchemeDesignator="caDSR"
          codeValue="2673929" codingSchemeVersion="1.0"/>
        <sa:Value codingSchemeDesignator="caDSR"
          codeValue="2673930" codingSchemeVersion="1.0"/>
        <sa:Value codingSchemeDesignator="caDSR"
          codeValue="2627046" codingSchemeVersion="1.0"/>
      </sa:ValueAnnotation>
    </xsd:appinfo></xsd:annotation>
  </xsd:attribute>
</xsd:complexType>
```

Fig. 5.
An Example of an XML Schema with Value Annotation

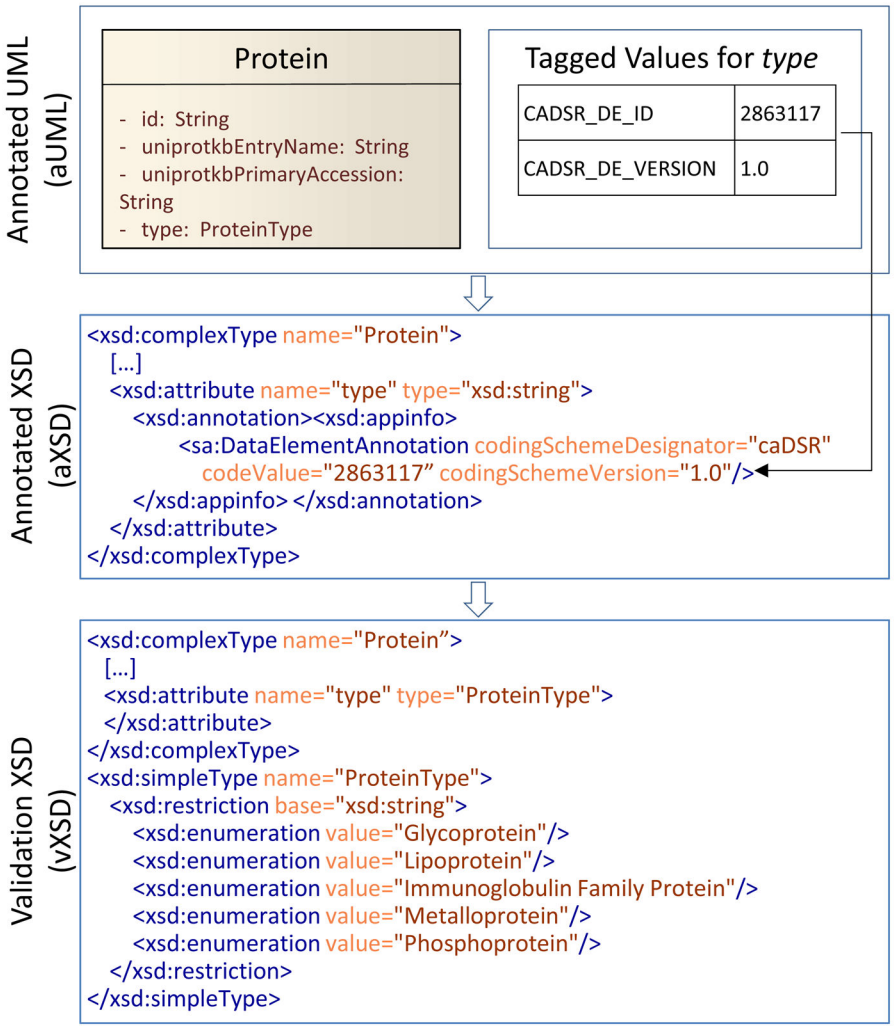

Fig. 6.

An Example of Validation XML Schema (Only One Example Data type Is Illustrated)

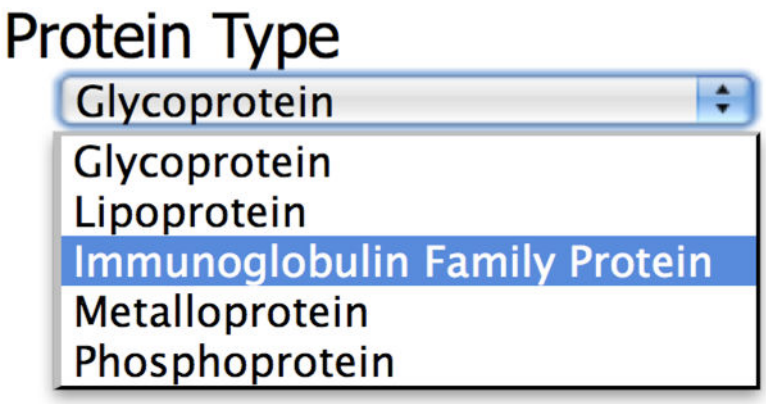

Fig. 7.
An Example of Authoring Data with Controlled Values

Fig. 8.

An Example Screenshot of Semantic Annotation for UML Model { xUML

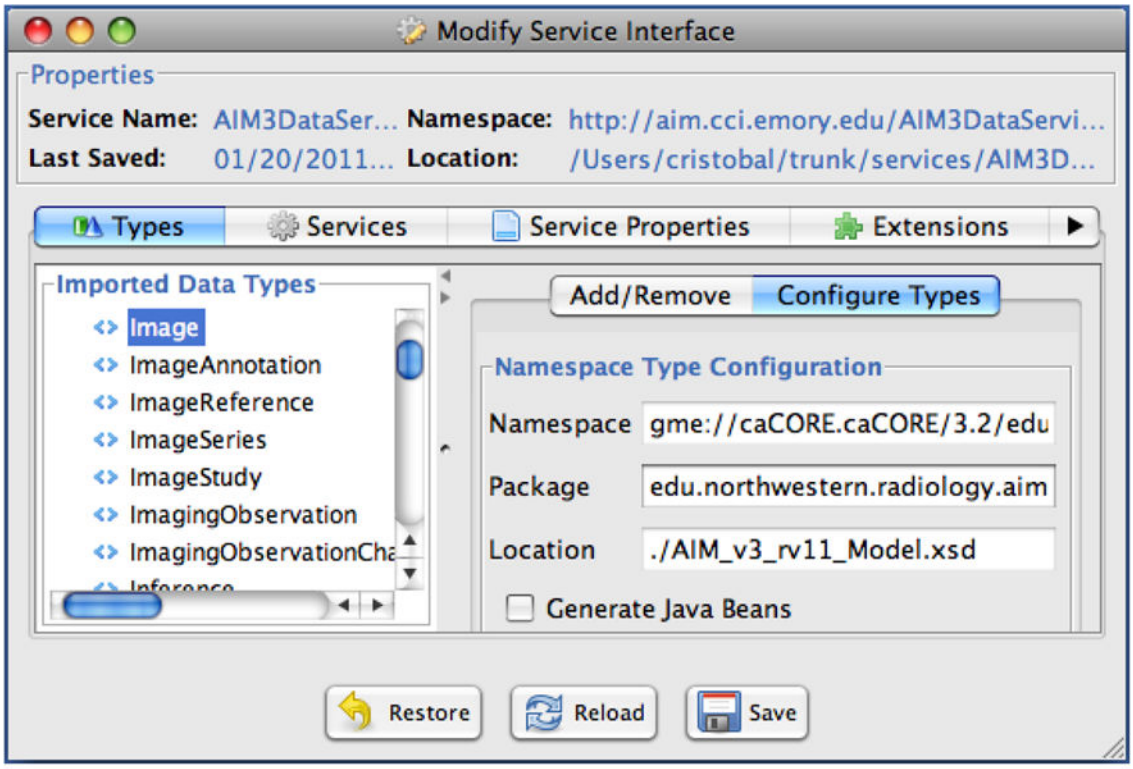

Fig. 9.
An Example screenshot of generated caGrid data service