HIWAS: Enabling Technology for Analysis of Clinical Data in XML Documents

Joshua Hui
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
(1) 408 -927-1721
jhui@us.ibm.com

Sarah Knoop
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
(1) 408 -927-2622
seknoop@us.ibm.com

Peter Schwarz
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
(1) 408 -927-1750
schwarz@almaden.ibm.com

ABSTRACT

The information contained in large collections of clinical data can be used for many valuable purposes, such as epidemiological studies, evidence-based medicine, monitoring compliance with best clinical practices, and cost-benefit analyses. However, the emerging standards for the electronic representation of clinical data, such as the Clinical Document Architecture (CDA) [4], are very complex and new tools are required to effectively extract and utilize the information contained in these documents.

In this paper, we present HIWAS, a research prototype of a new tool that creates a structural summary of a collection of XML documents, thereby enabling users to find relevant information for a specific purpose within complex XML documents. A HIWAS user can create a target model that contains just the information they need, in a simplified representation that can be queried efficiently and is compatible with existing relational business intelligence technology. By making these complex XML documents digestible with conventional tools, HIWAS lowers a key barrier to meaningful use of aggregated clinical data.

1. INTRODUCTION

Around the world, governments are taking steps to encourage the electronic interchange of healthcare information. A key part of this effort is the development of standards for the representation of clinical data, so that information produced by one organization can be consumed and acted upon by another. For example, in the United States, recent rulings by the Department of Health and Human Services have identified the establishment of such standards as a critical step on the path to achieving "meaningful use" of electronic medical records. In addition to the obvious benefit of better care coordination for individual patients, the information contained in large collections of electronic health records can be used for many other valuable purposes, such as epidemiological studies, evidence-based medicine, monitoring compliance with best clinical practices, cost-benefit analyses, and more. Although they were originally conceived as a medium for information exchange, standard-compliant documents are now increasingly being considered as a useful long-term storage representation for clinical data, thanks to their flexibility, extensibility, and ability to accurately preserve the context of a clinical event

A leading contender among the proposed standards for healthcare data is the XML-based Clinical Document Architecture (CDA), developed by the international healthcare standards organization Health Level Seven (HL7)[14]. The standard was designed to facilitate several goals, among them the ability to represent virtually any kind of clinical data. While this flexibility is one of the key benefits of CDA, it also poses significant challenges for the design of software intended to aggregate and analyze large collections of clinical data obtained from a variety of sources. In the first place, today's business intelligence tools, such as IBM Cognos [17], BIRT [5], SAS [30] or SPSS [21] are primarily designed to work with data in a tabular format, as one might find it in a relational database or spreadsheet. Therefore, data represented in XML must be converted to relational form before these tools can be applied, and a naive choice for the relational representation can result in poor performance on analytic queries. Furthermore, although numerous schema-mapping tools have been proposed and/or built, including Clio [10], Altova MapForce[1], and Stylus Studio [28], all of them rely heavily upon XML schemas as the means for describing the source, target and implementation of the mapping. The same is true of typical Extract/Transform/Load (ETL) tools, such as IBM Infosphere DataStage [18], and Oracle Warehouse Builder [27]. However, as we discuss in greater detail in the next section, the size, complexity and generic nature of the CDA schema makes such tools difficult to use in practice for this application.

This paper presents HIWAS, a research prototype for a new kind of tool that lowers these barriers to analyzing the information contained in large sets of standard-compliant healthcare documents. Although we focus specifically on the Clinical Document Architecture in this paper, the same model-driven methodology [12] was used to develop the entire family of HL7 Version 3 messaging standards [11], and similar methodologies are employed in other industries as well. The HIWAS tool gives those who work with such documents the ability to find the information they need for a particular purpose and extract it in a representation that can be analyzed using the primarily relational business intelligence technology that is available today.

Several key aspects of HIWAS differentiate it from other tools in this space. First, HIWAS inspects the collection of documents to be analyzed and builds a structure called a Semantic Data Guide

Proceedings of the VLDB Endowment, Vol. 4, No. 12

(SDG) that focuses the user's attention on those structural variants that actually exist in the collection, typically a much smaller set than those theoretically allowed by the schema. Secondly, HIWAS identifies document elements in the SDG with semantically-meaningful names derived from supplemental information, rather than generic element names derived solely from the XML schema. Lastly, HIWAS allows the user to selectively drag-and-drop elements of interest from the SDG into a target model that is constructed incrementally, rather than requiring a target schema to be constructed prior to mapping.

The remainder of this paper is organized as follows. In the next section, we provide additional background on the Clinical Document Architecture, to illustrate the mechanisms used in a typical XML-based healthcare standard to control document structure, and to show why accessing the information from such documents can be so challenging. In Section 3, we describe the warehousing environment in which our tool is intended to function, and provide an overview of how it is used. Section 4 describes our technology for summarizing the structural variations in a collection of documents and for finding information of interest, and Section 5 describes how a user can select, extract and restructure information for analysis with conventional tools. Section 6 presents a study, in which we built a purpose-specific warehouse from data contained in a collection of public health documents, and details the benefits we derived from using HIWAS. Section 7 summarizes the contributions of this work and outlines potential future improvements.

2. ANALYZING STANDARD-COMPLIANT HEALTHCARE DATA

As we noted in the introduction, XML-based standards like the Clinical Document Architecture pose several challenges for analytic applications. Some of these challenges come from the details of how the CDA standard represents clinical data, but others are more general and likely to apply in many situations where analysis of complex XML documents is required. We discuss both types of challenge in this section.

2.1 Generic Challenges for XML Analytics

Although XQuery and XML database technology have been around for some time, the vast majority of the marketplace relies on relational technology for querying, aggregation, ETL and report generation. In part, this is because relational technology is more mature, leading to better performance, especially on complex analytic queries. Beyer et al. [2] point out that XQuery, as originally specified, lacks key constructs for expressing OLAP-style queries, and workarounds using other constructs can lead to poor performance. Some of these shortcomings have been remedied by the adding a grouping construct to the language, but OLAP-style queries are still complex and difficult to write. Another important consideration is the need to integrate data represented in XML with reference information from other sources that is typically available in tabular form. For example, in a public health laboratory report, a laboratory test may confirm the presence of a specific strain of Salmonella, say Salmonella tennessee 6,7,14;z29;1,2,7. In the report, this kind of information would be represented as an observation whose value is a clinical code, say from SNOMED CT [25], that specifically identifies this organism. One can easily imagine, however, that a public health agency might want to count the number of Salmonella incidents for a given location and time period, regardless of specific strains. The challenge is that each document bears the precise code of the Salmonella serotype identified, and not the relationship with the genus Salmonella. The latter information is captured in the code system, which in the case of SNOMED CT includes over 300,000 concepts and their relationships, all of which is external to the documents being analyzed. This is but one example; clinical data typically contains coded values for many concepts, such as diagnoses, medications, and even mundane things like postal codes, all of which need external reference data to be interpreted and/or aggregated.

In theory, it should be possible to pose the necessary queries in SQL/XML [8] or a similar language, and execute them on a database management system that supports queries across both relational data and XML, such as IBM DB2 or Oracle. Our experience, however, has been that this approach is also quite difficult in practice. The complexity of the CDA documents (see below) makes these queries hard to write, and good performance is hard to achieve. Another alternative is to transform all or part of the XML data, and store it in relational form. However, naïve approaches to relational storage can also result in poor performance. We advocate transformation to a carefully-designed hybrid relational-XML schema, tailored to a specific set of use cases, that places key attributes in tables for easy retrieval and integration with reference data, while preserving rarely-used information in snippets of XML. HIWAS is a design-time tool intended to help domain experts with limited knowledge of the CDA standard create such transformations. In Section 3, we will describe in greater detail how HIWAS is used, but first we describe some particular characteristics of the CDA standard that make designing such transformations difficult with existing tools.

2.2 Designing Transformations for CDA

While the challenges outlined above apply to many situations involving analysis of XML data, particular aspects of the Clinical Document Architecture exacerbate these problems and introduce some additional ones. To start with, the CDA schema is very large and complex. Table 1 provides some statistics that illustrate the sheer size of the schema, which also includes mutually recursive types and other complex constructs.

Table 1 Statistics for the CDA schema.

	Number
Elements (with different types)	951
Attributes within unique elements	492
Complex types	220
Simple types	2016

Although the number of types and elements is large, a typical document only makes use of a small number of these types. See Section 6 for additional statistics that characterize some specific kinds of documents.

The size of the schema is not the only source of complexity. The following fragment of a CDA-compliant document illustrates some additional issues.

```
<observation classCode="OBS" moodCode="EVN">
  <templateId root="2.16.840.1.10.20.1.28"/>
  <id root="ab1791b0-5c71-b0de-0800200c9a66"/>
  <code code="282291009"
    displayName="Diagnosis"
```

```
    codeSystem="2.16.840.1.113883.6.96"
    codeSystemName="SNOMED CT"/>
  <text>
    <reference value="#prob-1"/>
  </text>
  <statusCode code="completed"/>
  <effectiveTime value="20070509"/>
  <value xsi:type="CD" code="46177005"
    displayName="End-stage renal disease"
    codeSystem="2.16.840.1.113883.6.96"
    codeSystemName="SNOMED CT">
    <translation code="584.9"
      codeSystem="2.16.840.1.113883.6.103"
      codeSystemName="ICD-9"
      displayName="Acute kidney failure"/>
  </value>
  <entryRelationship typeCode="REFR">
    <observation classCode="OBS"
      moodCode="EVN">
      <templateId root="2.16.840.120.1.50"/>
      <templateId root="2.16.840.120.1.57"/>
      <code code="33999-4"
        displayName="Status"
        codeSystem="2.16.840.1.113883.6.1"
        codeSystemName="LOINC"/>
      <statusCode code="completed"/>
      <value xsi:type="CE" code="55561003"
        codeSystem="2.16.840.1.113883.6.96"
        displayName="Active"
        codeSystemName="SNOMED CT"/>
    </observation>
  </entryRelationship>
</observation>
```

Note that many elements in this document fragment (and their types) are generic, like `observation`. Knowing the name or type of such an element provides very little information about the semantics of the data it contains. Such generic schema elements facilitate stylesheet-level interoperability, but pose problems for applications that wish to locate and/or extract a specific kind of information from a collection of documents. For example, using the element tags alone, it is not possible to craft an XPath expression to uniquely identify the observation in this document as a diagnosis.

There exist other cues in the document that provide additional information about content, but tools driven entirely by the XML schema cannot take advantage of them. Instead of relying on the schema, HL7's methodology uses an additional mechanism, *templates*, to constrain document structure and content to suit a specific clinical purpose. A template is a set of rules[1] for the structure of a clinical document that goes beyond what is required for conformance to the underlying CDA schema. For example, the template for a specific kind of observation, like a lab result, may require a timestamp element to be present, whereas the CDA schema for a generic observation may allow this element to be absent. Similarly, the template may require a specific clinical code from LOINC [26] be used to differentiate a lab result observation from other kinds of observations, or to encode the identity of the particular test that was conducted.

Although template definitions are not included in documents, note that the sample document contains `templateId` elements at various levels designating the templates to which those portions of the document conform. The document also contains other items, like codes, whose meaning is defined externally and can be used as cues that provide information about the semantics of document portions. Codes, template identifiers and other such cues allow XML elements that would otherwise be schema-wise identical to one another to be distinguished. For example, the outer observation in the sample document can be recognized as a Problem Observation from its templateId, and more specifically as a Diagnosis by the presence of the appropriate code from SNOMED CT. The inner observation includes template identifiers and codes that indicate a Problem Status Observation.

The sample document also illustrates how many of the constructs in a typical CDA document, such as `classCode`, `typeCode` and `entryRelationship`, do not convey clinical information, but rather reflect how the construct being documented was derived from HL7's Reference Information Model (RIM) [12]. The presence of this information leads to additional layers of structure that tend to obscure the actual content, and can mostly be ignored when extracting clinical data for analysis.

Even once elements of interest have been identified, designing a correct and complete transformation can still be a complex task. The CDA standard allows for considerable flexibility in how particular constructs are represented. The same or similar information may be represented in different ways, or appear in different locations within a document, depending on the vendor that produced the document or the context in which the information was obtained. All these variations must be taken into account in creating a transformation, and care must be taken to match corresponding pieces of information correctly.

Lastly, we note that much of the information in a CDA is not relevant for all use cases. For example, in the sample document, the `statusCode` which indicates whether an event (an observation, in this case) has taken place or is merely contemplated, may be important in a clinical care setting but is rarely of interest in a research study. A selective approach to transformation and target representation is indicated, rather than a single all-purpose solution.

3. AN OVERVIEW OF HIWAS

In the previous section, we described how standard-compliant clinical data is represented, and outlined the challenges in transforming such data for analysis. In this section, we describe how an investigator would build a clinical data warehouse using conventional tools, and contrast it with how the same task could be accomplished using HIWAS.

3.1 Building a Data Warehouse

Consider a typical warehousing environment, as shown in Figure 1. CDA documents created by various sources (electronic medical record systems, lab systems, etc.) are collected in an operational store whose primary purpose is day-to-day patient care. Queries against the operational store typically seek to retrieve one or more documents about a particular patient. To facilitate queries over aggregated data from many patients, information is extracted from the operational store to a data warehouse with a more-specific purpose. As we noted previously, much of the information in the operational store may not be relevant for the purpose envisioned for the warehouse. For

[1] The manner in which these rules are specified does not concern us here. The normative specification is typically in English, but implementors have used machine-processable representations, such as Schematron or Object Constraint Language to codify the rules.

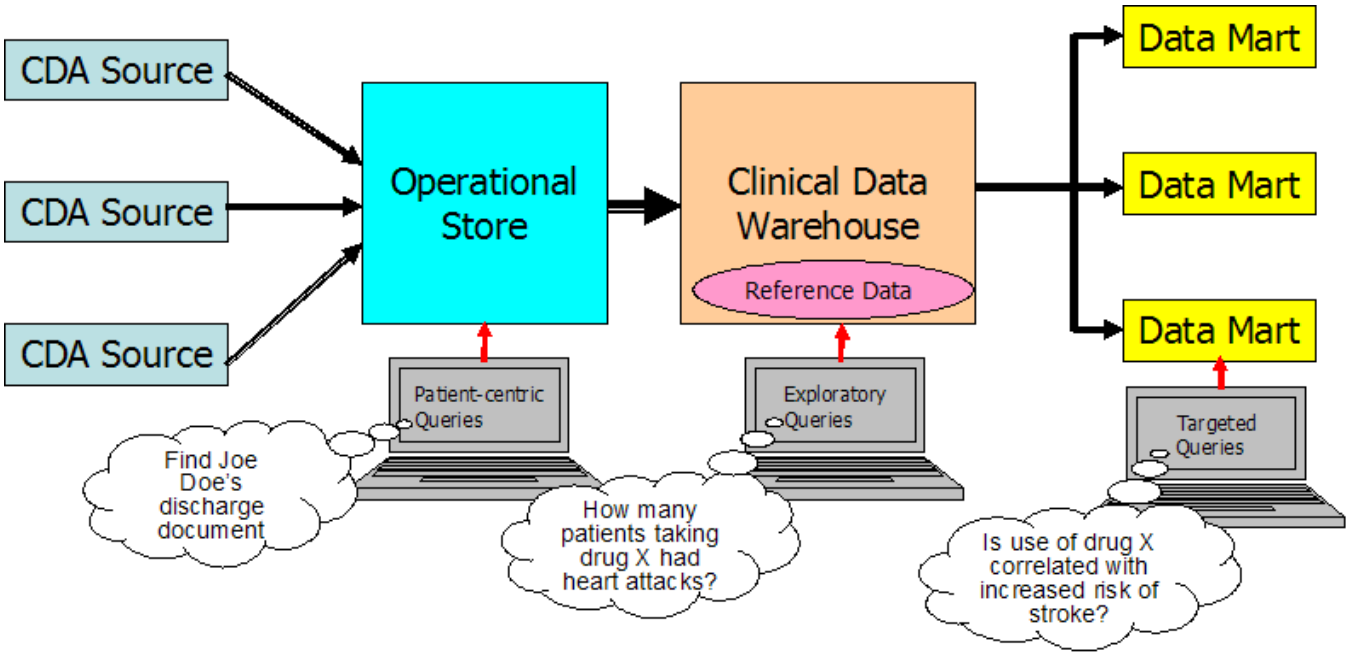

Figure 1 The HIWAS Warehouse Environment.

example, if the warehouse is intended for clinical investigations, custodial information about documents or test specimens is probably unnecessary. However, this information might be very important for a warehouse intended to monitor regulatory compliance.

Unlike queries posed against the operational store, queries against a clinical warehouse focus on aggregated data from many patients. Before undertaking an in-depth study, a researcher must establish how many patients meet basic criteria and which patients' data should be included in the study. Our goal is to make it possible to carry out such queries directly on the warehouse using conventional business intelligence tools. For the actual study, further cleansing and transformation of the data may be needed, and data for the selected patient cohort is often exported to a datamart using Extract/Transform/Load (ETL) tools. Once again, our goal is to facilitate the use of existing tools for this purpose.

We focus, therefore, on tools for building the warehouse, because it is at this stage in the workflow that data conforming to healthcare XML standards must be integrated with reference data and made compatible with existing tools. To build a warehouse of clinical information concerning, for example, cancer patients, an investigator using today's tools would be faced with a difficult and largely manual task. Starting from a complex specification and thousands of de-identified documents, the investigator would first have to undertake a period of manual inspection, to better understand the data available and to determine which pieces are relevant for this particular warehouse. The investigator would then need to decide how best to store this information, so that queries and relational-based business intelligence software could digest it efficiently and integrate it with reference information like value sets and disease taxonomies. Typically, this would involve designing both a relational target schema and an executable mapping to populate the target schema from CDA documents. Existing schema-mapping tools that rely heavily on the XML schema of the source documents as a means of describing the data to be transformed would be of little value, since, for example, the schema does not delineate a primary diagnosis of cancer from a family history of cancer, nor differentiate the document section containing current medications from the section containing laboratory test results. Schema mapping tools also assume that the target schema has been designed beforehand, and then require the expert to explicitly connect source and target elements that correspond to one another, typically by drawing lines in a graphical user interface. With documents as complex as CDA instances, the number of explicit connections needed to express a comprehensive mapping rapidly becomes unmanageable. Lastly, since the source of each correspondence is a generic schema element that might contain many different types of information, elaborate conditions must be added to each mapping to select only the items desired.

3.2 Designing a Warehouse Using HIWAS

The process for accomplishing the same task using HIWAS is illustrated in Figure 2. First, HIWAS helps the investigator explore the available data by constructing a Semantic Data Guide (SDG), a structural summary of a collection of XML documents that takes advantage of codes, template identifiers and other information in the documents to replace generic element names based on the XML schema with meaningful ones. The SDG also provides statistical information about the frequency with which specific constructs occur in a particular context, co-occurrence of concepts in documents, and the like. The SDG is described in greater detail in Section 4 of this paper.

After using the SDG to identify data of interest, HIWAS helps the investigator construct a simpler representation of that information, called a *target model*, as shown in step 3 of Figure 2. Like the source XML documents, the target model is hierarchical. Using a drag-and-drop paradigm, the user selects information of interest from the SDG and adds it to the target model. Although not powerful enough to allow arbitrary restructuring of the input, the

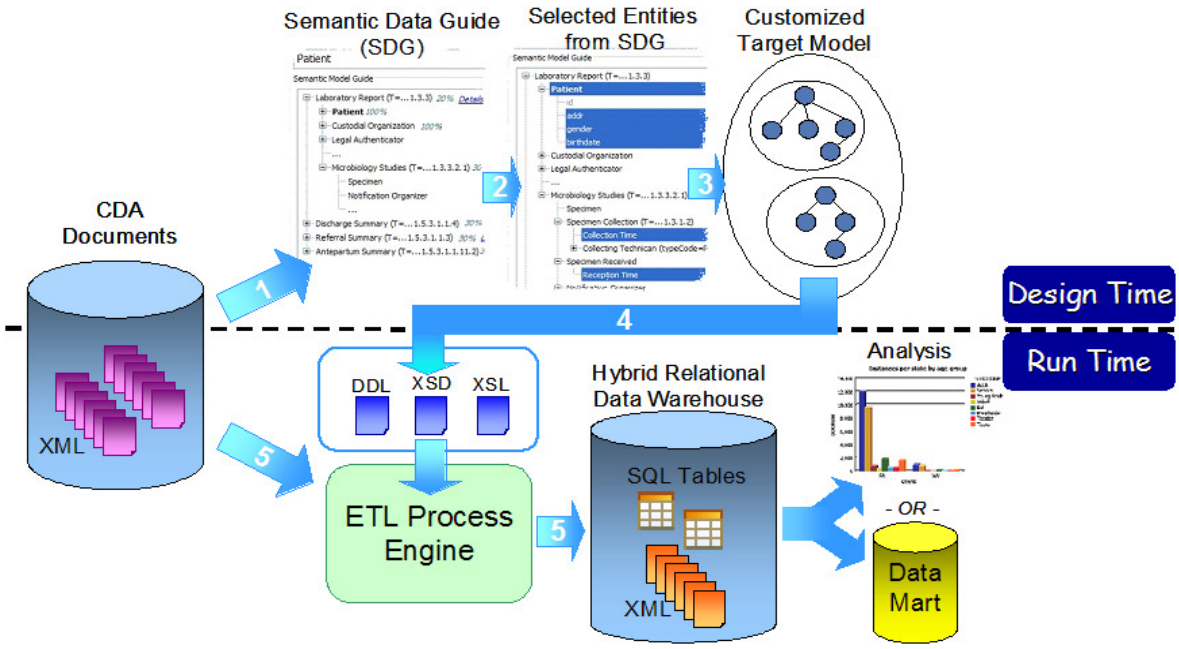

Figure 2 Building a Warehouse with HIWAS.

target model editor provides enough restructuring capability for most situations encountered in practice. Once the target model has been constructed, no additional modeling is necessary if the investigator wishes to keep the transformed data as XML. However, as we have noted, most business intelligence tools available today are designed to work primarily with relational data, and therefore, HIWAS can automatically produce a relational model based on the hierarchical target model the user has constructed. The target model editor is described in more detail in Section 5.

The final step is to instantiate the target model and populate it with data from transformed documents. Requirements for this stage vary widely, depending on whether the result is to be XML or relational, the number of documents to be transformed, how new documents are obtained (individually or in batches), etc. Although we implemented a simple runtime for demonstration and testing, rather than include a transformation engine within HIWAS we chose instead to produce standard artifacts that can be consumed by a variety of engines, in a runtime tailored to the user's requirements and environment.

Up to four artifacts may be produced (see step 4 of Figure 2). The first artifact (not shown in Figure 2) is an XML map file that contains a high-level specification of the mapping from source XML documents to XML documents that conform to the target model, represented in Mapping Specification Language (MSL) [21][23]. The MSL specification is used to generate the second artifact, an XSLT script or XQuery that can perform the actual transformation. The code that produces the transform from the mapping is part of another product, IBM Websphere Integration Developer (WID) [22]. Saving the map file with HIWAS is optional, but if the mapping created by HIWAS needs adjustments, WID also provides a graphical editor for this purpose. The third artifact produced by HIWAS is an XML schema (XSD) that corresponds to the target model. If the user wishes to produce relational tables, HIWAS can automatically augment this schema with directives for the DB2 Annotated Schema Decomposition Engine [16], a DB2 application for shredding XML documents into relational tables. Lastly, HIWAS can generate SQL DDL statements to create relational tables, if the user elects to produce them.

HIWAS is implemented as a set of Eclipse [6] plugins, which can easily be integrated with other ETL, modeling and business intelligence tools, especially those that are also based on Eclipse. For example, once a target model has been designed using HIWAS, the artifacts that are produced can be used by ETL tools like IBM InfoSphere Warehouse [20] or IBM InfoSphere DataStage [19] to instantiate and populate the model (Figure 2, step 5). The structure of models built using HIWAS can be analyzed with modeling tools like InfoSphere Data Architect[18], and report generation can be done using BI tools like BIRT or IBM Cognos.

4. THE SEMANTIC DATA GUIDE

The first task in analyzing a collection of CDAs or similar XML documents is to obtain an understanding of how the documents are structured and where the information of interest is located within them. As we have noted, the XML schema for the documents gives only very general information about their structure. Simple exploratory queries using XQuery or textual search are likely to overlook documents of interest, whereas queries designed to account for every hypothetical variation are likely to be more complex than necessary, perform poorly, and be impossible to understand.

HIWAS has therefore taken a different approach. Instead of relying on the schema, our solution presents the user with a summary of the structural variants that actually occur in the document collection being investigated. In addition, we leverage the template identifiers and other values whose meaning is defined by external terminologies to provide more meaningful

labeling and grouping of information. These two approaches are embodied in a data profiling structure called a Semantic Data Guide (SDG), an extension of the Data Guide structure proposed by Goldman and Widom [9] as a tool for understanding semi-structured databases. The SDG is constructed by parsing the documents in the repository to be analyzed (or a representative sample) while building a summary of the variations encountered and collecting various statistics. The Semantic Data Guide can be incrementally updated, to accommodate new types of documents.

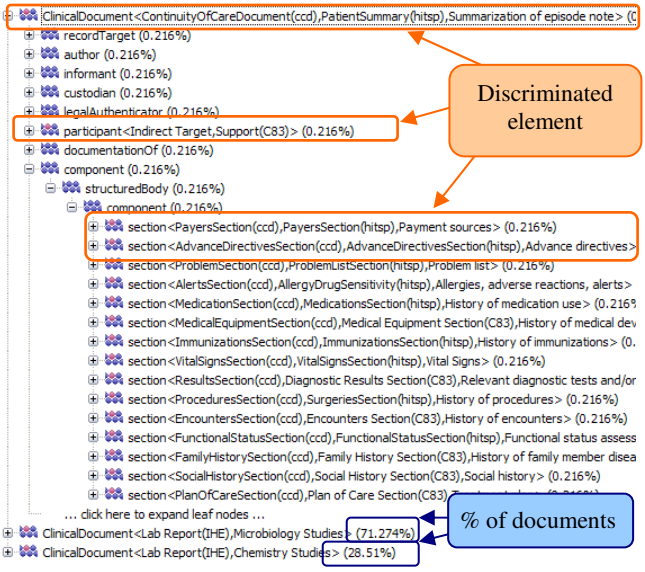

Figure 3 An example SDG.

The SDG is structured as a tree, in which each node represents the actual occurrence, in one or more documents, of a particular element in a particular context. Figure 3 shows an example of an SDG in the Navigator view of our HIWAS tool. At the top level, it lists the different types of CDA documents found in the collection, all of which have a root element named `ClinicalDocument`. Under the root element, there are other header elements, such as `recordTarget`, `author`, `custodian`, etc., as well as the element `structuredBody` which contains the actual clinical data. The structure of the SDG resembles the structure of the XML schema for CDA documents, but differs in several ways.

Firstly, in addition to the original XML element label (e.g. `ClinicalDocument`), some elements have other labels, such as "ContinuityOfCareDocument(ccd)", "Patient Summary(hitsp)" and "Summarization of episode note". We separate elements with the same generic name based on the values of elements (e.g. `templateId`, `code`) contained within them that provide additional information about what the element is intended to represent. We call these distinguishing elements *discriminators*, and describe them in greater detail below. In this example, the first element of the SDG represents `ClinicalDocument` instances whose structure is constrained by a pair of document-level templates, namely the "Continuity of Care Document (CCD)" template from HL7 and the "Patient Summary" template from HITSP's C32 standard[15], and this information is added to the node's label in the SDG. Furthermore, a top-level LOINC code indicates that this document is a "Summarization of Episode Note" and this information is also included in the node label. Another example is provided by the `section` elements contained within the `component` element of the `structuredBody`. Instead of a common generic "section" label for all of these elements, the SDG uses discriminators contained within each section to label each kind appropriately, e.g. as "Payers Section" vs. "Vital Sign Section". We refer to an element whose name includes additional information of this kind as a *discriminated element*.

Secondly, the SDG only shows what is actually present in the documents. For example, the SDG indicates that "authenticator" and "authorization" elements are present in Microbiology Studies documents, but not in Summarization of episode notes, even though the CDA schema allows them in both contexts.

Thirdly, the SDG includes various statistics to help the user understand the document collection. For each node in the SDG, available statistics include the percentage of documents in the collection that include the (discriminated) element represented by the node, the node's cardinality (the maximum and minimum number of times the designated element occurs under its parent), and the node's arity (the maximum and minimum numbers of child elements observed for the designated element), as well as coexistence ratios between elements.

Lastly, each SDG node, is associated with a small set of sample documents. The user can easily refer to an actual document to examine how data is stored in an element.

Collectively, the information in the SDG gives the HIWAS user considerable insight into the document collection. S/he can better understand the purpose of particular elements thanks to improved labeling, discover which structural variants actually exist in the collection, discover which variants are common and which are outliers, and understand how and when various constructs are used together. If more documents are subsequently added to the collection, the SDG can be updated incrementally, and HIWAS will identify newly-created nodes so that they can be considered for inclusion in the target model.

4.1 Discriminator configuration

As we noted above, we use discriminators to differentiate generic XML schema elements and give them more meaningful labels. The examples showed how template identifiers and codes could be used for this purpose, but CDAs contain numerous additional values that can be used as discriminators. Examples include the `typeCode` attribute of the `participant` element (which distinguishes different types of participants, e.g. "Referring Physician" vs. "Indirect Target"[2]) and the `use` attribute of the `addr` element which differentiates things like Permanent Home Address, Office Address, etc. Value sets for these attributes are part of the HL7 vocabulary [13].

Rather than "wire in" a predefined set of discriminators, HIWAS provides a configuration mechanism that allows the user to define an extensible set of context-sensitive discriminators. Although HIWAS can be used with any XML standard, we provide a default configuration tailored for CDA documents.

[2] In HL7-speak, an "indirect target" is a participant not present during an act or not affected by it, but related to the patient in some way, for example, as an emergency contact.

Below is an excerpt from a HIWAS discriminator configuration file that illustrates a few ways in which discriminators may be defined.

```
<discriminator id="1"> ------------------ (1)
  <name>participant</name>
  <attribute type="identifier">
       typeCode
  </attribute>
</discriminator>
<discriminator id="2"> ------------------ (2)
 <child>
   <name>code</name>
   <attribute type="identifier">
       Code
    </attribute>
    <attribute type="identifier">
       codeSystem
   </attribute>
 </child>
</discriminator>
<discriminator id="3"> ------------------ (3)
 <name>ClinicalDocument</name>
 <child>
    <name>templateId</name>
    <attribute type="identifier">
       Root
    </attribute>
 </child>
</discriminator>
```

The first rule, (1), states that in any XML element named `participant`, the attribute named `typeCode` should be used as a discriminator. The result will be that `participant` elements with different `typeCode` values will be treated as separate elements in the SDG.

The second rule differs from the first one in that it applies not to an element with a specific name, but rather to any XML element which has a child element named `code` that contains both `code` and `codeSystem` attributes. Any element containing a `code` will be split into variants based on the joint values of `code` and `codeSystem`.

The last rule demonstrates how both these approaches can be combined. In this case, the value of the `root` attribute of a `templateId` element is used as a discriminator, but only if the `templateId` element is the child of a `ClinicalDocument` element. Although not shown in these examples, one can also combine values from multiple child elements to define a discriminator. In this case, the element will not be discriminated unless it has all the specified child elements. It is also possible to specify that a discriminator be disabled in specific contexts.

Discriminator values are used not only to differentiate elements, but also to label nodes in the SDG. The discriminator value itself can be used in the label, or it can be used as a lookup key for a display name defined elsewhere. For example, if a `ClinicalDocument` element contains a `templateId` element with a `root` attribute value of "2.16.840.1.113883.10.20.1", this value is looked up in a table, yielding the display name "ContinuityOfCareDocument(ccd)" for use in labels. Currently, the lookup table is also stored as a configuration file, but we expect terminology servers for many standard value sets to be available in the future, including code systems like LOINC and template sets defined by various organizations.

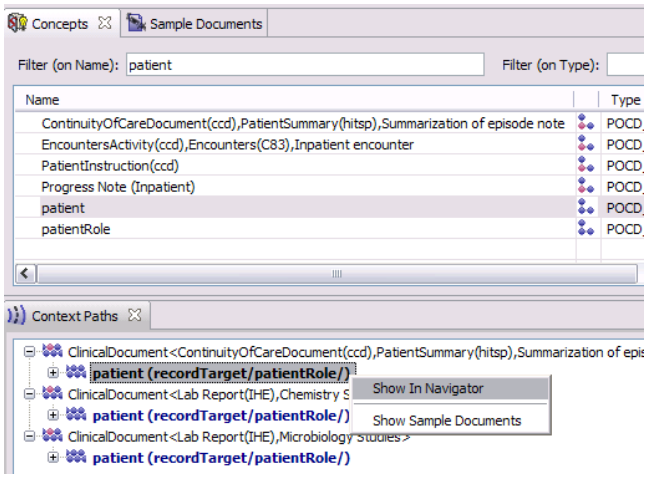

Figure 4 Other views of the SDG.

4.2 Other views of the SDG

Apart from the tree-based navigator view shown in Figure 3, the HIWAS tool also provides other views and functionality to facilitate searching for information in the SDG. These include a Concept view, which flattens all the elements in the SDG into a simple list that can be searched by (discriminated) element name or type. Since the same concept can be associated with multiple nodes in the SDG, HIWAS also provides a Context Paths view that gives a concise summary of all the locations where a concept is found, and allows the user to compare the structures associated with the concept in each location. An example, shown in Figure 4, is a `patient` element which is part of the CDA header information and is found in all three types of documents in the repository.

From the context path view, a user can easily switch back to the navigator to examine the global picture. Lastly, in addition to textual concept search, a HIWAS user can also find concepts that are potentially related because they share a common subset of discriminator values.

5. CREATING THE TARGET MODEL

The Semantic Data Guide allows a HIWAS user to explore the available collection of documents to determine what information is available and where in the documents it can be found. The next step toward making the data available for analysis is to identify the data elements that are needed for a particular purpose, and to define a simpler representation for this restricted set of data that is easily consumed by standard business intelligence tools.

The HIWAS target model editor allows the user to incrementally construct a hierarchical target model that can be realized as transformed XML documents or as a set of relational tables. The target model editor uses a drag-and-drop paradigm to construct the target model. A data element can be selected from any of the SDG views described above. The selected element is then dragged to the target model and inserted at a specific location, i.e. as a new child for an existing node.

By selecting a data element from the SDG and dragging it into the target model, the user indicates that when an instance of the XML subtree represented by the SDG node is found in a source document, it is to be copied and inserted into the target document

at the indicated location within an existing subtree, typically one that was created by copying another subtree from source to target. When there is at most one occurrence of the source subtree in the source document, and the target location can occur at most once in the target document, there is no ambiguity. However, the source subtree can occur multiple times (if its root element, or an ancestor of its root element, has been observed to occur more than once in a source document), as can the target location (if it is embedded in a subtree that can occur more than once). In these cases, a rule is needed to determine which source subtree instance(s) should be copied to which target subtree instance(s).

In HIWAS, we use the hierarchical nature of the documents as a heuristic for matching subtrees. When a new SDG node is inserted below a target model node, HIWAS determines the SDG node that corresponds to the target node (which was created, explicitly or implicitly, by a previous insertion). HIWAS then attempts to determine the minimal common subtree of the SDG that includes both the new and existing target elements. When transforming a source document, HIWAS will copy into each existing target subtree all instances of the new subtree that share the same common minimal subtree in the source document.

For example, consider a set of laboratory report documents that each contains one or more `act` elements, each act associated with a different group of tests (Hematology, Microbiology, etc.) all performed on a single specimen. Within each act is one or more observations, each of which corresponds to the results of a specific test performed on the specimen. As illustrated in Figure 5, the SDG will contain nodes for each kind of act, and beneath each act there will be a node for the specimen, and one for each test result.

Suppose a user drags the SDG node representing a Rabies Test observation from the Microbiology Studies act into the target model, as a child of the root node. HIWAS interprets this as a request to include all subtrees that represent Rabies test results in the target document. Now, suppose the user also drags the "specimen" node from the Microbiology act to the target model, and inserts it as a child of the previously inserted Rabies Test observation node. HIWAS will interpret this as a request to copy specimen information from the minimal common subtree shared by the specimen node and the test node into the test result in the target model. In this example, the minimal common subtree shared by both nodes has as its root the "act" node that contains both the specimen and Rabies Test result information, so information about the specimen for the act will be copied to the result structure for the rabies test in which the specimen was used.

As a further example, suppose the document element associated with a Microbiology Studies act also included a list of technicians involved in the test. If the `technicians` node was also inserted into the Rabies Test observation subtree, the information about all the technicians associated with the act would be added to the test result subtree.

Because the SDG is a summary of many documents, the minimal common subtree containing two nodes in the SDG is not necessarily the minimal common subtree for those two nodes in any specific document in which they appear. For example, the root node of the common subtree in the SDG may occur multiple times in source documents, but both nodes of interest may never occur as descendants of any single instance of the apparent common root. In this case, the actual root of the minimal common subtree occurs farther up the hierarchy, at some ancestor of the apparent root. The HIWAS SDG keeps track of enough extra information to detect this in certain special cases that occur commonly.

In addition to insertion of data elements, the target model editor supports several additional operations for customizing the target model. By default, target model nodes are named using the discriminated element name of their source, but the user can rename any node in the target model. One can also create initially-empty nodes in the target model, below which subtrees from unrelated parts of the source document can be grouped. The root of the target model is such an initially-empty node. Lastly, the user can delete unwanted subtrees from the target model, which allows a complex structure to be inserted into the target and subsequently pruned.

Once the target model has been completed, an XLST script or XQuery can be generated by the tool to transform the original CDA document to the XML document conforming to this model.

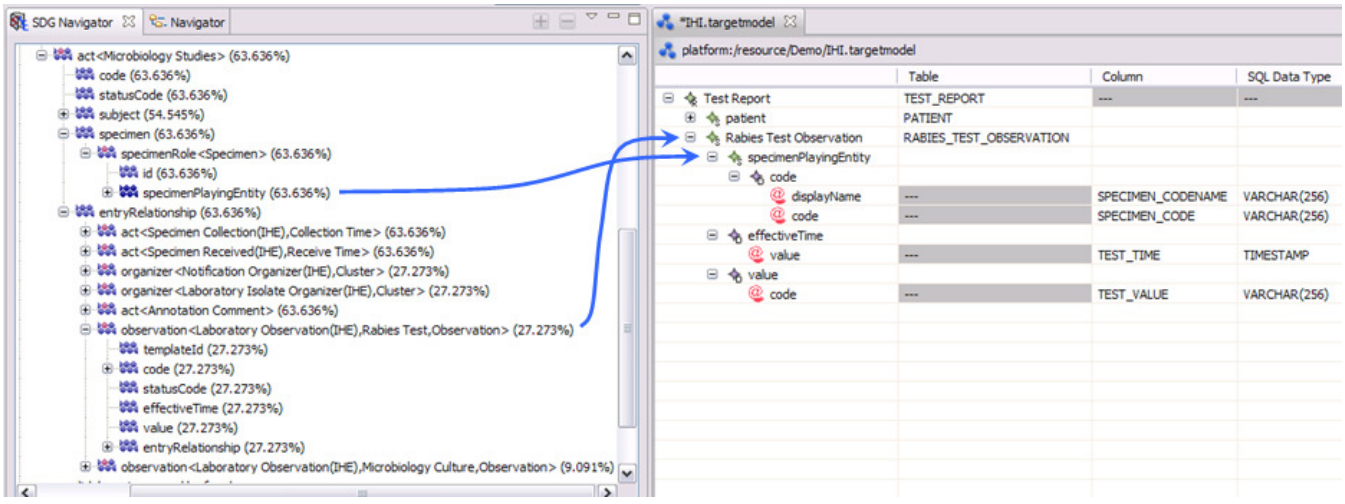

Figure 5 The HIWAS Target Model Editor.

5.1 Creating a Relational Model

If, as is most often the case, a relational target is desired, the user can further map the data elements of the target model to a set of relational tables. The default relational mapping generated by HIWAS is a set of tables that mirrors the hierarchy in the target model. In general, each element in the target model that can repeat gives rise to a table in the relational model, and non-repeating elements supply the column values, as shown in the last three columns of Figure 5. Information about actual element cardinalities from the SDG is used to eliminate unnecessary tables, i.e. those that correspond to elements in the target model that can occur multiple times in theory (according to the schema) but actually occur only once in practice. Keys are generated automatically to link tuples in a child table to the correct parent.

Users can modify the default relational mapping in various ways. Tables and columns, whose default names are derived to ensure uniqueness more than readability, can be renamed as appropriate. HIWAS attempts to guess the correct data type for each column based on the type of the corresponding XML element, but because the HL7 data types used in CDA make very limited use of XML types, the default type will usually be a character string. The user can select an alternate type in the editor, and if necessary, supply the name of an SQL function for conversion.

A user can also store portions of the CDA as XML, taking advantage of support for hybrid relational-XML databases. Designating the type of a model element as "XML" in the editor causes the subtree rooted at the element to be stored as an XML column in the appropriate table. In the same model, particular elements within the subtree (e.g. ones that need to be joined with other data) can be surfaced as regular relational columns. This approach allows the full XML context of a piece of information to be preserved for reference, while exposing its key elements in relational form to facilitate querying.

6. AN EXAMPLE: PUBLIC HEALTH REPORTS

In the previous sections, we have tried to demonstrate the need for a tool like HIWAS, and to describe in general terms how HIWAS is used to design a warehouse. In this section, we consider a specific example in greater detail.

Public Health laboratories have a need to exchange information about outbreaks of communicable diseases, often across administrative or political boundaries. A solution developed for this purpose by IBM, the Public Health Affinity Domain [3], uses documents based on the XD-LAB specification [24] to exchange information about which strains of various pathogens have been detected, drug resistance, and so forth. XD-LAB is a general-purpose CDA-based standard for laboratory test results, developed by an organization called Integrating the Healthcare Enterprise (IHE). XD-LAB constrains the basic CDA model using the techniques described in Section 2, i.e. by defining a set of templates and other coding standards that specify in detail how specific kinds of laboratory results should be represented. A few additional templates were defined specifically for public health.

An XD-LAB document for public health is structured as follows. The header, which is similar across all types of CDA-compliant documents, contains patient demographic information, information about the laboratory that performed the tests, the ordering physician, etc. The body of the document contains one or more laboratory specialty sections, depending on the test or tests that were performed (Microbiology Studies, Chemistry Studies, Urinalysis, etc.). Within each section are report items, whose format also varies depending on the kind of tests conducted. An item can represent an individual test, a battery of tests (test panel) or a complete study. A section also contains information about the specimen on which the tests were performed (blood, urine, food, etc.), including when the specimen was collected and received. Furthermore, if a communicable disease was identified, a special Notifiable Condition (or Notification of Disease) entry is created with case and outbreak information, as applicable. The important thing to note is that the document structure is highly variable, depending on the nature of the tests conducted for each pathogen, the test subject (human, animal, or food), and the test outcome (positive vs. negative).

6.1 Documenting an Influenza Outbreak

Our purpose in studying these documents was to experience the complexity associated with the various tasks involved in identifying and transforming clinical data for OLAP analysis. For the purposes of this paper, we focus on Influenza test reports in the Microbiology Studies section. Our specific goal was to create a warehouse model for studying the distribution of influenza cases among different regions and populations.

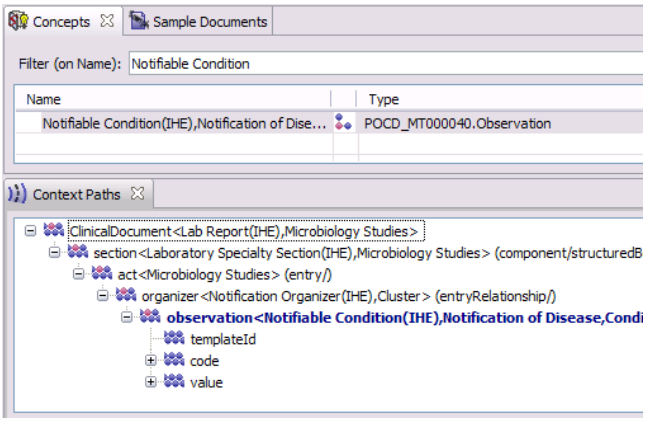

Figure 6 A Notifiable Condition in the Concepts view.

We began by using the Spatiotemporal Epidemic Modeling (STEM) system [7] to generate a set of documents representing the stream of information that a public health laboratory might receive during an influenza outbreak. From these documents we generated a Semantic Data Guide and used it to locate the specific information needed for the Influenza report. We found the patient demographic information by entering the search keyword "patient" in the tool's Concepts view, and then added all the demographic information to the target model by dragging the `patient` node from the SDG and dropping it in the target model. The Notifiable Condition observation was added to the target model in the same way (see Figure 6). Next, we added information about the specimen collection and reception times to the Notifiable Condition by dragging the corresponding nodes from the SDG and inserting them under Notifiable Condition in the target model. The final step was deletion of unnecessary items like `templateId`, `classCode`, `moodCode`, etc.

The completed target model (Figure 7) contains six pieces of information: patient demographics (gender, birth time and home

zip code) and positive test result information (specimen reception and collection time and virus type). We chose the default SQL mapping to materialize the selected data in a relational database.

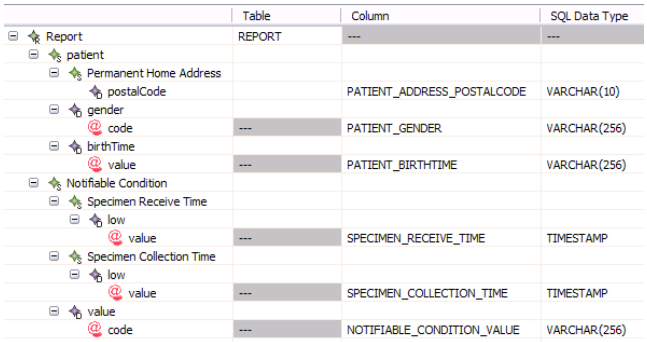

Figure 7 The model for the Influenza report.

The final step was to generate the artifacts (DDL, XSLT and Annotated XSD Schema) needed to instantiate the target model, deploy them in a suitable runtime system, and execute them to create and populate the relational warehouse.

To support OLAP analysis, we also populated the warehouse with a set of dimension tables, such as location, gender, age group, virus hierarchy and time. The star schema for the warehouse database is shown in Figure 8.

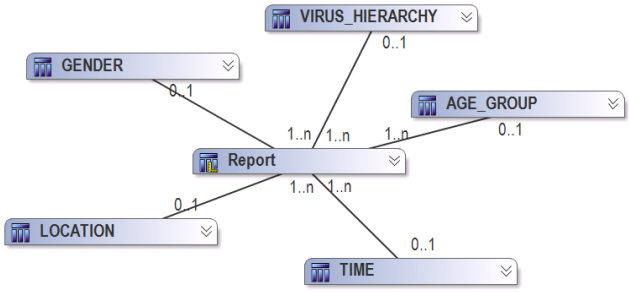

Figure 8 Warehouse database tables.

After populating the warehouse, we used Cognos BI Server to generate reports like those shown in Figure 9, to better understand disease propagation and distribution under the simulated scenario.

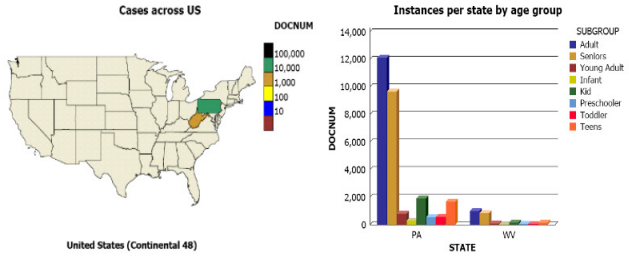

Figure 9 BI reports generated by Cognos.

6.2 Benefits of HIWAS

The exercise we described above verified the benefits of using the HIWAS tool. Firstly, the Semantic Data Guide enabled users of the tool to avoid parsing and understanding the CDA schema. The documents in the input collection used only a small subset of the elements and types defined in the schema. A typical influenza test report has around 105 unique elements (elements with distinct types) and 88 attributes, which represent 11% and 18% of the elements and attributes defined by the schema, respectively. Other public health documents are similar in this regard, as can be seen from the first five rows of Table 2.

Table 2 Statistics for CDA documents.

Document type	% Elements	% Attributes	Size
Influenza	11.0%	17.9%	18K
HIV	10.5%	14.6%	16K
Lead Poisoning	11.1%	18.7%	16K
Salmonella Poisoning	11.7%	20.7%	56K
E. Coli	11.1%	17.7%	43K
Continuity of Care Document	13.4%	29.1%	190K

The last row of Table 2 contains analogous numbers for a typical Continuity of Care Document, a summary of patient records for the purpose of discharge and care transfer that contains information like vital signs, problem diagnoses, allergies, medical history, social history, medications, etc. These documents contain much more information than public health documents, and are correspondingly larger, but the fraction of schema elements and attributes used is just slightly higher, with around 59% of elements overlapping with those used in the public health documents. The advantage of using the Semantic Data Guide, which is driven by the document instances rather than the schema, is clear. Furthermore. although the number of constructs present in the documents is small compared to the schema, the selection of elements used for our influenza report is even smaller. Only 6 elements were selected for the target model, which represents less than 6% of the elements in typical public health documents.

Secondly, the discriminated element names used to label nodes in the Semantic Data Guide were much more helpful than the generic element names used in the documents. Using our discriminator configuration, the tool was able to identify 19 discriminated elements in the influenza documents. A few examples are given in Table 3. With discriminated element names, users can identify elements of interest without knowing the particular code or template identifier that distinguishes one generic item from another.

Table 3 Discriminated elements in the Influenza report.

Discriminated Name	XML Element Name	Discriminators
Referring Physician	participant	typeCode="REF"
Performer	participant	typeCode="PRF"
Annotation Comment	act	code="48767-8" codeSystem="2.16.840.1.113883.6.1" templateId="1.3.6.1.4.1.19376.1.5.3.1.4.2"
Specimen Collection Time	act	code="33882-2" codeSystem="2.16.840.1.113883.6.1" templateId="1.3.6.1.4.1.19376.1.3.1.2"
Laboratory Observation, Influenza Serotype	observation	code="20951-0" codeSystem="2.16.840.1.113883.6.1" templateId="1.3.6.1.4.1.19376.1.3.1.6"

Discriminated Name	XML Element Name	Discriminators
Notifiable Condition, Notification of Disease	observation	code="170516003" codeSystem="2.16.840.1.113883.6.96" templateId="1.3.6.1.4.1.19376.1.3.1.1.1"

Thirdly, the XML-to-XML transformations needed to populate the target model were automatically constructed by the HIWAS tool with much less user interaction than conventional tools would require. The need to construct a complete target schema in advance was eliminated, and each user gesture added an entire subtree to the target model, instead of requiring the user to create element-by-element mappings between the source schema and the target schema at each level. Moreover, HIWAS added the correct conditions to each mapping so that only the desired elements were selected. For example, in the case of the specimen collection time, 11 conjunctive conditions were specified in the XSLT transformation to identify the specific act element containing this information. Three conditions (code, codeSystem and templateId) were required to select Microbiology Studies documents, 3 more to select Laboratory Specialty Sections within those documents, 2 conditions were needed to select Microbiology Studies acts, and 3 conditions were used to select the Specimen Collection Time act.

Finally, the study illustrates how different structures can be used to represent similar information, even within the same kind of document. For example, the location of the Microbiology Culture laboratory observation in the influenza test document is different when the test result is positive than when it is negative. Figure 10 shows the two paths that correspond to these alternatives.

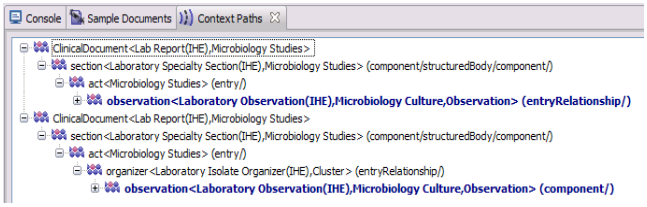

Figure 10 Multiple context paths.

The first laboratory observation is for a negative test result, and just shows a negative finding of influenza. The second path is for a positive result, in which the test result is part of an Isolate Organizer. The organizer represents an additional microbiology study performed on isolates derived from the original specimen, in order to identify the particular strain of influenza. HIWAS can help users to identify and examine these differences, and select the correct entities for the target model.

6.3 Other lessons learned

Our study of public health documents also revealed some requirements that we had not initially anticipated. Type conversions are a case in point. For simple types derived from the built-in types of XML, HIWAS will generate an XML-to-SQL type conversion based on the built in type and add it to the schema annotation that describes how to generate the corresponding column value. However, there are several simple types defined by HL7 that derive from XML's string datatype but should be mapped to a different datatype in SQL. The simple type `ts` is a typical example. Intended to represent a timestamp, its definition restricts the element value to strings with a particular format defined by HL7 that does not conform to any of the standard string representations of an SQL timestamp. To accommodate such types, we added a feature to HIWAS that allows users to provide user-defined functions (SQL UDFs) to handle type conversions.

Another problem we discovered is that some values that make good discriminators in one context are not helpful in others. For example, codes are often good candidates for discriminators, but not always. Although it is useful to use an associated code to distinguish an act element that represents an Annotation Comment from one that represents a Microbiology Study, in other cases, codes are used as a source of values for describing an entity. For example, the code in the element `specimenPlayingEntity` describes the type of specimen being tested (stool sample, blood, brain tissue, etc.). The structure and meaning of the element remains the same regardless of the code value. Therefore, if we use code as a discriminator for elements of this type, a different SDG node will be created for each kind of specimen. We call this *over-discrimination*, and to avoid it we allow an exclusion list to be specified as part of the definition of a discriminator. In the public health document study, there were about 10 element contexts for which a specific discriminator was excluded.

7. CONCLUSIONS AND FUTURE WORK

As nations and technology providers around the world move towards adopting HL7 CDA as the standard for healthcare information exchange, one needs to concurrently acknowledge the challenges in extracting the valuable clinical information these documents contain, especially for purposes beyond immediate patient care. The prevalence of relational database systems and the dearth of XML tools that can handle such complex documents represent gaps on the path towards meaningful use of standard-compliant clinical data. HIWAS bridges these gaps in several ways. Through the use of the Semantic Data Guide, HIWAS enables investigators to understand the structure of the documents available to them and find information of interest quickly. The data mapping technology that underlies the target model editor simplifies the task of transforming the data to a form that can be analyzed with existing tools.

While the current version of HIWAS is a useful tool, our experience performing various mapping exercises has revealed several areas where it could be improved or extended. Firstly, the hierarchical structure of the target model may not reflect the primary/foreign key relationships that are ultimately desired. The target model is inherently document-centric, whereas a more patient-centric perspective may be desired in the warehouse. Currently, we use manually-created SQL views to fine-tune the relational schema produced by HIWAS, but a more integrated approach would be less laborious.

Secondly, although our current strategy for producing a relational representation of clinical data is to materialize it in a warehouse, one could also envision a "virtual warehouse" in the form of SQL/XML views over the original data, an approach that may become more attractive as XML database technology matures. We have recently extended HIWAS with the capability to create such views, allowing users to choose the transformation technology best suited to their requirements.

We have also encountered situations in which the target schema is known *a priori*. We view HIWAS' ability to work without a pre-

existing target as a differentiator from existing schema mapping tools and an advantage in many contexts, but there would also be value in developing a hybrid mapping tool combining HIWAS' ability to locate information using the SDG with the schema-matching and map-generation capabilities of existing tools.

Expanding the SDG to include information about content as well as structure is also an interesting possibility. For example, statistical information about the distribution of values in a field might provide users with additional insight into their data.

Another potential area for extension concerns the configuration of discriminators. We have mentioned how a tighter integration between HIWAS and a terminology server could reduce the need for configuration files. A more ambitious goal would be to produce discriminator configurations directly from the specification of the standard. The process of creating and curating standards based on CDA is evolving toward the use of mainstream modeling tools like UML, which should enable automated processing and analysis of the specifications in ways that are difficult with today's natural-language representation.

We have also already mentioned the problem of over-discrimination, which we currently handle by statically defining exclusion rules in the discriminator configuration. However, the decision whether or not to consider a code or some other value as a discriminator in a certain context ultimately depends on the purpose of the warehouse. Rather than change the configuration and rebuild the SDG for each use case, we would prefer to give users the option of turning discriminators on or off dynamically as they explore the data and build their target model.

Lastly, we note that while our model for specifying discriminators works well with the CDA standard, and certain standards used in other industries, e.g. RIXML [29], there are standards like XBRL [31] that use different conventions to specify the semantic relationships among document elements and to constrain document structure. It would be interesting to extend HIWAS to support such standards as well.

8. REFERENCES

[1] Altova Inc., Altova MapForce, http://www.altova.com/mapforce.html

[2] Beyer, Kevin S. et al, XQuery for Analytics: Challenges and Requirements. In *Proc. 1st Int'l Wkshp. on XQuery Implementation, Experience and Perspectives (XIME-P)*:3-8, Paris, 2004.

[3] Carmeli, Boaz et al, "Public Health Affinity Domain: A Standards-Based Surveillance System Solution", *BioSurveillance*:147-158, 2007.

[4] Dolin R. H. et al, HL7 Clinical Document Architecture, Rel 2. JAMIA 13:30-39, 2006.

[5] Eclipse Foundation, Business Intelligence and Reporting Tools, http://www.eclipse.org/birt/

[6] Eclipse Foundation, Eclipse, http://www.eclipse.org

[7] Eclipse Foundation, The Spatiotemporal Epidemic Modeling (STEM) Project, http://www.eclipse.org/stem

[8] Eisenberg, A. and Melton, J.: Advancements in SQL/XML. *ACM SIGMOD Record 33, 3*: 79-86, 2004.

[9] Goldman, R. and Widom, J. DataGuides: Enabling query formulation and optimization in semi-structured databases. In *Proc. 23rd VLDB*: 436–445 Athens, August 1997.

[10] Haas, L. et al, Clio grows up: from research prototype to industrial tool. In *Proc. 24th ACM SIGMOD Conf.*:805-810, Baltimore, 2005.

[11] Health Level Seven, HL7 V3 Messaging Standard, http://www.hl7.org/implement/standards/v3messages.cfm

[12] Health Level Seven, HL7 V3 Reference Information Model, http://www.hl7.org/v3ballot/html/infrastructure/rim/rim.html

[13] Health Level Seven, HL7 Vocabulary Domains, https://www.hl7.org/library/data-model/RIM/C30202/vocabulary.htm

[14] Health Level Seven International, http://www.hl7.org/

[15] HITSP, C32 Summary Documents Using HL7 CCD, http://www.hitsp.org/ConstructSet_Details.aspx?&PrefixAlpha=4&PrefixNumeric=32

[16] IBM Corp., DB2 Annotated Schema Decomposition Engine, http://www.ibm.com/developerworks/data/library/techarticle/dm-0604pradhan2/

[17] IBM Corp., IBM Cognos. http://www-01.ibm.com/software/data/cognos/

[18] IBM Corp., IBM InfoSphere Data Architect, http://www-01.ibm.com/software/data/optim/data-architect/

[19] IBM Corp., IBM InfoSphere DataStage. http://www-01.ibm.com/software/data/infosphere/datastage/

[20] IBM Corp., IBM InfoSphere Warehouse, http://www-01.ibm.com/software/data/infosphere/warehouse/

[21] IBM Corp., IBM SPSS Statistics, http://www.spss.com/

[22] IBM Corp., IBM WebSphere Integration Developer, http://www-01.ibm.com/software/integration/wid/

[23] IBM Corp., Mapping Specification Language, http://www.ibm.com/developerworks/websphere/library/techarticles/1003_spriet1/1003_spriet1.html

[24] IHE International, XD-LAB Content Module, http://www.ihe.net/Technical_Framework/upload/ihe_lab_TF_rel2_1-Vol-3_FT_2008-08-08.pdf

[25] International Health Standards Development Organization, SNOMED-CT, http://www.ihtsdo.org/snomed-ct/

[26] LOINC.org, Logical Observation Identifiers Names and Codes, http://loinc.org/

[27] Oracle Corp., Oracle Warehouse Builder, http://www.oracle.com/technetwork/developer-tools/warehouse/overview/index.html

[28] Progress Software Corp., Stylus Studio 2011 XML, http://www.stylusstudio.com/xml_mapper.html

[29] RIXML.org, Research Information Markup Language,. http://www.rixml.org/

[30] SAS Institute Inc, http://www.sas.com/

[31] XBRL International, Extensible Business Reporting Language. http://www.xbrl.org