SemanticDB: A Semantic Web Infrastructure for Clinical Research and Quality Reporting

Christopher D. Pierce*[,1], David Booth[2], Chimezie Ogbuji[3], Chris Deaton[4], Eugene Blackstone[1] and Doug Lenat[4]

[1]*Cleveland Clinic, Cleveland, OH, USA;* [2]*PanGenX, Auburndale, MA, USA;* [3]*Case Western Reserve University, Cleveland, OH, USA;* [4]*Cycorp, Austin, TX, USA*

Abstract: Semantic Web technologies offer the potential to revolutionize management of health care data by increasing interoperability and reusability while reducing the need for redundant data collection and storage. From 1998 through 2010, Cleveland Clinic sponsored a project designed to explore and develop this potential. The product of this effort, SemanticDB, is a suite of software tools and knowledge resources built to facilitate the collection, storage and use of the diverse data needed to conduct clinical research and health care quality reporting. SemanticDB consists of three main components: 1) a content repository driven by a meta-model that facilitates collection and integration of data in an XML format and automatically converts the data to RDF; 2) an inference-mediated, natural language query interface designed to identify patients who meet complex inclusion and exclusion criteria; and 3) a data production pipeline that uses inference to generate customized views of the repository content for statistical analysis and reporting. Since 2008, this system has been used by the Cleveland Clinic's Heart and Vascular Institute to support numerous clinical investigations, and in 2009 Cleveland Clinic was certified to submit data produced in this manner to national quality monitoring databases sponsored by the Society of Thoracic Surgeons and the American College of Cardiology.

Keywords: Clinical data, clinical research, electronic medical records, inference, ontology, quality reporting, RDF, semantic web

1. INTRODUCTION

Demand for medical data to support clinical research and mandated health care quality reporting has increased dramatically. This has led to a proliferation of registries, academic databases, and *ad hoc* subject-specific databases to manage patient medical data as multiple hospital departments and sections attempt to meet these demands. Even though these registries are designed to capture data about different treatments or diseases, they frequently are redundant, containing data on some of the same patients and sharing many of the same variables such as demographics, risk factors and outcomes. Because much of the data contained in these registries is manually abstracted from electronic or paper charts, not only are those data redundancies inefficient, but values for the same variables may differ among registries, which can undermine the reliability of research findings, erode the trust of investigators and administrators, and call into question the accuracy of quality metrics derived from them.

After several years of investigation, experimentation and proof of concept, Cleveland Clinic embarked on a project that, by 2003, focused on the use of emergent semantic web technology to address these issues [1]. Our goal was to create a data management system with sufficient flexibility and extensibility that any variable could be added to the system, data could be collected once and reused for multiple purposes, and intelligent agents could be devised for automated discovery and machine learning, particularly across different medical disciplines. We chose semantic web technology over conventional relational techniques for several reasons:

- it facilitates automated inference, which enables simpler queries with more complete results, and automated discovery by intelligent agents;
- it allows information that is expressed in disparate vocabularies to be readily integrated without degradation of meaning or quality;
- it adapts to vocabulary and data model changes more readily than conventional relational techniques; and
- it is standards-based and web-friendly.

This effort resulted in the creation of SemanticDB, a content repository and data production system, which was released for use at Cleveland Clinic in January of 2008.

SemanticDB was developed and implemented using an existing registry of heart surgery and cardiovascular intervention cases with over 500 variables, and spanning more than 30 years with almost 200,000 patient records. At the time we began our project, this registry was being used for research generating over 100 publications per year, internal administrative reporting, and external quality reporting. By using such a large and complex registry to develop our system, we were forced to deal with issues of functional range, scalability and performance that could have been overlooked with a smaller test case.

*Address correspondence to this author at the Cleveland Clinic, JJ-40, 9500 Euclid Avenue, Cleveland, OH 44195, USA; Tel: +1-216-445-8196; Fax: +1-216-636-0925; E-mail: piercec1@ccf.org

2212-392X/12 $58.00+.00

This paper is an overview of SemanticDB's architecture, data model, inference patterns, and uses in research and quality reporting. Before turning to these topics, we offer a brief introduction to some of the semantic web concepts that underlie SemanticDB.

2. BASIC CONCEPTS IN SEMANTIC WEB TECHNOLOGY

"A little bit of semantics goes a long way." -- Professor James Hendler, Rensselaer Polytechnic Institute.

Semantic web technology was designed to facilitate automated reasoning at web scale. It is based on the Resource Description Framework (RDF) [2] a W3C standard knowledge representation language that captures information as assertions or *triples*. Each assertion represents a single statement of fact and consists of three parts roughly corresponding to *subject-verb-object* of a simple sentence, or *subject-predicate-object*. A set of RDF assertions is known as an RDF graph because it can be viewed as a directed graph of arcs and nodes, as the subject or object of one assertion is used as the subject or object of other assertions. Cleveland Clinic's registry of 200,000 patient records comprises an RDF graph of roughly 80 million RDF assertions.

Inference is the process of concluding new facts or assertions from existing facts or assertions. Although any process that creates new assertions from existing assertions could be viewed as inference, RDF inference is usually performed by a carefully constructed *inference engine* or reasoner. For example, from the assertion (written in an RDF syntax called N3 [3]):

:t1 a :AorticValve .

which means that the thing called :t1 is an aortic valve, an inference engine having knowledge that an :AorticValve is a kind of :HeartValve could infer the following assertion:

:t1 a :HeartValve .

Such an inference makes sense to a human who knows anatomy, but an inference engine must be explicitly told what inferences it should perform. SemanticDB uses a set of specific inference rules and an ontology representing the domain to execute these inferences.

An inference rule is a template involving variables or placeholders that will be replaced with actual values when an inference engine matches the rule against existing RDF assertions. For example, the following rule, written in N3, indicates that every aortic valve is a heart valve:

{ ?v a :AorticValve . } => { ?v a :HeartValve . } .

In N3, the symbol "?v" (or any symbol beginning with "?") is treated as a variable, so when this rule is applied to the assertion above, :t1 will be substituted for ?v. The symbol "=>" is read as "implies". If assertions are found in the knowledge base that satisfy the left side of this rule, then the inference engine adds the assertions on the right side -- the *entailments* -- to the knowledge base. This style of inference is known as *forward chaining*.

In some cases it would be inefficient or infeasible to attempt to pre-compute all of the new facts that might theoretically be produced by a set of rules in a forward chaining mode, especially when many would be irrelevant to a given query. In such cases, an inference engine that supports *backward chaining* can be used instead. With backward chaining, inference is performed only when a query is issued: the inference engine starts from the query, treating it as a goal, and then works backward from this goal to figure out a sequence of available inference rules that would enable the existing data to support the conclusions needed to answer the desired query. Neither forward nor backward chaining is universally better than the other. Each approach has its advantages and disadvantages. Some inference engines, such as Cyc [4], support both forward and backward chaining or even a mix of the two.

The :HeartValve inference illustrated above is very simple and involves only one step of inference. Indeed, inferences need not be complex to be extremely useful. For example, this inference alone enables a query for :HeartValve operations to return operations involving :t1, even though :t1 was recorded as being an :AorticValve. However, the main value of inference is normally obtained when an inference engine applies a set of inference rules recursively, drawing conclusions that may involve long chains of inference and many rules. For example, if other inference rules indicate that a :HeartValve is part of the :Heart, and that the :Heart is located in the :Thorax, then a query for operations in the :Thorax would find those that involve :t1, through two steps of inference.

In addition to specific rules, an ontology can tell an inference engine what inferences to perform. An ontology is "a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts" [5]. For example, a medical ontology may indicate that the aortic valve is a kind of heart valve. This may be expressed formally in the N3 RDF syntax using the OWL [6] ontology language as:

:AorticValve owl :subClassOf :HeartValve .

Given this assertion, an OWL reasoner already knowing the semantics of owl:subClassOf will know to conclude that :t1 is a :HeartValve if an assertion has already indicated that :t1 is an :AorticValve. The use of ontologies therefore simplifies the task of specifying what inferences are desired.

SemanticDB relies heavily on inference to perform or facilitate several functions. Both forward and backward chaining are used, with the decision to use one or the other based on practical trade-offs between query latency performance and data storage space. This allows queries to be simpler while producing complete results, and it allows the system to operate more efficiently by intelligently avoiding the pre-computation of unnecessary intermediate data.

3. SEMANTICDB ARCHITECTURE OVERVIEW

SemanticDB was designed to support the data management requirements of both clinical research and quality-of-care reporting. This involves the flexible capture of new data elements through the generation of new data entry forms, secure storage of this data, flexible cohort identification, merging of additional data derived from multiple data sources that use diverse vocabularies, protocols and data formats, and production of customized views of the

data [7]. To achieve these functional goals, SemanticDB consists of three main subsystems: 1) a content repository; 2) a query interface tailored to the task of identifying patient cohorts, known as Semantic Research Assistant (SRA); and 3) a customized data production pipeline.

3.1. Content Repository

The content repository of SemanticDB facilitates data collection, document management, and formal knowledge representation for use in managing longitudinal clinical data on individual patients. This repository manages XML [8] patient record documents and converts them to RDF graphs as needed for downstream semantic processing [7]. Its architectural features include the ability to automatically generate XForms [9] data input screens from a meta-model describing the required data collection fields, a file-based transactional storage system, an RDF-based work flow management system for controlling the data capture processes, and XML-to-RDF data transforms using automatically generated XSLT [10] templates.

The content repository's primary architectural requirements include the automation of certain aspects of data entry, structure, storage, display, and retrieval of patient data with minimal intervention by traditional database administrators and computer programmers. It is designed to use a declarative style in which the desired objective is formally described, rather than indicating the specific steps required to achieve it. One component takes a meta-model and generates document schemas, a formal patient record ontology, data transforms, validations, and document templates. Another component takes an abstract representation of a data entry form (with references to terms in the model of the domain) and generates a user interface using XForms.

The architectural style adopted (and corresponding set of constraints) is based on the desire to automatically generate and maintain a flexible content management system built around a centralized patient record vocabulary authored by domain experts and knowledge engineers. The vocabulary incorporates a meta-model that provides a formal semantics for hierarchical XML document composition, and an abstract framework for modeling medical terms in a way that facilitates semi-automated use of native XML and RDF representations somewhat interchangeably. XSLT templates used to transform the XML patient records to RDF are also generated from the meta-model. A small set of predetermined generic rules are applied to the meta-model to produce the XSLT templates that translate the hierarchical relationships between the data elements expressed in the XML representation of the patient data to corresponding nodes in an RDF graph. The generated XSLT templates correspond to (and match) elements in the XML patient record document. Each template is responsible for producing some RDF content, in RDF/XML format, that captures statements about what the content in the XML document denotes. In short, the transformations capture the semantics of the meta-model and the intuition that the XML vocabulary encodes the RDF meaning of the domain.

Given a workflow that starts with the meta-model, produces its data management artifacts (e.g., the XSLT templates), and generates an RDF dataset for a given SemanticDB instance, structurally additive changes to the meta-model can be made asynchronously without incurring a misalignment between the data and the generated transforms and schemas. Certain leaf nodes in the input document correspond to controlled vocabularies used as "pick lists" in the data collection user interface. The collection of their allowed values are properly modeled in the generated OWL ontology, following best practices regarding the modeling of value partitions. So, when new values are added to the controlled vocabulary, the transforms and ontologies for the new meta-language can be regenerated and used to replace older artifacts and -- upon subsequent data collection -- new content is properly reflected in the RDF dataset without loss or corruption of the old data.

Over the course of the SemanticDB project, RDF assertions were saved to two different RDF stores. Initially, we used the relational schema for RDF employed by RDFLib [11] with some modifications to support the entire SPARQL [12] language for efficient storage and retrieval from a MySQL database [13]. An optimized SPARQL endpoint to this RDF store yielded one to two orders of magnitude improvement in query response time over an unoptimized endpoint [13]. This work was the basis for a doctorate at the Case Western Reserve University's Electrical Engineering and Computer Science department. In 2010, we added the Oracle 11g Spatial product [14] as another RDF store fed by SemanticDB.

Once patient records have been transformed into RDF and placed in an RDF repository, SemanticDB performs graph expansion – a process in which forward chaining inference is performed and the resulting assertions are added back into the RDF knowledge base. This allows subsequent queries to be simpler while still returning complete results, without requiring backward chaining inference that would slow the query. For example, a query for heart valve procedures would immediately include aortic valve procedures in its results. Graph expansion is currently driven by a small set of inference rules written in N3 and implemented as a production rule system using a RETE [15] decision network algorithm running in a large, symmetric multi-processing cluster. The graph expansion component relies on traditional sideways information passing strategies to optimize the firing of the rules by the production rule system with respect to a set of *a priori* queries until a fixed-point is reached. The decision network is built from a set of rules re-written with respect to the queries using the generalized magic sets (GMS) method and the individual RDF graphs for each patient record are fed through the decision network concurrently.

3.2. Query Interface

The cohort identification interface, Semantic Research Assistant (SRA), shown in Fig. (**1**), was developed in 2007 in partnership with Cycorp, Inc. It is based on the Cyc[4] inference engine, a powerful reasoning system and knowledge base with built-in capability for natural language processing, forward-chaining inference and backward-chaining inference. SRA incorporates Cyc's natural language processing to permit a user to initiate a cohort selection query by typing an English sentence or sentence fragment. Natural language is convenient for users, but

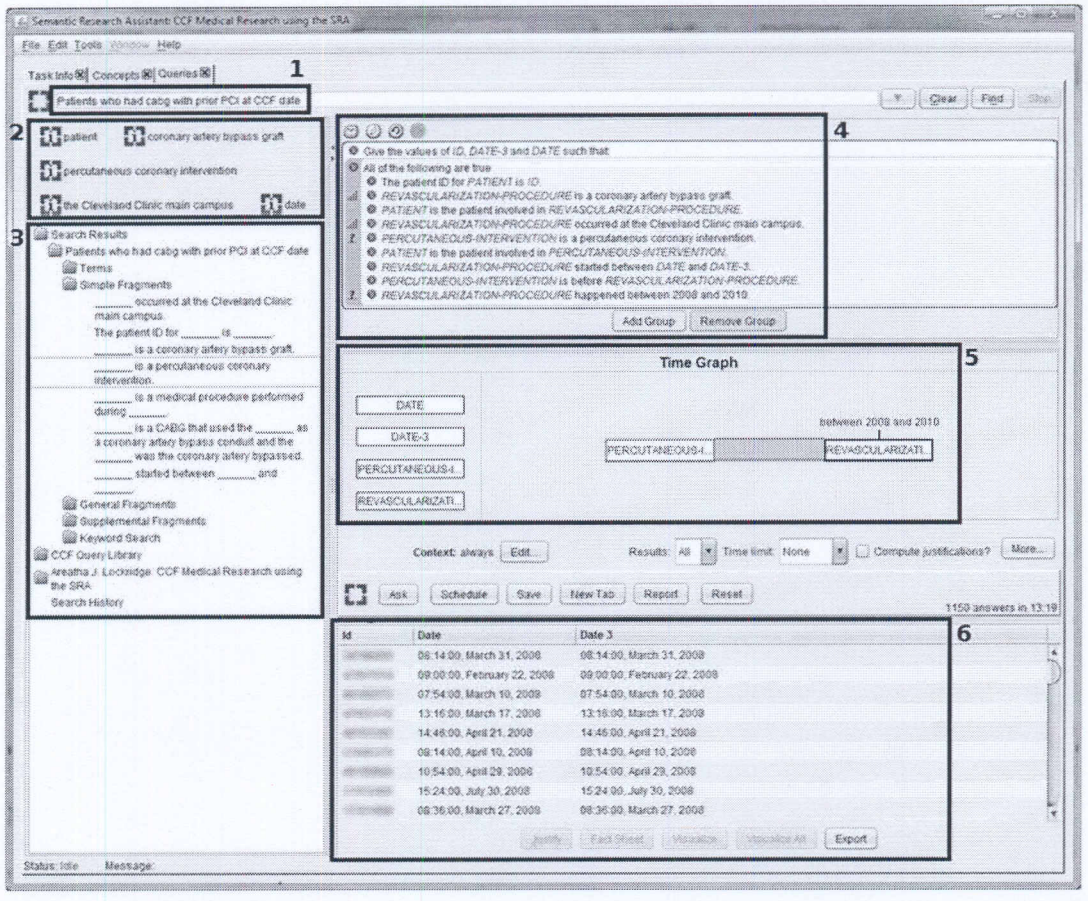

Fig. (1). The Semantic Research Assistant (SRA). This annotated screen shot shows how the SRA was used to query for patients who had a coronary artery bypass graft (CABG) between 2008 and 2010 (inclusive) and after a percutaneous coronary intervention (PCI). In box 1, the user types an English sentence fragment for the desired query: "Patients who had cabg with prior PCI at CCF date". The SRA parses this sentence, looking up specialized terms such as "cabg" and "PCI" in its knowledge base, and proposes likely interpretations for fragments of the query in box 3. The user selects desired interpretations from box 3 and drags them to box 4, where they are combined to form the full query, which is displayed in a precise, English-like dialect for the user to verify. Alternately, the user may drag commonly used concepts from box 2 to box 4, for inclusion in the query. In box 5, the user has the option of further constraining the query with specific temporal constraints that are displayed visually on a timeline. Box 6 shows a sample of the query results. (For privacy, patient IDs have been obscured.) If the query is likely to take longer than the user wishes to wait, the query can be scheduled and the user will be notified when the results are ready. When the query results are ready they can be saved or exported as a CSV cohort file for use in the next step in the research process or the data production pipeline.

notoriously ambiguous, especially when interpreted by machine. To prevent misunderstandings, SRA proposes plausible interpretations corresponding to fragments of the user's natural language query, and allows the user to select and confirm the desired interpretation. Temporal constraints are extremely important in cohort identification, but the precise logic for them is hard to interpret correctly from a natural language query. Therefore, the SRA interface also allows the user to view and express temporal constraints graphically on a timeline.

Cohort identification currently uses the most sophisticated forms of inference, both because it combines natural language processing with formal queries, and because it performs temporal reasoning. Again, Cyc's built-in backward chaining capabilities are used to retrieve only the data that is needed to answer the user's cohort identification query.

Internally, the SRA uses domain-specific medical ontologies in conjunction with the Cyc general ontology and knowledge base of real world facts [16] to convert the natural language query into a formal representation in CycL [17], which is then used to generate the appropriate SPARQL queries. The SPARQL queries are submitted to the SemanticDB RDF store for execution, and if necessary, SRA submits the query results to Cyc for further inference-based processing to arrive at the proper results. These results are returned to the SRA interface in the form of a list of cases (patient IDs and index event identifiers) as they are retrieved from the repository. Use of the SRA is illustrated in Fig. (**1**).

3.3. Data Production Pipeline

The data production pipeline takes a cohort file typically created by the SRA, selects corresponding patient records from the content repository, combines those records with additional data from other sources, such as blood tests results and echocardiogram findings, and processes the resulting

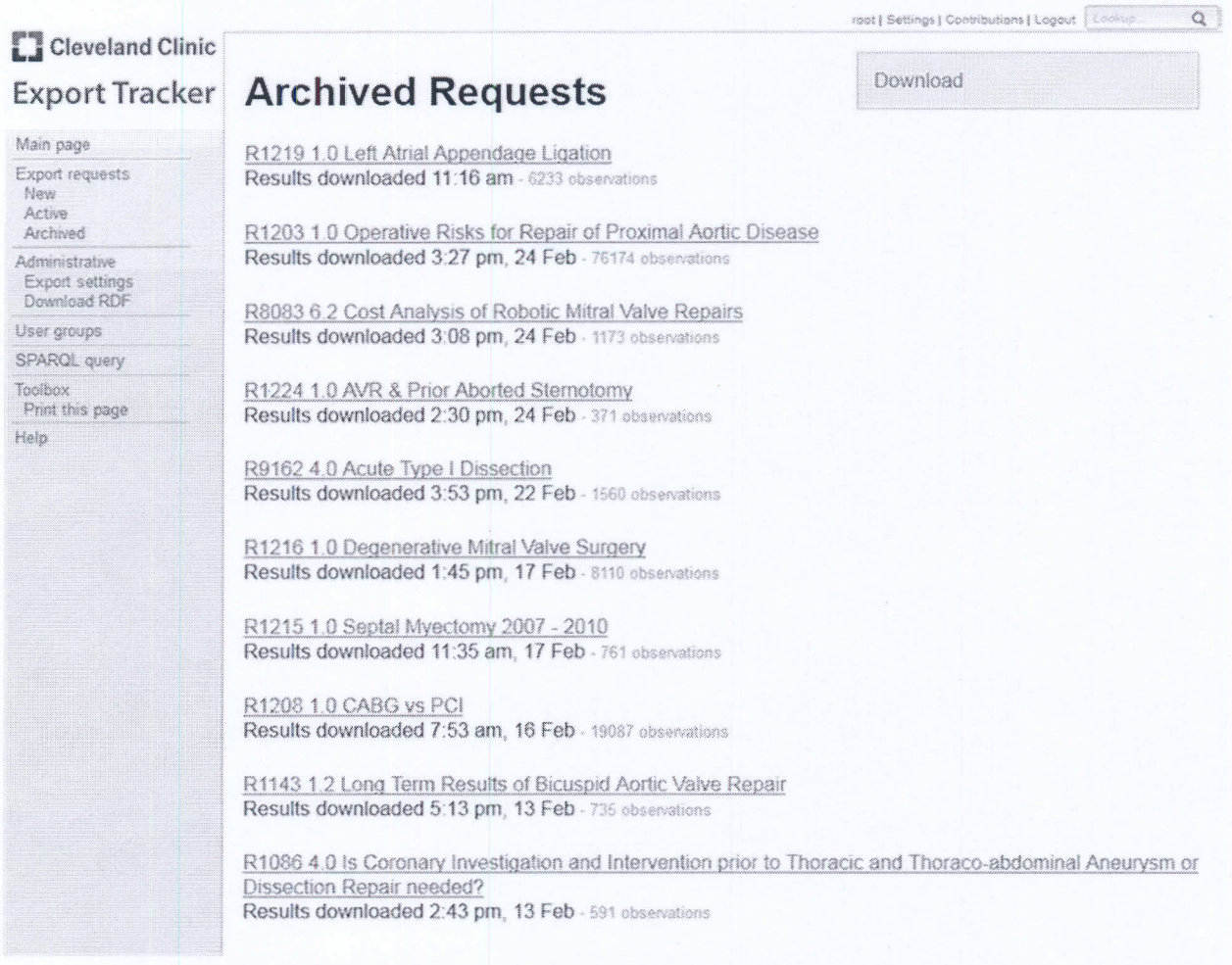

Fig. (2). Export Tracker. The Export Tracker manages a queue of pending data production requests. Each request takes a cohort file and produces a set of data modules comprising some 300 commonly needed variables ready for statistical analysis. Because this data generation is computationally intensive for cohorts involving many observations, and because researchers often wish to run the same requests multiple times as patient records in SemanticDB are updated, both the requests and the results are automatically archived for potential reuse. This screen shot shows several data requests that have been run and automatically archived.

records to generate datasets appropriate for statistical analysis or reporting. This pipeline is driven by a combination of various automation scripts, which retrieve and normalize the required data, and a web-based interface known as "Export Tracker", shown in Fig. (**2**), which manages an ongoing queue of data requests. The resulting data are also used with further inferencing to generate administrative and quality measurement reports required both internally and externally, as discussed in Section 6.

In addition to custom software, the SemanticDB components described above were constructed using software from open source projects including an XML & RDF content repository bundled with 4Suite [18] (for storing XML patient records and invoking XSLT conversions to RDF), RDFLib [11] (for processing RDF data from Python modules that generate data exports), FuXi [19] (for inference rules processing), MySQL [20] (for storing intermediate results), Simile Exhibit [21] (for browsing RDF patient records), Mozilla's Firefox with XForms extensions [22] (for patient data collection forms), and Callimachus [23] (for the "Export Tracker"), and commercial systems including Cyc [4] (for the SRA and report generation), Oracle 11g Spatial [14] (for RDF storage) and ViaDuct [24] (for data mapping).

4. PATIENT RECORD ONTOLOGY

As described above and elsewhere [25], SemanticDB employs a domain meta-model, captured in RDF, to drive much of its functionality. This meta-model is constructed using a small number of properties and classes with clearly defined semantics, which are used to automatically generate an XML schema for structured data collection, an OWL patient record ontology to serve as a schema for the data in RDF, and an XSLT template to convert data captured as XML into RDF. Its main function is to describe data elements required for collection in the repository rather than attempting to capture the full ontology of the domain. Consequently, the model focuses on the hierarchical relations present in the XML patient record, which are then used to drive XML-based data capture tools. This is accomplished by using the *dnode:contains* predicate to represent hierarchical XML relations.

For example, the snippet of the meta-model shown below describes the data to be collected for an aortic valve during a diagnostic procedure such as an echocardiogram. The class ptrec:AorticValve has common RDF schema properties including a rdfs:label, which can be used in generating human-readable views of the data, rdfs:comment, which can include a human-readable definition of the class, and rdfs:subClassOf, which specifies a "kind-of" relationship to another class. In this case, the rdfs:subClassOf property is used to specify the cardinality of the AorticValve class because it is a kind of OptionalSingleton meaning that there can be zero or one instance of this class in a given context. In addition, there are properties unique to the SemanticDB meta-model that are designated with the "dnode:" namespace. As noted above, the dnode:contains property specifies that the class is an XML parent to the class listed as the object of the property. The Boolean property dnode:index indicates whether the class should be included in the RDF graph generated from an XML instance of the model. The dnode:inheritsConstraints property specifies that the properties of its object in this case the ptrec:CardiacValveData class, are also properties of the ptrec:AorticValve class. This allows classes in the RDF domain model, which must be unique, to appear in multiple subtrees of the generated XML representation. In the Patient Record model, the ptrec:CardiacValveData class is inherited by each of the four specific valve classes – aortic, mitral, tricuspid and pulmonary.

Excerpt of the SemanticDB meta-model, in N3:

```
:AorticValve a dnode:DataNodeClass;
 rdfs:comment """The aortic valve is situated at exit
   of the left ventricle of the heart where the aorta
   begins, and lets blood from the left ventricle be
   pumped up into the aorta but prevents blood once it
   is in the aorta from returning to the heart.""";
 rdfs:label "Aortic Valve";
 dnode:Index "true";
 dnode:inheritsConstraints :CardiacValveData;
 dnode:contains
   :LeftVentricularOutflowTractMeanGradient,
   :LeftVentricularOutflowTractPeakGradient;
 rdfs:subClassOf dnode:OptionalSingleton .
:CardiacValveData a dnode:DataNodeClass;
 rdfs:label "Cardiac Valve Data";
 dnode:contains
   :CardiacValveMeanGradient,
   :CardiacValveOrificeAreaData,
   :CardiacValvePeakGradient,
   :CardiacValveRegurgitantJetNumber,
   :CardiacValveRegurgitation,
   :CardiacValveRegurgitationFraction,
   :CardiacValveStenosis;
 dnode:inheritsConstraints
   :CardiacValveAnatomicalPathologyReference,
   :CardiacValveEtiologyReference,
   :CardiacValveTypeData;
 rdfs:subClassOf dnode:ZeroOrMore.
```

SemanticDB also makes use of some of the more semantically rich relations available in OWL. To facilitate easier queries, any dnode:contains relationship in the meta-model can be aliased with a more descriptive predicate that will be used both in the generated OWL ontology and the RDF graph of a particular patient record. For example, in the meta-model a patient contains a birth date as shown in the following RDF triple.

```
ptrec:Patient dnode:contains ptrec:BirthDate .
```

This assertion is used to generate a parent-child relationship in an XML patient record as shown in the following example.

```
<Patient>
<BirthDate>1958-03-05T00:00:00</BirthDate>
</Patient>
```

But the meta-model also has an alias for this particular dnode:contains relation using the predicate "hasBirthDate" so that an RDF representation of this same relation in the example above has the following subject-predicate-object form.

```
ptrec:Patient
   ptrec:hasBirthDate
     "1958-03-05T00:00:00"^^xsd:dateTime .
```

Due to its focus on data collection and management, the ontology generated from SemanticDB's meta-model is a skeletal depiction of the medical domain relevant to a patient record. However, careful use of standard terminologies facilitates linking the generated ontology to other, richer ontologies to support more useful inference. For example, each of these four valve classes, when represented in the OWL ontology, can then be linked to other ontologies that include additional information about these valves such as anatomical relationships, physiological characteristics, and prosthetic device properties.

4.1. Capturing the Medical Domain

Typically, registries of medical data are constructed around an indexing event, such as a particular kind of surgical procedure, and the patient's state and clinical experience before, during and after the procedure, and are documented by specific variables. For example, a patient undergoing cardiac surgery would have preoperative risk factors documented, such as diabetes status and history of myocardial infarction, and post-operative complications, such as reoperation for bleeding. This registry structure works well as long as the indexing event is the event of interest for a given use. However, if one's interest is another event – say a cardiac catheterization that was performed some time prior to the surgery – it is no longer clear that the description of the state of the patient before the cardiac surgery also applies to the state of the patient prior to the catheterization. The myocardial infarction that occurred before the surgery may have occurred after the catheterization, and thus would be a post-procedure state for the catheterization. With this kind of structure, each event of interest requires a separate registry often having considerable content overlap with other registries, resulting in a system that is both difficult to use and wasteful of resources needed to collect and retrieve data.

To avoid these problems, our goal in constructing a model of the cardiovascular medicine domain in SemanticDB was to produce a record of the patient's

medical history that would be useful from many frames of reference including different events of interest and definitional perspectives. To accomplish this, we divided the patient record into three fundamental parts – patient demographics, patient history, and medical events – and then defined core data elements (described below) within each of these parts that could be used to derive values consistent with particular definitions, as illustrated in Fig. (**3**).

Patient demographics are characteristics of the patient that do not change, such as birth date, sex at birth, race, etc., or things that do change, but whose change history is not medically relevant, such as billing address, phone number, etc. Patient medical history consists of disease, treatment and social histories documented during clinical encounters. Rather than capture all histories anew for each encounter, the SemanticDB patient record model documents changes in medical history only after the initial baseline history is collected during the first patient encounter. New or updated medical history data documented in subsequent encounters are tagged with the encounter date-time so that a correct history can be determined for any index event regardless of when it occurred. Medical events include encounters, diagnostic exams, treatment procedures, and morbidities (adverse events). For each event, the model identifies the event type/description, start and end date-times (modeled to allow fuzzy time), event place, and data source. Any number of additional variables can be added to this core set of event properties to fully document a given event. By documenting all relevant medical events independently, one may choose any of one them as an index event and determine the patient's pre-event and post-event characteristics by interrogating data associated with events that occurred before and after the index event respectively.

Definitional flexibility derives from a common denominator focus on the collection of core data elements rather than secondary or derived data. The definition of terms such as "current smoker" or "surgical site infection" can vary among registries and through time within the same registry. A core data elements approach takes these terms and breaks them down into atomic variables that can be logically combined to derive the particular values required by a particular registry. In the case of current smoker, definitions vary by how recently a patient must have quit smoking to still be considered a current smoker, and what the patient smoked (*e.g.*, cigarettes only, any tobacco product, etc.). In the SemanticDB patient record model, we collect the patient's smoking status (never, quit or current), the date the patient quit smoking if they quit, and the materials smoked. These core data elements allow us to derive an appropriate value for the field "current smoker?" regardless of whether the intended definition includes patients who smoked within one year or one month of the index event. Of course it is often impractical to collect core data elements for all fields. We adopted a pragmatic approach that sought a middle ground between infinitely reusable atomic detail and special purpose definitions with limited reusability.

5. USES IN CLINICAL RESEARCH

Clinical research uses prospectively and retrospectively collected clinical data to investigate the short and long term effects of medical care on survival, quality of life, cost of care delivery and other dimensions. In contrast to clinical trials in which patients are randomly and blindly assigned to different treatment groups, comparative effectiveness studies use data on actual nonrandom clinical care to compare the effectiveness of different treatments. Clinical research involves several steps including identifying the cohort of patients who match specific inclusion and exclusion criteria, compiling and collecting the various data needed on these patients to perform the study, creating a dataset suitable for statistical analysis, and conducting and interpreting the statistical analyses.

To identify patient cohorts for individual studies in SemanticDB, we use the SRA query tool described above. Queries specifying patient inclusion and exclusion characteristics are constructed in SRA by entering a natural language string, selecting from a library of query fragments derived from the natural language input, and using a graphical tool for diagramming temporal relationships among events. When a valid query has been composed, the SRA generates the appropriate SPARQL queries and submits them against the SemanticDB RDF store. Results in the form of patient and index event identifiers are streamed back to the SRA interface where they can be stored in SRA for further use in other queries and exported as delimited files for use elsewhere.

Based on the patient cohort identified by SRA, data can be pulled from SemanticDB for building the analysis dataset by either submitting the cohort list to the Export Tracker utility in Fig. (**2**) to extract standard data modules, running additional queries in SRA against the saved cohort, or composing SPARQL queries directly in SemanticDB's SPARQL endpoint.

The Export Tracker takes a cohort list of index events, generated by the SRA or any other source, and matches it against patient records in SemanticDB. If no match is found for a particular index event, or if the match is ambiguous (e.g., multiple index events on the same day), Export Tracker flags these cases so that the issues can be resolved by the user before resubmitting the data export request. When a valid cohort list has been accepted by Export Tracker, it runs a predefined set of data export modules that contain some 300 variables needed for most studies including pre-event patient characteristics, index event details, post-event complications, and the results of long-term patient follow-up. These data exports are generated by retrieving the patient's XML data file to guarantee that the data are current, augmenting it with data from other sources using an external data service in SemanticDB, and automatically converting it to RDF for query and rule-based transformations, as previously described.

If data needed for a study are present in SemanticDB, but not provided in a standard data export module, they can be retrieved for the study cohort using queries composed in SRA, or, if the queries involve more complex logic, through hand-built SPARQL queries run through the SPARQL endpoint. SemanticDB can also support the collection of study-specific data not already in the repository, through routine data collection and integration. New variables can be added to the meta-model, and Web-based data collection forms can be generated for use in manual abstraction of these

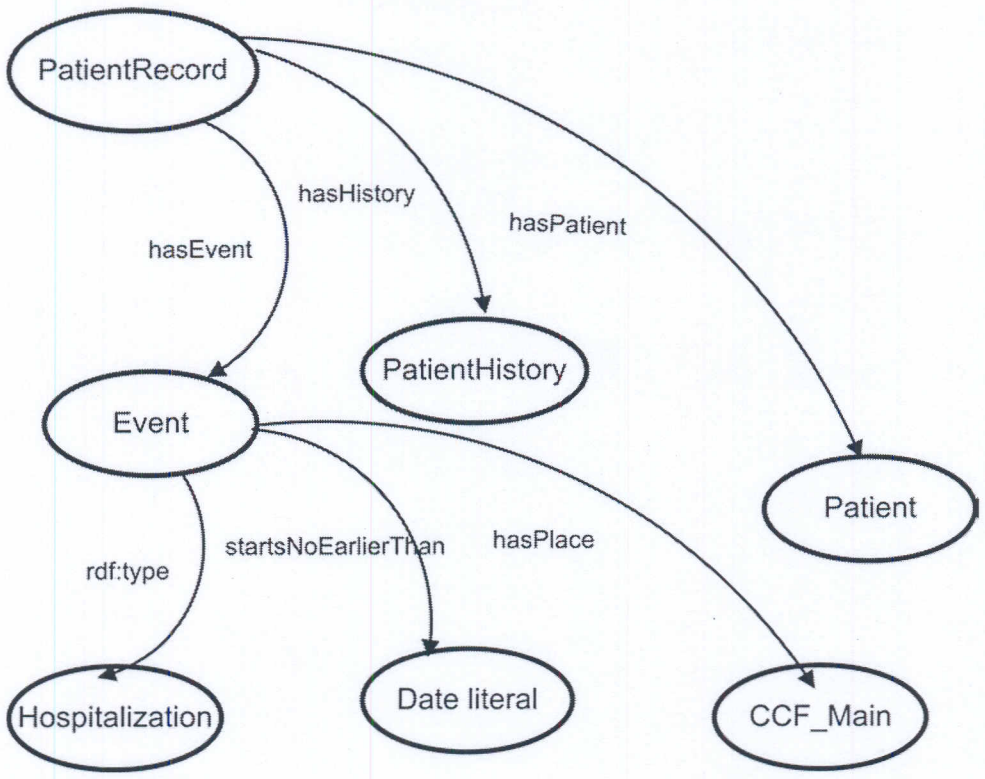

Fig. (3). Portions of the patient record ontology used in SemanticDB. The PatientRecord includes information about medical events ("Event" branch), the patient's medical history ("Patient History" branch), and patient demographics ("Patient" branch).

fields. Once these data have been added to SemanticDB, they can be retrieved by any of the methods discussed above.

From 2009 through June of 2011, over 200 clinical investigations utilized SemanticDB to identify study cohorts and retrieve appropriate data for analysis. These studies ranged from relatively simple feasibility assessments to extremely complex investigations of time-related events and competing risks of the patient experiencing a certain outcome after treatment. Prior to the deployment of SemanticDB, the cohort identification and data export queries for all of these studies would have been performed by a skilled database administrator (DBA) interpreting instructions from domain experts. Using SemanticDB, a non-technical domain expert performed all of the queries except those requiring the manual construction of SPARQL, which made up a less than 1% of all queries of SemanticDB. Not only did this reduce the costs of performing research queries by circumventing the need for a DBA, but it is also likely to have improved accuracy through elimination of the interpretation of the query instructions by a DBA who is often unfamiliar with the domain or study.

6. USES IN QUALITY REPORTING

Healthcare outcomes reporting provides a means of quantitatively measuring the quality of care that patients have received at an institution, for use by both practitioners and administrators within that institution, and for comparison across institutions. Within Cleveland Clinic's Heart and Vascular Institute (HVI), SemanticDB has been used to generate outcomes reports both for internal and external consumption. Internal reports are generated monthly, and external reports are generated quarterly for both the Society of Thoracic Surgeons (STS) Adult Cardiac Surgery National Database and the American College of Cardiology (ACC) CathPCI Database.

An increasing number of agencies and medical societies are requiring such reports, and this places a heavy burden on healthcare institutions to produce the required data in the formats, and according to the precise definitions, that these agencies require. Unfortunately, although many of the variables required by these agencies are similar and partially overlap -- such as history of smoking -- precise definitions vary significantly, and this makes it difficult to use the same source data for all reports. Typically, an institution is forced to manually abstract the data for each report separately to conform to the requirements for that report. This duplication of effort not only wastes time and money, but it also creates potential legal and administrative issues if conflicting data are entered inadvertently. However, by basing our ontology on *core data elements* (as described above), SemanticDB is able to use automated inference rules to produce values for multiple variables required both within a report and across multiple reports.

For example, Fig. (**4**) shows four core data elements – Anti-anginal medication, Date/time, Medication prescribed and Medication taken – that have been extracted from three pieces of source information in the patient's medication record: Medication type: Anti-anginal medication, Date/time, Medication taken or prescribed. In some cases, the extraction of core data elements from source information can be fully automated. This is possible when the source information exists in structured form – not as part of an

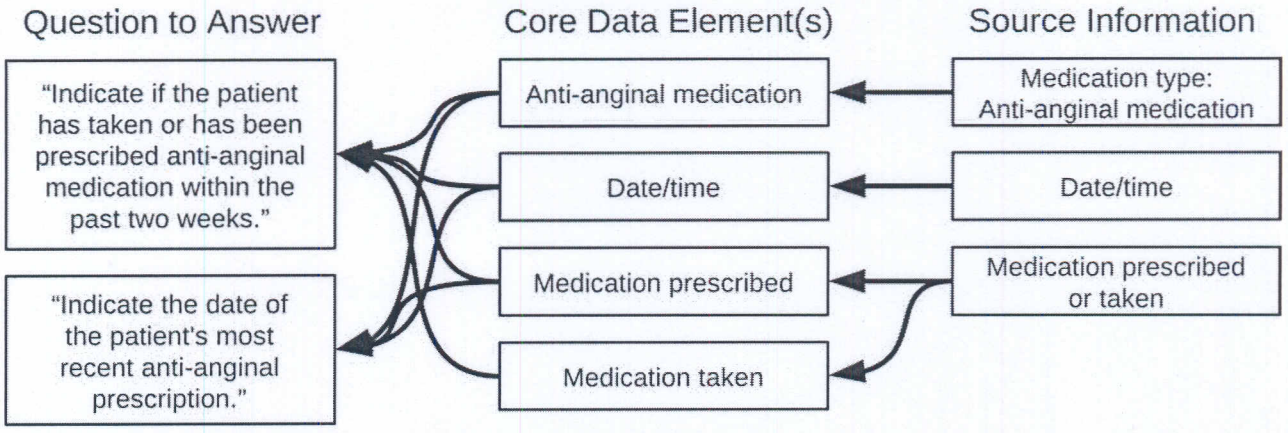

Fig. (4). Derivation of quality metrics from core data elements. Source information is used to derive atomic "core data elements" that are then used in combination by automated inference to produce answers to the specific questions required in each report.

arbitrary narrative – with fine enough granularity. In other cases, the extraction of core data elements may require manual abstraction by trained medical personnel examining the patient record. Ideally, the original source information should be collected in structured form with fine enough granularity to enable fully automated downstream use of the information. However, because the information collection procedures for a patient are widely distributed throughout that patient's many points of care, and they often involve multiple medical specialties using disparate information systems, the goal of fully automating the extraction of core data elements has not yet been achieved.

Once the values for core data elements have been extracted, they are combined using fully automated inference rules to compute the answers to specific questions required of a report. For example, Fig. (**4**) illustrates how these core data elements are used to answer the question: "Indicate if the patient has taken or has been prescribed anti-anginal medication within the past two weeks.", as required in the CathPCI v4.3 report. Using additional inference rules, these same core data elements are also used to compute the answer to the question: "Indicate the date of the patient's most recent anti-anginal prescription."

The core data elements ontology itself, and the inference rules that generate the answers to reporting questions, are derived through manual analysis of the various questions that must be answered, in order to identify the critical concepts that can be turned into core data elements. As the system is used to produce more reports, more common data elements are reused and the payoff increases.

For example, the CathPCI report v4.3 seq. #6130 "Mid/Distal LAD, Diag Branches Stenosis" requires that the reporting institution "Indicate the best estimate of most severe percent stenosis in mid/distal left anterior descending (LAD), including all diagonal coronary artery branches as determined by angiography." Within this question, the phrase "all diagonal coronary artery branches" expands into several critical concepts in cardiac anatomy:

- Diagonal 1
- Diagonal 2
- Diagonal 3
- Lateral First Diagonal
- Lateral Second Diagonal
- Lateral Third Diagonal
- Left Anterior Descending Major Septal Perforator

These concepts are captured as core data elements in the ontology, which also captures important relationships between these concepts. Once these core data elements have been defined in the ontology, and the associated mappings from the source information have been defined, they are used to write queries and inference rules that will automatically map patient data from the source information to the answers required for specific reporting questions. For example, the following two automated rules, written in the CycL language [17], are used in constructing the answer to the CathPCI report v4.3 seq. #6130 "Mid/Distal LAD, Diag Branches Stenosis" question. CycL terms beginning with "?" are treated as query variables.

```
(ist CCF-CAE-QueryMt
   (and
    (elementOf ?ARTERY-TYPE
     (TheSet
      MiddleLeftAnteriorDescendingArtery-Coronary
      LeftAnteriorDescendingDistalArtery-Coronary
      CoronaryArtery-Diagonal1
      LateralFirstDiagonalCoronaryArtery
      CoronaryArtery-Diagonal2
      LateralSecondDiagonalCoronaryArtery
      CoronaryArtery-Diagonal3
      LateralThirdDiagonalCoronaryArtery
      LeftAnteriorDescendingMajorSeptalPerforator))
    (cathOrPCIHasStenosisForCoronaryRegion
     ?INDEX ?ARTERY-TYPE ?DEGREE)))
  (implies
   (and
    (rdf-type ?INDEX InterventionalCatheterization)
    (hasFinding ?INDEX ?STENOSIS)
    (rdf-type ?STENOSIS CoronaryArteryStenosis-Finding)
    (cCFCoronaryArtery ?STENOSIS ?REGION-TYPE)
    (cCFVesselStenosisDegree ?STENOSIS ?DEGREE))
   (cathOrPCIHasStenosisForCoronaryRegion
    ?INDEX ?REGION-TYPE ?DEGREE))
```

Because report generation involves time intervals that cannot be known in advance, Cyc's backward chaining capabilities are used to select only the data that is needed for a particular reporting period.

This automation of the reporting process, through inference about core data elements, has not only reduced the burden of duplicate effort, it has imposed transparency and consistency that have reduced errors in the reported data. Both the Society of Thoracic Surgeons and the American College of Cardiology have certified Cleveland Clinic to use SemanticDB to generate and submit data to their national health care quality registries.

7. CONCLUSIONS

We have gained significant experience in the 8+ years since we first started to apply semantic web technology to electronic medical records in support of clinical research and quality-of-care measurement. The benefits of this technology have now been amply proven over multiple years of production use at the Cleveland Clinic's Heart and Vascular Institute. However, being a pioneer was not always easy. Initially the biggest challenges were due to immature or unavailable semantic web tools. This affected both initial design of SemanticDB and our work efforts, as we had to build much more of the infrastructure from scratch than would now be necessary. Semantic web tooling has improved greatly since the SemanticDB project began in 2003, so others embarking on this route will have an easier voyage. Some lessons learned:

- Because RDF-related data capture tools were not available, we placed more emphasis on XML than we would now. Tools such as Callimachus [23] now enable data to be captured directly to RDF.
- Performance has been adequate but not stellar, largely because we have focused first on functionality. However, it is clear that there are a number of ways that performance could be improved significantly beyond what we have been able to achieve if we decided to address them. We have not seen any essential barriers to a more efficient system.
- Ontology alignment with current medical practice has been a challenge. For example, patient medical history is usually collected by clinicians from the perspective of the current encounter and often lack sufficient precision to convert to core data elements that can be repurposed downstream. The most effective solution to this problem would involve modifications to the clinical documentation practices that would better enable the data's downstream reuse. Furthermore, there is a great deal of potential benefit in applying semantic techniques to the data collection process to make it more efficient, accurate and user friendly. For example, semantic inference could be used to detect errors or suggest data entry elements based on knowledge of the patient's history and treatment context based on machine learning.
- Maintaining semantic alignment in the face of versioning: different versions of instance data, rules and ontologies must be kept aligned as changes occur. Although this coordination can be done manually, it would be more accurate and reliable using automated support.
- Our experience indicates that Cyc's use of optimal or approximate computability, rather than provably complete computability of inference rules, is an efficient and fully sufficient approach to query generation and answering in the context of clinical research and quality reporting.
- Some of the most challenging difficulties encountered in trying to use inference on clinical data in SemanticDB derived from the lack of a complete longitudinal record of the patient's medical experience independent of a given organization. In the United States, patient medical records reside with health care organizations rather than with patients. Consequently, when patients obtain their health care from multiple institutions, each of these institutions captures previous health care as medical history, which often lacks details such as timing of events needed by inference rules. The fuzziness of medical history data makes it difficult to know the actual sequence of and timing between events that appear as variables in inference rules.

ACKNOWLEDGEMENTS

Support for development of SemanticDB was provided by Cleveland Clinic with backing by Delos Cosgrove (CEO), Martin Harris (CIO), and Joe Turk (Information Technology Director), and a portion of a grant from the State of Ohio's Third Frontier human genetics and biomedical engineering initiative. Developers who worked on the project in addition to the authors include Sivaram Arabandi, Steven Battle, Brian Beck, Alice Chan, En Cheng, John Clark, Samantha Davis, Adam Dutko, Brendan Elliott, Peter Kisule, Jeffrey Laing, James Leigh, William Stellhorn, and Larry Streepy. Employees of Cycorp who worked on the project include Blake Shepard, Keith Goolsby, Ronald Loui, David Schiender, David Baxter and Steve Collins. Jerry Scott brought Cleveland Clinic and Cycorp together and then managed and coordinated the work of the different teams. The project was also aided by input from members of a scientific advisory board including James Hendler, Doug Lenat, Brian Levy, and Barry Smith.

CONFLICT OF INTEREST

Declared none.

REFERENCES

[1] Blackstone EH, Lenat DB, Ishwaran H. In: Olsen L, Grossmann C, McGinnis JM, Eds. Learning what works: Infrastructure required for comparative effectiveness research workshop summary. Washington DC, The National Academy Press 2011; 123-144.

[2] Klyne G, Carroll JJ, McBride B; World Wide Web Consortium. Resource Description Framework (RDF): Concepts and Abstract Syntax. http://www.w3.org/TR/rdf-concepts/ (Accessed July 22, 2011).

[3] Berners-Lee T, Connolly D; World Wide Web Consortium. Notation3 (N3): A readable RDF syntax. http://www.w3.org/-TeamSubmission/n3/ (Accessed July 22, 2011).

[4] Cycorp, Inc.. The Cyc Knowledge Server. http://www.cyc.com-/cyc/technology/whatiscyc (Accessed July 22, 2011).

[5] Wikipedia.org. Ontology (information science). http://en.wiki-pedia.org/wiki/Ontology_%28information_science%29 (Accessed July 22, 2011).

[6] Bechhofer S, van Harmelen F, Hendler J, Horrocks I, McGuinness D, *et al.*; World Wide Web Consortium. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/ (Accessed July 22, 2011).

[7] Wood D, Eds, Ogbuji C, A Role for Semantic Web Technologies in Patient Record Data Collection. Linking Enterprise Data. Springer 2010; pp 241-261.

[8] Bray T, Paoli J, Sperberg-McQueen M, Maler E, Yergeau F; World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Fifth Edition). http://www.w3.org/TR/xml/ (Accessed July 22, 2011).

[9] Boyer J; World Wide Web Consortium. XForms 1.1. http://-www.w3.org/TR/xforms11/ (Accessed July 22, 2011).

[10] Clark J; World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/xslt (Accessed July 22, 2011).

[11] rdflib.net. RDFLib; A Python library for working with RDF. http://code.google.com/p/rdflib/ (Accessed July 22, 2011).

[12] Prud'hommeaux E, Seaborne, A; SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/ (Accessed October 29, 2011).

[13] Elliott B, Cheng E, Thomas-Ogbuji C, Ozsoyoglu ZM. A complete translation of SPARQL into efficient SQL. IDEAS Proceedings 2009; 31-42.

[14] Oracle, Inc. Oracle Spatial and Oracle Locator. http://tinyurl.com/-3lk53ne (Accessed July 22, 2011).

[15] Wikipedia.org. Rete Algorithm. http://en.wikipedia.org/wiki/-Rete_algorithm (Accessed July 22, 2011).

[16] Cycorp, Inc.. The Cyc Knowledge Base(TM). http://cyc.com/cyc/t-echnology/technology/whatiscyc_dir/whatsincyc (Accessed July 22, 2011).

[17] Cycorp, Inc.. The Syntax of CycL. http://www.cyc.com/cycdoc/-ref/cycl-syntax.html (Accessed July 22, 2011).

[18] SourceForge.net. 4Suite and related projects at SourceForge. http://foursuite.sourceforge.net/ (Accessed July 22, 2011).

[19] Google. FuXi 1.0: A Python-based, bi-directional logical reasoning system. http://code.google.com/p/fuxi/ (Accessed July 22, 2011).

[20] SourceForge.net. MySQL-Python. http://mysql-python.sourceforg-e.net/ (Accessed July 22, 2011).

[21] Massachusetts Institute of Technology. Exhibit: Publishing Framework for Data-Rich Interactive Web Pages. http://www.-simile-widgets.org/exhibit/ (Accessed July 22, 2011).

[22] Mozilla Foundation. Mozilla Firefox Web Browser. http://www.-mozilla.com/en-US/firefox/new/ (Accessed July 22, 2011).

[23] CallimachusProject.org. Callimachus. http://callimachusproject.-org/ (Accessed July 22, 2011).

[24] iSoft, Inc. Viaduct(TM). http://www.bridgeforwardsoftware.com/-viaduct.php (Accessed July 22, 2011).

[25] Ogbuji C, Blackstone E, Pierce C; World Wide Web Consortium. Case study: A Semantic Web content repository for clinical research. http://www.w3.org/2001/sw/sweo/public/UseCases/ClevelandClinic/ (Accessed July 22, 2011).

Received: July 27, 2011 Revised: October 26, 2011 Accepted: October 30, 2011