# Improving Query Mechanisms for Biomedical Ontologies

Ioana Branescu-Raspop, Victor Lorin Purcarea and Radu Dobrescu

***Abstract*—** The paper presents several solutions for improving access to data in biomedical ontologies, considering a particular dataspace which contain highly heterogeneous data or has an unknown or unreliable structure. The main goal is to provide means for explorative querying, which can be performed without any prior in-depth knowledge of the queried data. After presenting the features and the advantages in using Web services technologies, two new achievements are discussed: 1) an architecture for multiple sources data integration using interfaces harmonized by a wrapper component which is accessed by a mediator for implementing the required functionalities and 2) the prototype of a portable question-answering system named MQAS (Medical Question Answering System) which takes queries expressed in natural language and an ontology as input, and returns answers drawn from one or more knowledge bases. Conclusions on the results of preliminary tests performed in Clinical Institute in Bucharest and purposes for further work are also presented.

***Keywords*—** biomedical ontology, query interface, semantic web technologies, web services.

## I. Introduction

It is challenging for biomedical researchers to stay current with the literature in their field. At the other hand, to ensure best practice and treatment for patients, clinicians must access medical information, acquire new knowledge, and achieve information mastery in their field. Information retrieval systems are widely used; however, they return documents that have to be read by the user to extract relevant information. Inspired by the work of Cimino and Ayres [1] which developed a self-service query interface for re-use in clinical and translational research of the data captured in the course of routine patient available in the Biomedical Translational Research Information System (BTRIS) at the US National Institutes of Health (NIH), we have developed an ontology-driven Question Answering system to interface the Semantic Web to provide answers for a wide array of questions that arise in clinical work and biomedical research..

While semantic information can be used in several different ways to improve question answering, an important consequence of the availability of semantic markup on the web is that this can indeed be queried directly. In other words, we can exploit the availability of semantic statements to provide precise answers to complex queries, allowing the use of inference and object manipulation. Moreover, as semantic markup becomes ubiquitous, it will become important to be able to ask queries and obtain answers, using natural language (NL) expressions, rather than the keyword-based retrieval mechanisms used by the current search engines. More of that, our methodology for question answering was implemented in a prototype tool which returns answers (known facts) first, and only later the documents from which the facts are extracted.

The proposed prototype titled MQAS (Medical Question Answering System) is a natural language based front-end for the semantic web which takes queries expressed in natural language and an ontology as input, and returns answers drawn from one or more knowledge bases (KBs), which instantiate the input ontology with domain-specific information. Also, MQAS is coupled with a portable and contextualized learning mechanism to obtain domain-dependent knowledge by creating a lexicon. The learning component ensures that the performance of the system improves over time, in response to the particular community jargon of the end users.

## II. Semantic Web Technologies

The goal of the Semantic Web is to solve the current limitations of the Web by augmenting Web information with a formal representation of its meaning. A direct benefit of this machine processable semantics would be the enhancement and automation of several information management tasks, such as search or data integration. The current focus of Semantic Web research is more and more directed towards supporting intelligent data exchange. In this case the information that is being annotated is not unstructured text but rather semi-structured information available from databases or exchanged between Web services.

This work was supported in part by doctoral program POSDRU/107/1.5/S/76813.

Eng Ioana Branescu-Raspop is PhD. student, Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: ioana.branescu@gmail.com

Prof. Victor Lorin Purcarea, Ph.D., is with Department of Healthcare Marketing, Technology and Medical Devices, Medical Informatics and Biostatistics, "Carol Davila" University of Medicine and Pharmacy, Bucharest, Romania, e-mail: victor.purcarea@gmail.com.

Prof. Radu Dobrescu, Ph.D, is with Faculty of Automatic Control and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: rd_dobrescu@yahoo.com

Semantic Web service technology aims to automate performing such tasks based on the semantic description of Web services. Using these descriptions the right services can be selected and combined in a way that would solve the task at hand. There are two major approaches to Web service composition. First, given the specification of a start and final state, pre/post condition reasoning is performed to select and combine the right services. Second, using the parametric design paradigm, generic task workflows are formally specified and than populated with the right Web services depending on the task at hand.

In this paper is presented a methodology which allows to utilize the expressiveness of ontologies in an incremental manner within a distributed environment. This novel integration methodology is oriented towards the requirements and challenges of the application domain. It utilizes the expressiveness of ontologies to bridge semantic heterogeneity that originates from the distributed and autonomous manner in which data is collected throughout the domain. The basic idea is that an integrated access to co-existing datasets is already useful for researchers. A concept is presented which implements this methodology within a distributed environment based on Semantic Web technologies, i.e., the Resource Description Framework (RDF) data model and the WSMO (Web Service Modelling Ontology) which overlaps the OWL-S ontology. The resulting system is flexible and built upon loosely coupled components.

### A. Resource Description Framework

Resource Description Framework (RDF) is the standard model for data interchange on the Web; prescribed framework for representing resources in a common format [2]. RDF is a graph-structured data model in which information is modelled as a set of triples. Each triple defines an atomic statement of the form (Subject, Predicate, Object) which states that the Subject has a property Predicate with value Object. RDF defines two different types of nodes. Subjects and predicates are always resources whereas objects are either resources or literals. Resources are identified by globally unique Uniform Resource Identifiers (URIs), which are a proper superset of URLs. In order to simplify the representation of structures which consist of several triples (such as lists) RDF allows for anonymous resources whose identifiers are only unique within the local context. Each object is either a resource or a literal. Literals are atomic values with optional type information (e.g., integer or string). RDF uses XML Schema data types, but users are also able to define their own.

An RDF graph is a directed labeled graph in which subject and object are labeled nodes and predicates are directed, labeled edges ranging from the subject to the object. The following definitions determine the meaning of terms used in the characterization of an RDF graph [3]:

*Definition 1* (RDF Node)

The set of RDF Nodes $T$ is defined as $I \bigcup B \bigcup L$ where $I$ is an infinite set of URIs, $B$ is an infinite set of Blank Node Identifiers, $L$ is an infinite set of Literals and $I$, $B$ and $L$ are pairwise disjoint.

*Definition 2* (RDF Triple)

A RDF Triple $T$ is a tuple $(s,p,o) \in (I \bigcup B) \times I \times (I \bigcup B \bigcup L)$.

*Definition 3* (RDF Graph)

An RDF Graph $G$ is a set of RDF triples.

The RDF data model offers several properties that render it interesting for a deployment in the biomedical domain. First, it combines flexibility with expressiveness. As data is modeled as a network of objects, RDF is well-suited for the canonical representation of heterogeneous datasets and structures and therefore fosters interoperability. Second, RDF provides explicit formal semantics which allow to decompose an RDF dataset into comprehensible atomary statements, even if there is no thorough understanding of the data, e.g., due to missing schema information. Third, RDF enforces the explicit definition of entities, identifiers and relationships. For this reason, under the assumption of a suitable naming convention, resources can be uniquely identified on a global scale and RDF data can be easily combined with information from other datasets. This supports the development of incremental ontology-based approaches to information integration. For example, metadata, annotations or lineage information can be easily added to existing data. Furthermore, new attributes or concepts can be introduced and added to other datasets. At the same time RDF is characterized by its consistency, as data, metadata and semantics can be represented within one model.

### B. Generic Web Service Ontologies

The recent developments in Web Services for Ontologies are based on the Web Ontology Language (OWL) [4] and are defined as OWL-S [5]. OWL is a semantic markup language for publishing and sharing ontologies on the world wide web, and together with RDFS is considered a more expressive way for describing things in the world and how they are related using classes and properties.

A common characteristic of all emerging frameworks for semantic Web service descriptions (OWL-S) is that they combine two kinds of ontologies to obtain a service description. First, a generic Web service ontology, such as OWL-S, specifies generic Web service concepts (e.g., Input, Output) and prescribes the backbone of the semantic Web service description. Second, a domain ontology specifies knowledge in the domain of the Web service, such as types of service parameters and functionalities that fills in this generic framework.

The OWL-S ontology is conceptually composed of three sub-ontologies for specifying what a service does (Profile), how the service works (Process) and how the service is implemented (Grounding). A fourth ontology contains the Service concept which links together a ServiceProfile, a ServiceModel and a ServiceGrounding concept [6].

1. The *Profile Ontology* specifies the functionality offered by the service, the semantic type of the inputs and outputs, the details of the service provider and several service parameters, such as quality rating or geographic radius. This description is used for discovering the service.

2. The *Process ontology*. Many complex services consist of

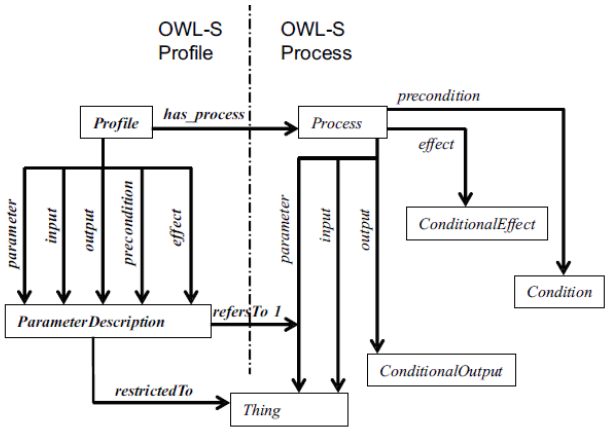


Fig. 1 Connection between QWL-S Profile and Process (after [7])

smaller services executed in a certain order. OWL-S allows describing such internal process models. These are useful for several purposes. First, one can check that the business process of the offering service is appropriate (e.g., product selection should always happen before payment). Second, one can monitor the execution stage of a service. Third, these process models can be used to automatically compose Web services. A Profile contains several links to the Process. Figure 1 shows these links, where terms in bold-face belong to the Profile ontology and the rest to the Process ontology.

Firstly, *a Profile* states the *Process* it describes through the unique property *has_process*. Secondly, the *Input, Outputs, Preconditions and Effects* (IOPE) of the Profile correspond to the IOPEs of the Process. The IOPE's play different roles for the *Profile* and for the *Process*. In the *Profile* ontology they are treated equally as parameters of the *Profile*. In the *Process* ontology only *Inputs* and *Outputs* are regarded as sub-properties of the *process:parameter* property. The *Preconditions* and *Effects* are just simple properties of the *Process*. The link between the IOPE's in the *Profile* and *Process* part of the OWL-S descriptions is created by the *refersTo* property which has as domain ParameterDescription and ranges over the *process:paramater*.

3. The *Grounding ontology* provides the vocabulary to link the conceptual description of the service, specified by the Profile and Process, to actual implementation details, such as message exchange formats and network protocols. The Grounding ontology specializes the *ServiceGrounding* as a *WSDLGrounding* which contains a set of *WsdlAtomicProcessGrounding* elements, each grounding one of the atomic processes specified in the *ProcessModel*.The grounding to a WSDL description is performed according to three rules:

R1. Each AtomicProcess corresponds to one WSDL operation.

R2. Each input of an AtomicProcess is mapped to a corresponding message-part in the input message of the WSDL operation. Similarly for outputs, each output of an AtomicProcess is mapped to a corresponding message-part in the output message of the WSDL operation.

R3. The type of each WSDL message part can be specified in terms of a OWL-S parameter (i.e., an XML Schema data type or a OWL concept).

Let remind that OWL is particularly designed for use by applications that need to process the content of information, and to facilitate greater machine interpretability of Web content than what is supported by XML and RDF, by providing additional vocabulary along with a formal semantics. In fact, it allows constraints on properties, equivalence and disjointness of classes; union, intersection and complement of classes, and finally to characterize properties [8].

## III. Optimizing Query Access to Ontologies

Query access to an ontology is probably the most traditional access to structured information. In general, for this type of services we enable the requester to specify the ontology to be queried, the query itself, the query language used by the query, and, possibly, a specific query engine to use. A query engine typically returns a set of variable bindings associating the variables of the query with the possible classes or relationships.

Combining ontologies and Semantic Web Services has benefits for both sides. On the one hand, by providing Web Services access to various ontology-manipulation operations and supporting composition of these Web Services, we enable applications to use ontologies just as they use data, through the same communication protocols and technologies that they use to access other services. On the other hand, by specifying protocols and infrastructure for ontology manipulation and access, we can streamline the variety of ontology-management tools developed in recent years by providing a unified infrastructure, specifying inputs, outputs and effects of each operation.

### A. Ontology Web Services capabilities

Using Web services capabilities for query access to ontologies not only provide a standard communication layer that allows any application to send its query and the ontology to be queried to a remote application dedicated to computing the result in a potentially structured way, but also a lot of facilities relating to generation of ontology views, translation of ontologies between formalisms, mapping and alignment of ontologies, ontology versioning, reasoning services, and so on.

Ontology Views. An ontology view provides a self-contained subset of an ontology based on the user's specification. This notion of a view is different from views in databases where views are defined as SQL queries. The binding for variables essentially provides a definition of a new virtual table or set of virtual tables in a database. However, such an approach can't be transposed to ontologies: the binding for variables is a set of the identifiers of the concepts that match the query, and not a set of concepts. If the variable binding resulting from a query can be seen as new tables for a database, they are by no mean a portion of an ontology or a restriction of its domain. A request for a web service providing

an ontology view contains the source ontology and the view definition. The result is also an ontology—in this case a subontology of the source ontology corresponding to the view (or a reference to it).

Ontology Translation. Ontologies can be represented in different ontology languages. Translation services convert an ontology from one representation language to another. Such services can be completely generic, performing the conversion based on general semantic mapping between primitives of the two languages. There are many initiatives currently underway to create such mappings for popular languages on the web.

Management of multiple ontologies.The Semantic Web approach is based on federating and combining different ontologies. Therefore, the existence of overlapping ontologies or different versions of the same ontology necessite to establish correspondences between multiple ontologies, recognize their differences, compose them, or handle versioning issues. Providing ontology mapping, merging, and versioning of ontologies through Web Services enables applications to handle semantic heterogeneity dynamically.

Improving semantics classification tools. The integration of formal OWL ontologies provides a highly relevant semantic classification of messages, and the reuse by other applications of ontological knowledge base is also guaranteed. The interoperability and the knowledge exchange between systems must be ensured by ontology integration. In order to ensure its reuse and interoperability with systems which requesting for its service of classification, it is benefic to mplement semantic classifier tool based on SOA [9].

Ensuring semantic interoperability. To bridge the gap between operational data and formal representations of concepts, the information model of a biomedical ontology is formally defined using OWL language to create a site-specific clinical domain. The links between the formal data source representations and the domain concepts are made through ontological mappings implemented via the ontology knowledge organization system. It is during the query translation process that domain concepts are annotated with domain classes and properties. Once this is done, the results are fully represented in terms of a formal ontology and their semantics are hence readily exploitable by computers.

### *B. Integration of multiple sources*

Integration of multiple, distributed, and heterogeneous sources are essential for developing query interfaces for ontologies based biomedical questions answering. Ensuring data quality in data integration is an issue or challenge because of their varying quality levels. The Meta data of data sources do not provide quality details and it is difficult to choose best query plan. It is also difficult to predict the resultant data quality before integration [10]. To mitigate above issues, this paper proposes a service oriented data integration architecture.

The access to biomedical data (such as data from relational databases) as well as the provision of an integrated view over the distributed sources of information are essential preliminaries for the implementation of the proposed concept. The utilized methods must be applicable without prior integration, mapping or annotation of the data sources. As a result, lightweight approaches and tools for further incremental semantic integration can be implemented. This includes, e.g., the annotation and transformation steps, which can be implemented on top of a global ontology-based view on the primary data sources as well as the results of previous integration steps. The global view also serves as an interface for users and applications. The implementation concept builds upon the RDF data model. To incorporate a fine-grained authentication model and preserve local access autonomies it implements a federated approach. From a querying perspective an example of the resulting system design is shown in fig. 2.

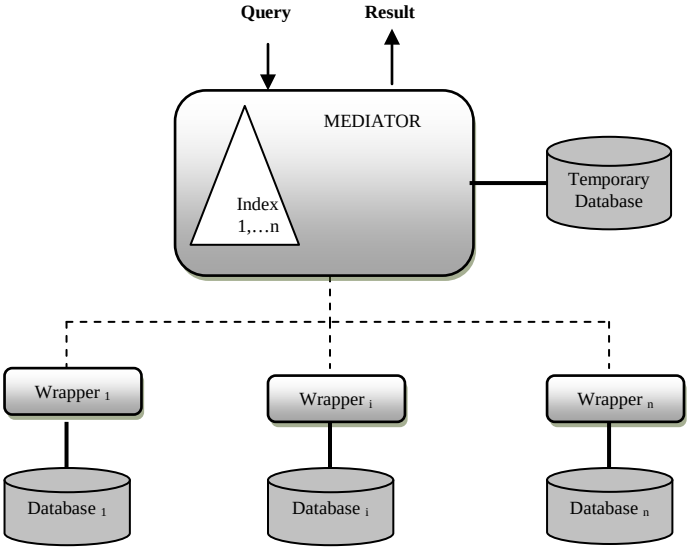


Fig. 2. Architecture for multiple sources data integration

Here, the primary data as well as annotations and materialized implicit knowledge is spread among several distributed databases. The interfaces of those databases are harmonized by a wrapper component which is accessed by a mediator for implementing the required functionalities. A global index structure allows the mediator to determine relevant databases when executing queries or data transformation workflows. In order to implement this concept, components are required which allow to integrate non-RDF databases into the resulting global graph, provide efficient distributed query processing and execute flexible data transformation workflows. These components need to be: 1) fully functional within a distributed environment and allow to re-model local authorization models; 2) lightweight in terms of deployment and maintenance efforts; 3) fully support the described incremental integration process; 4) able to manage large data volumes (several 100 M triples) efficiently.

The architecture implements a query execution model which consists of four steps: 1) Parse and optimize the query; 2) Execute subqueries at the remote systems (partly sequentialized with bind joins); 3) Load local results into a global database; 4) Execute the query on the global database.

This is basically equivalent to executing the query on a relevant subgraph of the global graph. Therefore an RDF store

can again be used as a global temporary database. Let note that the proposed architecture supports distributed indexing as well as query optimization and execution. Wrappers export standardized interfaces to the remote systems which are provided by instances of a RDF database system. All components are multithreaded and able to process several queries in parallel. Mediator and wrappers communicate via messages exchanged over plain sockets. The temporary global database should be also implemented as an instance of the RDF system, but because the developed optimizations generally cut down heavily on the number of intermediate results it is also possible to replace it by an in-memory database.

The proposed architecture can be successfully implemented for ontology based data integration with high quality of service, by fulfillment of the following requirements: 1) Local ontologies must be created from the local schemas of the data sources; 2) A global ontology must be created by using hybrid ontology approach; 3) data quality such as completeness and accuracy in data integration must be considered for each data source; 4) data source owners must be advetised about inadequate quality of the data in case of poor data quality.

## IV. MQAS Architecture

MQAS provides an API, which allows future integration in other platforms and independent use of its components. Fig. 3 shows the different components of the architectural solution, which is modular, flexible and scalable. The Linguistic Component (LC) and the Relation Similarity Service (RSS) are the two central components of MQAS. A key feature of MQAS is the use of a plug-in mechanism, which allows MQAS to be configured for different Knowledge Representation (KR) languages. For now the plug-in mechanisms is set for the RDF and OWL servers.

To reduce the number of calls/requests to the target knowledge base and to guarantee real-time question answering, even when multiple users access the server simultaneously, the MQAS server accesses and caches basic indexing data from the target Knowledge Bases (KBs) at initialization time. The cached information in the server can be efficiently accessed by the remote clients. Therefore, a mechanism is provided to update the cached indexing data on the MQAS server. This mechanism is called by these agents when they update the KB.

As regards portability, the only entry points which require human intervention for adapting MQAS to a new domain are the configuration files. Through the configuration files it is possible to specify the parameters needed to initialize the MQAS server. The most important parameters are: the ontology name and server, login details if necessary, the name of the plug-in, and slots that correspond to alternative names that an instance may have. Optionally, the main concepts of interest in an ontology can be specified.

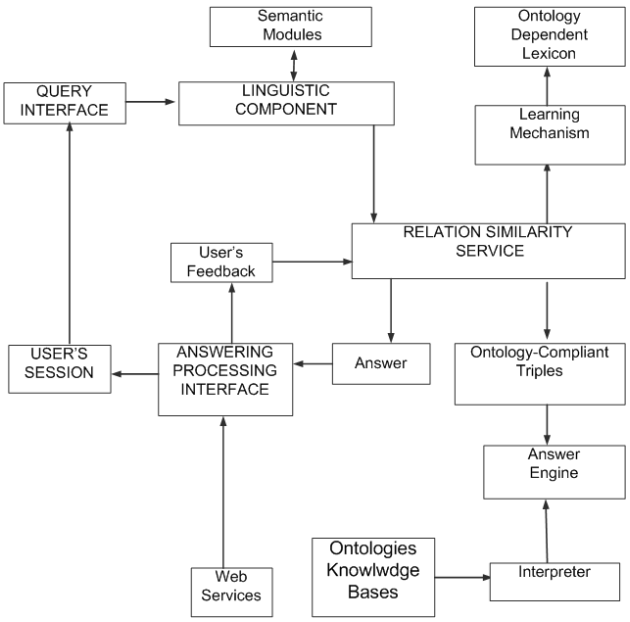


Fig. 3 The MQAS Global Architecture

The core to the overall architecture is the triple-based data representation approach. A major challenge in the development of the current version of MQAS is to efficiently deal with complex queries in which there could be more than one or two terms. These terms may take the form of modifiers that change the meaning of other syntactic constituents, and they can be mapped to instances, classes, values, or combinations of them, in compliance with the ontology to which they subscribe. Moreover, the wider expressiveness adds another layer of complexity mainly due to the ambiguous nature of human language. So, naturally the question arises of how far the engineering functionality of the triple-based model can be extended to map a triple obtained through a NL query into a triple that can be realized by a given ontology.

To accomplish this task, the triple in the current MQAS version has been slightly extended. Therefore an existing linguistic triple now consists of one, two or even three terms connected through a relationship. A query can be translated into one or more linguistic-triples, and then each linguistic triple can be translated into one or more Ontology-Compliant-Triples. Each triple also has additional lexical features in order to facilitate reasoning about the answer, such as the voice and tense of the relation. Another key feature for each triple is its *category*. These *categories* identify different basic structures of the NL query and their equivalent representation in the triple. Depending on the category, the triple tells us how to deal with its elements, what inference process is required and what kind of answer can be expected [11]. In what follows we provide an overview of the main components of the MQAS architecture.

### A. Linguistic Component (LC)

When a query is asked, the LC's task is to translate from NL to the triple format used to query the ontology (Query-Triples). This preprocessing step helps towards the accurate classification of the query and its components by using

standard research tools and query classification. Classification identifies the type of question and hence the kind of answer required. The annotations returned after the sequential execution of these resources include information about sentences, tokens, nouns and verbs. These features, returned for each annotation, are important to create the triples, for instance, in the case of verb annotations, together with the voice and sense it also includes information that indicates if it is the main verb of the query (the one that separates the nominal group and the predicate). This information is used by the Linguistic Component to establish the proper attachment of prepositional phrases previous to the creation of the Query-Triples.

Some examples can be seen in Table 1), including 2 basic queries requiring an affirmation/negation or a description as an answer (the first 2 rows) and queries constituted by a wh-question, where the relation is implicit or unknown (the last 2 rows).

Categories not only tell us the kind of solution that needs to be achieved, but also they give an indication of the most likely common problems that the Linguistic Component and Relation Similarity Services will need to deal with to understand this particular NL query and in consequence it guides the process of creating the equivalent intermediate representation or Query-Triple. For the intermediate representation, we use the triple-based data model rather than logic, mainly because at this stage we do not have to worry about getting the representation completely right. The role of the intermediate representation is simply to provide an easy and sufficient way to manipulate input for the RSS.

TABLE I. EXAMPLES OF NL QUERIES AND EQUIVALENT TRIPLES

| **wh-generic term** | **Linguistic Triple** | **Ontology Triple** |
|---|---|---|
| Who are the researchers in the semantic web research area? | <*person/organization*, researchers, semantic web research area> | <*researcher*, has-research-interest, semantic-web-area> |
| **description** | **Linguistic Triple** | **Ontology Triple** |
| Who are the academics? | <*who/what is, ?,* academics> | <*who/what is, ?,* academic-staff-member> |
| **wh-combination (and)** | **Linguistic Triple** | **Ontology Triple** |
| Does anyone has interest in ontologies and is a member of research group? | <*person / organization*, has interest, ontologies> <*person/organization*, member, research group > | <*person*, has-research-interest, ontologies> <*research-staff-member*, has-project-member , research group > |
| **wh-generic with wh-clause** | **Linguistic Triple** | **Ontology Triple** |
| What researchers, who work in faculty, have interest in ontologies? | <*researchers*, work, faculty> <*researchers*,has-interest,ontologies> | <*researcher*, has-project-member, faculty > <*researcher*, has-research-interest, ontologies > |

### B. Relation Similarity Service (RSS)

The RSS is the backbone of the question-answering system. The RSS component is invoked after the NL query has been transformed into a term-relation form and classified into the appropriate category. The RSS is the main component responsible for generating an ontology-compliant logical query. Essentially, the RSS tries to make sense of the input query by looking at the structure of the ontology and the information stored in the target KBs, as well as using string similarity matching, generic lexical resources, and a domain-dependent lexicon obtained through the use of a Learning Mechanism.

An important aspect of the RSS is that it is interactive. In other words, when ambiguity arises between two or more possible terms or relations the user will be required to interpret the query. Relation and concept names are identified and mapped within the ontology through the RSS and the Class Similarity Service (CSS) – not represented in fig.2 considering it is part of RSS. The similarity services should use the ontology semantics to deal with ambiguous situations.

Where the ambiguity cannot be resolved by domain knowledge the only reasonable course of action is to get the user to choose between the alternative readings. Moreover, since every item on the Onto-Triple is an entry point in the knowledge base or ontology, they are also clickable, giving the user the possibility to get more information about them. Note also that the category to which each Onto-Triple belongs can be modified by the RSS during its life cycle, in order to satisfy the appropriate mappings of the triple within the ontology.

The RSS procedure for basic queries has several steps. The query is classified by the Linguistic component as a basic generic-type, then the first step for the RSS is to identify that validity (consistence) of terms through the use of string distance metrics. Whenever a successful match is found, the problem becomes one of finding a relation which links these terms. By analyzing the taxonomy and relationships in the target KB, MQAS finds that the only correct relation between two terms. Having done this, the answer to the query is provided. Whenever multiple relations are possible candidates for interpreting the query, if the ontology does not provide ways to further discriminate between them, *string matching* is used to determine the most likely candidate, using the relation name, the learning mechanism, eventual aliases.

### C. Answer Engine

The Answer Engine is a component of the RSS. It is invoked when the Onto-Triple is completed. It contains the methods which take as an input the Onto-Triple, and infer the required answer to the user's queries. In order to provide a coherent answer, the category of each triple tells the answer engine not only about how the triple must be resolved (or what answer to expect) but also how triples can be linked with each other. For instance, MQAS provides three mechanisms (depending on the triple categories) for operationally integrating the triple's information to generate an answer. These mechanisms are:1) And/or linking; 2) Conditional link to a term; 3) Conditional link to a triple.

### D. Learning Mechanism

Since the universe of discourse we are working within is determined by and limited to the particular ontology used,

normally there will be a number of discrepancies between the natural language questions prompted by the user and the set of terms recognized in the ontology. In such a case, it becomes necessary to learn the new terms employed by the user and disambiguate them in order to produce an adequate mapping of the classes of the ontology. The learning mechanism (LM) in MQAS consists of two different methods, the *learning* and the *matching*. The latter is called whenever the RSS cannot relate a linguistic triple to the ontology or the knowledge base, while the former is always called after the user manually disambiguates an unrecognized term (and this substitution gives a positive result). When a new item is learned, it is recorded in a database together with the relation it refers to and a series of constraints that will determine its reuse within similar contexts [12].

## V. Tests and Experimental Results

MQAS is implemented in Java as a modular web application, using a client-server architecture and is supported by the platform INSER@SPACE, which was developed within the INSEED project. INSEED was a research project having the aim to create a modern framework for training and skills forming in higher education in science, design and services management and to promote innovation in service industry based on a model of open, continuous education and cloud type distributed computing infrastructure with virtualized resources and accesible as services, interconnected at European structures.

INSER@SPACE, like collaborative environment for dissemination of the INSEED project results, is an approach developed in the UPB that provides support for a modern education of the future, both at research and curricula development level in many services sectors. This involves using cutting edge technology infrastructure of cloud in different ways, among them organizing and providing content area dedicated to Service Science - through a shared environment knowledge (SSKE - Service Science Knowledge Environment); The discussed application was integrated in connection with the E Health top-level ontology (see http://sske.cloud.upb.ro/sskemw/index.php/E-Health) included in SSKE. The prototype is still under tests, but in the same time it will be developed to include more data.

*Test scenario*

In order to demonstrate the functionality of a self-service query interface, we developed a first simplified version of a software tool based on a set of domain-specific query modules that can be selected by the users and related to each other only through "AND" and "OR" relationships. We assessed the theoretical capabilities for the query tool to address the previous user query requirements with respect to the various domains, attributes and relationships. Analysis considered the capabilities of the initial version of the tool as well as of a subsequent extended version. System requirements were enumerated and then divided into those to be included in the first version of the new tool and in the later version.

User queries were collected by the courtesy of a research team of the Clinical Institute of Fundeni, Bucharest, as part of their routine user interactions. Each query was developed manually after discussion between a database analyst and user. Requests were classified based on the data domains requested (demographics, laboratory results, etc.), the types of data attributes specified (date range, value range, cardinality, etc.), and the relationships between the data.

Multiple modules may be selected. Each module includes a collection of optional query prompts (e.g., date and value and patient age ranges). The modules are defined using XML data structures that specify the prompts to be included and the SQL queries generated by the tool based on user input to the prompts. The domains and features of the initial and in work a later version of the system are shown in Table 2.

TABLE II. Features Planned for the Query Tool

| **Query Function** | **Version 1** | **Added features in later Version** |
|---|---|---|
| Domains | Laboratory Results<br>Clinical Documents<br>Medications | Diagnoses<br>ECG& Echo<br>Procedures |
| Query Features | Age Range, Date Ranges, Value Range | Controlled Terminology<br>Cardinality |
| Relationships | AND, OR | NOT, Before, After |

Rather than wait until all modules and all prompts were completed, we chose to deploy an initial version of the system with what we consider to be a "critical mass" of the most popular data domain modules and the most important query parameter prompts.

*Collection of User Queries*

A total of 22 user queries were collected over the past year. Some examples of correlation between the domains of interest and the attributes of those domains are shown in Table 3, according to the following key to Domains: *A*=Admission, *B*=Blood Bank, C=Clinical Documents, D=Demographics, *Dx*=Diagnosis, *L*=Laboratory Tests, *E*=Echo, M=Medications, *P*=Pathology and the key to Attributes: a=age, c=controlled terms, d=date range, t=text search, v=discrete values. Italic items represent features planned for the future version. For example, Query #3 involved the domains Laboratory (implemented) and Pathology (planned); Query #5 involved the domains Clinical Documents and Laboratory (implemented), respectively Diagnosis and Pathology (planned).

TABLE III. Characteristics of User Queries

| **Query #** | **Domains** | **Relationship** | **Attributes** |
|---|---|---|---|
| 1 | *B*, C, D, L, M | AND | c, d, t, v |
| 2 | *A*, D, *E* | AND | a, t, v |
| 3 | L, *P* | AND | d, t, v |
| 4 | D, L, M | AND | a, d, t, v |
| 5 | C, *Dx*, L, *P* | AND, OR | t, v |
| 6 | L, M, *P* | AND, OR | a, v |

*Comments on results*

Till now we do not collect enough queries to have a concluding analysis. From the mentioned 22 queries, only four can be handled totally by the initial version of the tool, but only 2 of them should not be handled by the later planned version of the tool. Based on our experience with the implementation of the tool thus far, we believe that the addition of new domains and attributes will be straightforward; their limitations are development time, not algorithmic complexity.

The user queries touched on all the domains implemented in the current version of the query tool, as well as most of the planned domains. As of this writing, a few domains (Alerts, Allergies, ECG, and Pulmonary Function) have yet to be requested. Four queries involved domains that we had not previously planned to provide: Research Study (three queries) and Discharge/Transfer events (one query).

*Further work objectives*

In order to maximmize the potential of the above described prototype, the intention is to integrate it in a extended cloud-type structure based on the concept of Computing Continuum (ComCon for short) [13]. ComCon represents physical computing means, embracing clouds, communicating objects, sensors and smart devices, possibly utilising open source approaches, which have informational representation in the comprehensive hierarchy of human needs and problems. ComCon semantically binds proved available experience in the needs' satisfaction, including related knowledge and necessary resources. In order to integrate our platform in such unique knowledge architecture that satisfies European Internet users' need in Web resource that centralizes generic and private human experience and related knowledge, the following issues will be approached:

- Development of Need-Oriented Programming modules, which programs are sets of executable instructions (for human or for the computer), on the one hand, and knowledge available both for human and software agents on the other hand.

- Development of software tools that provide automatic synthesis of software and/or scenarios of activities aimed to a satisfaction of needs or to problems' solving, especially answering to automatically generated questions

- Development of software tools that take charge of semantic search in the Web and allows users to operate their environment by means of a formulation of goals, of commands or/and by a description of a course of actions

- Development of innovative form of Web service, namely, query-service interface, able to recognise a current user's need or a problem and, given all available data, provides users with adequate service.

Another challenge is to enhance the role of biomedical ontologies as particular embodiment of knowledge, methods, and even philosophy. When seen as meta-models, ontologies "can be pushed into artefacts and embedded in working environments, shaping them and guiding in this way user's experience and expectations" [14]. The goal is to explore the way ontologies realize technological mediation processes toward the artefacts that contain them. Therefore, instead of the traditional focus on quantitative technological risk, we will focus on technology's soft impacts, especially how technology influences values, norms, aspirations, needs, identities, responsibilities, meanings, and human relations.

## VI. Conclusion

Ontologies are now increasingly used in information systems to enrich the data analysis process, allowing the user to use ontology concepts for semantic queries. They can represent a domain knowledge that is not possible with a relational database information model. In the present study, we proposed a framework to link a database to an external ontology using a semantic web tool.

We demonstrated the performance impact of ontology use in the querying process. Using the ontology as query support enriched the data analysis process and extracted the knowledge needed to answer medical questions. The ontologies include structured vocabulary and formalized knowledge to express queries, and act as a mediator between the user and the data.

The framework we propose is designed to promote progress in the healthcare services, with the goal, not of developing an underlying formal ontology for biomedical purpose, but rather of achieving appropriate mappings of existing available biomedical ontologies.

The results of our study are likely to lead to the development of additional requirements for the tool. We hope to continue with all the current planned extensions, even those that have not yet been required of user queries, because we believe it is only a matter of time before a user makes a request that will need them. The objectives for this future work aim including optimization of query performance, expanding the synonyms in our controlled terminology lookup tool, rendering query results into formats that meet user requirements, and adding post-processing data analysis and visualization tool.

We acknowledge that the proposed approach is not easy to apply because of several distinctions and constraints it requires. We believe, however, that the approach promises significant benefits, both practical (clinical) and theoretical (scientific), in the long run.

## References

[1] J.J. Cimino and E. J. Ayres, "The clinical research data repository of the US National Institutes of Health", *Studies in Health Technology and Informatics*, 2010, 160(Pt 2), pp. 1299-1303.

[2] RDF vocabulary description language 1.0: RDF schema. Website. Available online at http://www.w3.org/TR/rdf-schema

[3] F. Prasser, A. Kemper, and K. A. Kuhn, "Efficient distributed query processing for autonomous RDF databases". In *Proceedings of the 15th International Conference on Extending Database Technology*, 2012, pp. 372-383

[4] OWL web ontology language overview. Website. http://www.w3.org/TR/owl-features/

[5] *The OWL-S Services Coalition*. 'OWL-S: Semantic Markup for Web Services. White Paper.", Available online at www.daml.org/services/owl-s/1.0/owl-s.pdf, 2003.

[6] J. Zhang, "A Review of Ontology and Web Service Composition", *Project Report*. CSIRO Publishing., Available online at http://eprints.usq.edu.au/5640/

[7] M. Sabou, D. Oberle, and D Richards, "Enhancing Application Servers with Semantics", Available online at http://www.researchgate.net/publication/2872473
[8] A. Pesaranghader, and S. Muthaiyah, „Semantic Similarity Using First and Second Order Co-occurrence Matrices and Information Content Vectors, *WSEAS Transactions on Computers*, Issue 3, Vol. 12, 2013, pp.95-104
[9] S. Lgarch, M. K. Idrissi, and S. Bennani, "A Reusable and Interoperable Semantic Classification Tool Which Integrates Owl Ontology", *WSEAS Transactions on Computers*, Issue 9, Vol. 11, September 2012, pp.294-308
[10] Hema M. S., and Chandramathi S., „Quality Aware Service Oriented Ontology Based Data Integration", *WSEAS Transactions on Computers*, Issue 12, Vol. 12, 2013, pp.463-473
[11] D. L. Castillo and N.S. Abraham, „Knowledge management: how to keep up with the literature", *Clinical Gastroenterology Hepatol*., 2008, 6(12), pp.1294-1300
[12] I. Udroiu, I. Tache, N.Angelescu, and I. Caciula, „Methods of measure and analyse of video quality of the image", *WSEAS Transactions on Signal Processing*, Vol. 5, Iss. 8, 2009, pp. 283-292
[13] Bucholz, D., and Dunlop, J., The future of enterprise computing: preparing for the Compute Continuum, IT@Intel White Paper, Available online at http://www.intel.com/content/dam/doc/, 2011
[14] L. Anticoli and E. Toppano, "Technological Mediation of Ontologies: the Need for Tools to Help Designers in Materializing Ethics", International Journal of Philosophy Study, Vol. 1, Iss.3, 2013, pp.23-31

.