Cybernetics and Systems: An International Journal

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ucbs20

IMPACT OF REFLEXIVE ONTOLOGIES IN SEMANTIC CLINICAL DECISION SUPPORT SYSTEMS

Arkaitz Artetxe [a b], Eider Sanchez [a b c], Carlos Toro [a], Cesar Sanín [d], Edward Szczerbicki [d], Manuel Graña [c] & Jorge Posada [a]

[a] Vicomtech-IK4 Research Centre, San Sebastian, Spain

[b] Biodonostia Health Research Institute, eHealth Group, Bioengineering Area, San Sebastian, Spain

[c] University of the Basque Country (UPV/EHU), San Sebastian, Spain

[d] Faculty of Engineering and Built Environment, University of Newcastle, Newcastle, New South Wales, Australia

To cite this article: Arkaitz Artetxe , Eider Sanchez , Carlos Toro , Cesar Sanín , Edward Szczerbicki , Manuel Graña & Jorge Posada (2013): IMPACT OF REFLEXIVE ONTOLOGIES IN SEMANTIC CLINICAL DECISION SUPPORT SYSTEMS, Cybernetics and Systems: An International Journal, 44:2-3, 187-203

To link to this article: http://dx.doi.org/10.1080/01969722.2013.762256

Cybernetics and Systems: An International Journal, 44:187–203

ISSN: 0196-9722 print/1087-6553 online
DOI: 10.1080/01969722.2013.762256

Impact of Reflexive Ontologies in Semantic Clinical Decision Support Systems

ARKAITZ ARTETXE[1,2], EIDER SANCHEZ[1,2,3], CARLOS TORO[1], CESAR SANÍN[4], EDWARD SZCZERBICKI[4], MANUEL GRAÑA[3], and JORGE POSADA[1]

[1]*Vicomtech-IK4 Research Centre, San Sebastian, Spain*
[2]*Biodonostia Health Research Institute, eHealth Group, Bioengineering Area, San Sebastian, Spain*
[3]*University of the Basque Country (UPV/EHU), San Sebastian, Spain*
[4]*Faculty of Engineering and Built Environment, University of Newcastle, Newcastle, New South Wales, Australia*

Ontology processing is arguably a time-consuming process with high associated computational costs. Query actions constitute a crucial part of the reasoning process and are a primary source of time consumption. Reflexive ontologies (ROs) is a novel approach intended to reduce time consumption problems while providing a fast reaction from ontology-based applications.

In this article we present the implementation of a knowledge-based clinical decision support system (CDSS) for the diagnosis of Alzheimer's disease, which was the benchmark used to evaluate the impact of RO in the overall performance of the system.

The implementation details and the definition of the implementation methodology are exposed in this article, along with the results of the evaluation. Some novel techniques that aim to optimize the performance of ROs are also presented with highlights of the test application introduced in our previous work.

KEYWORDS CDSS, clinical decision support system, knowledge engineering, reflexive ontologies

Address correspondence to Arkaitz Artetxe, Vicomtech-IK4 Research Centre, Mikeletegi Pasealekua 57, Donostia, San Sebastian 20009, Spain. E-mail: aartetxe@vicomtech.org

INTRODUCTION

Ontologies are defined in the computer science domain as the explicit specification of a conceptualization (Gruber 1995) and are used in an increasing number of computerized applications (McGuiness 2003). Ontological engineering has become a recurrent research topic in different areas of knowledge (Chandrasekaran et al. 1999).

One of the areas where the use of knowledge engineering techniques enhances traditional approaches is the medical domain. One example of the aforementioned is in a recently presented Clinical Decision Support System (CDSS) by Toro et al. (2012), where ontologies are used to model the domain and then extract explicit and implicit knowledge, allowing a practical semantic approach to the diagnosis stage of the medical practice. In the aforesaid and similar approaches, massive performing of queries and subsequent processing of their results is one of the most intensive tasks.

Performing those queries over the knowledge base implies a high computational cost, especially when compared to classical implementations; for example, using relational databases instead. Therefore, the development of techniques aimed toward acceleration of the ontology query process without losing reasoning ability meets a clear need.

In order to address the aforesaid gap, Toro et al. (2008) introduced the concept of reflexive ontologies (ROs) as a method intended to speed up query processes by storing performed queries in the ontology itself (self-containment). As pointed out in the aforesaid work, one of the benefits of having self-contained queries lies in the speeding up of the query process, in a way similar to what database caching does with relational data models (Tolia and Satyanarayanan 2007).

The acceleration of the querying process is based on the hypothesis that similar queries tend to be recursive over time. Similarly, some parts of the ontology tend to be asked more frequently than others. RO is able to utilize this information in order to perform queries in a more cost-effective way.

In this article we present the implementation of a knowledge-based CDSS for the diagnosis of Alzheimer's disease (AD), making use of an RO. In the implemented system we performed a benchmarking in order to observe the performance differences between a system using RO and a system using conventional ontologies. Simultaneously, the system was used to design and test new techniques that delve into the extraction of implicit knowledge from the RO.

The goal of this comparative evaluation was to determine the reliability of the system and the effectiveness of the proposed optimizations.

The article is organized as follows: in the following section we present related work regarding the improvement of query answering along with some background concepts related to ROs and autopoiesis. Next we describe the implemented system. Then we present the evaluation methodology

and the results obtained from the evaluation. In the final section, we discuss conclusions and future work.

RELATED WORK

Relevant work has been reported recently in the field of query answering performance improvement over knowledge bases. Kollia et al. (2011) introduced optimization techniques that improve query answering performance for SPARQL-OWL (Simple Protocol and RDF Query Language; Web Ontology Language) queries. One of the optimizations presented in Kollia et al. (2011) consisted of utilizing precomputed information (e.g., the class hierarchy) in order to find the answer to a query by means of a cache lookup. This technique, along with some other optimizations such as axiom reordering, is shown to help improve query answering performance.

Our approach is similar to the one presented by Kollia et al. (2011) in the sense that both benefit from previously computed information in order to perform a cache-like access to the query answer. However, our approach goes further in the sense that RO keeps track of all of the queries made over the ontology instead of using some precomputation made by the reasoner.

Amir and McIlraith (2005) introduced an approach known as *partition-based logical reasoning*, which argues to improve the efficiency of the reasoning process. Algorithms for reasoning with partitions of related logical axioms were presented in their work.

Grau et al. (2005) proposed the concept of partitioning a Web Ontology Language ontology in subdomains (modeled as separate ontologies) using e-connections to combine them. The aim of this approach was to reduce the knowledge base portion that the reasoner had to work with by keeping irrelevant components of the ontology unloaded.

The work by Amir and McIlraith (2005) and Grau et al. (2005) was based on the idea of reducing the search space within the knowledge base in order to improve reasoning efficiency. Our work tackles the reasoning time issue from a different perspective, which is based on query caching rather than ontology partitioning.

The defining property of RO that claims to accelerate the querying process is similar to query caching techniques traditionally used in the context of relational databases. In particular, within the domain of web applications, many different techniques have been presented in order to generate efficient caches from web content in constant transformation (Altinel et al. 2002; Amiri et al. 2003). Analogously, an RO has the ability to generate and maintain an efficient cache system even when dynamic knowledge bases are involved.

ROs have the ability to maintain a history of the queries made over the ontology itself, which acts as a cache due to the faculty of query retrieval, which was explained in depth in Toro et al. (2008). In addition, an RO has

the ability to provide a more refined cache system, other than a simple query history, due to the property of self-reasoning over the query set. In this manner, the size of the cache can be reduced (or extended depending on specific needs) according to implicit knowledge.

Cobos et al. (2008) proposed an architecture that uses the reflexivity concept in order to perform a fast semantic retrieval in the film heritage domain. The results of the experiment showed a clear efficiency gain, with an improvement of two orders of magnitude in the execution time. Although the concept of using an RO for a fast query recovery is the same for both cases, the architecture and implementation differ to some extent from our approach.

The experiment by Cobos et al. (2008) was carried out using only simple queries (containing a simple condition clause) and the ontology they used within the experiment included 63 individuals. In this article we test the RO concept in a more complex environment, because we use complex queries and our domain ontology contains more than 1×10^4 individuals. In addition, our system handles a nonstatic ontology—that is, an ontology that grows over time—whereas the system used by Cobos et al. (2008) worked with a static ontology. The implementation of autopoiesis (self-creation, as explained in the section on autopoiesis) makes possible the use of nonstatic ROs.

Reflexive Ontologies

The RO concept was introduced by Toro et al. (2008) to define the capability of an abstract structure of knowledge (an ontology and its instances, in this case) of maintaining, in a persistent manner, every query performed on it and storing those queries as individuals of a class that extends the original ontology. According to the formal definition proposed by Toro et al. (2008), "a Reflexive Ontology is a description of the concepts, and the relations of such concepts in a specific domain, enhanced by an explicit self contained set of queries over the instances" (p. 176).

The advantages of having self-contained queries are related to the acceleration of the querying process as well as to the implicit knowledge that potentially can be obtained. This article delves into the first benefit; that is, the ability to speed up the querying process.

Figure 1 shows the logical structure of an RO, which is, basically, a traditional ontology extended with a reflexive structure (mainly composed by the query instances in the left part of the image). As can be seen, every query (Q_p) is related to at least one class of the ontology (C_i) and one—or more—instances (I_k).

According to the authors, in order to be compliant with the RO concept, an ontology must fulfill the next five properties:

1. Query retrieval: The system must be able to store every query—and subquery—performed on it as well as to return it when required.

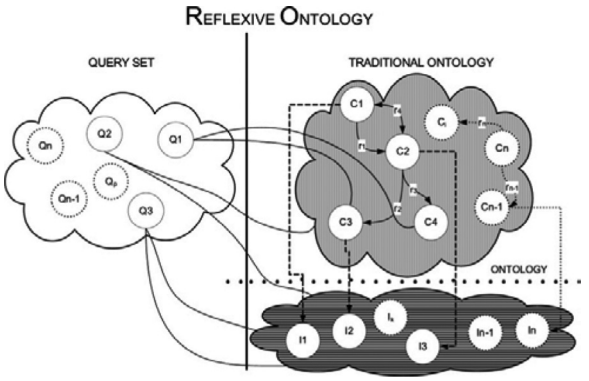

FIGURE 1 Schematic representation of the structure of an RO (Toro et al. 2008).

2. Integrity update: The system must have the faculty of actualizing the query set, every time a new individual is added, removed, or modified within the ontology (only in case the query contains such individual).
3. Autopoietic behavior: The system must have the capacity of self-creation because, for every new query launched, the system will evolve (refer to next section for further details).
4. Support of logical operators: The system must be able to handle (at least) the following logical operators: AND, OR, and NOT.
5. Self-reasoning over the query set: This property refers to the faculty of the system to (i) discover patterns of queries, (ii) recommend ontology refinement based on the queries performed over the system, and (iii) discover nonexplicit relationships between sets of queries.

Autopoiesis

In regards to etymology, autopoiesis means "self-creation or self-production" (Toro et al. 2008, p. 174). The concept was originally introduced by biologists Maturana and Varela in 1972 and describes a system that is capable of creating, modifying, and destroying components of the system itself depending on external perturbations. According to the definition given by Maturana and Varela (1980),

> [. . .] an autopoietic machine is a machine organized (defined as a unity) as a network of processes of production (transformation and production) of components that produces the components which, through their interactions and transformations, continuously regenerate and realize the network of processes (relations) that produced them [. . .] (pp. 78–79)

ROs have an autopoietic behavior because their structure is regenerated in response to external changes such as the launching of new queries or modification of the information stored in the ontology. Moreover, the

ontology is capable of storing the history of performed queries, which, for instance, allows knowing which parts and concepts of the ontology are consulted more regularly.

Accordingly, the autopoietic behavior ensures the integrity of the whole RO. When a new individual is created, modified, or removed from the ontology, the reflexive structure is updated. The updating process consists of modifying or generating new references to individuals for each query instance related to the change. By creating new connections (pointers to individuals) as a result of external perturbations, the system behaves as an autopoietic system or organism, according to the definition given by Maturana and Varela (1980).

SYSTEM IMPLEMENTATION

Our testing system is a CDSS for the diagnosis of AD. It has been implemented within the scope of a large-scale Spanish research project aiming at the early diagnosis of AD.

The CDSS implemented is a collaborative and multidisciplinary tool where physicians can (1) introduce patient data and the results from clinical tests, (2) review and edit these data at any time and location, and (3) get support from the system to assist them during decision making about the diagnosis of AD.

This system consists of three different modules, as presented by Sanchez et al. (2012): (1) the ontologies module, (2) the reasoning module, and (3) the query system. Briefly, each of these modules is described below.

Domain Ontology

Our ontology module, presented in Sanchez et al. (2011), consists of a domain ontology defined by experts for the specific domain of the AD diagnostic system. Our domain ontology provides a description of the different clinical tests carried out on patients for early detection of AD. This ontology was implemented in OWL-DL[1] and is mapped to SNOMED-CT[2] and SWAN[3] in order to provide, respectively, standardized terminology and bibliographic endorsement of the knowledge and criteria embedded. Figure 2 shows the main classes of the ontology and its relationships.

This knowledge base contains not only the classes depicted in Figure 2 and its instances but also those required for a reflexivity enhancement.

[1]OWL DL: Web Ontology Language (Description Logics [DL] stands for its expressiveness) http://www.w3.org/TR/owl-guide/

[2]SNOMED CT: Systemized Nomenclature of Medicine Clinical Terms, http://www.ihtsdo.org/snomed-ct/

[3]SWAN: Semantic Web Applications in Neuromedicine, http://www.w3.org/TR/hcls-swan/

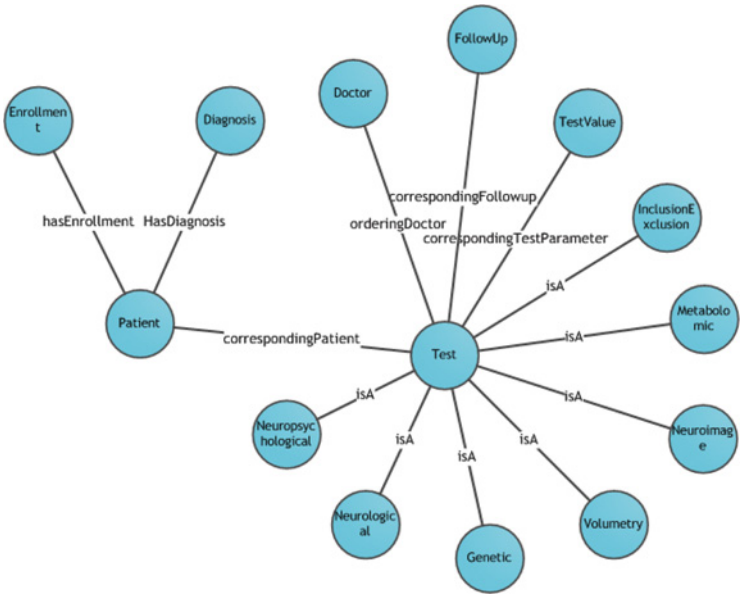

FIGURE 2 Ontology of our CDSS (color figure available online).

Reasoning Module

The reasoning module performs a semantic reasoning process based on a set of rules given by clinicians. As can be seen in, Figure 3 every rule contains its own ID, a weight, the rule itself, and the corresponding bibliographic source.

In the current version of the CDSS the set of rules contains 138 rules, although the number may vary in the future. A rule is composed of at least two clauses: the clause corresponding to the *if* part and the one corresponding to the *then* part; a third one, *else*, is optional. A rule is said to be simple if it contains a unique clause or complex if more than one clause are present. Connectors are logical operators (AND, OR, and NOT) used to build more refined clauses. Every clause is formed by four elements: (1) class, (2) property, (3) modifier, and (4) value. Figure 4 depicts the structure of a clause along with a simple example.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<RuleSet>
    <LoadRule>
        <RuleID>HUVR_1</RuleID>
        <Rule>If (( CLASS InformacionClinicaNeurologica with the PROPERTY InformacionClinicaNeu
        AND ( CLASS CriteriosInclusionExclusion with the PROPERTY CriteriosInclusionExclusion_C
        AND ( CLASS Reclutamiento with the PROPERTY Reclutamiento_InformacionSociodemografica_A
        then ( CLASS InformacionDiagnosticaPaciente with the PROPERTY InformacionDiagnosticaPac
        <weight>0.2</weight>
        <AccordingTo>
            <classes>
                <class>JournalArticle</class>
            </classes>
```

FIGURE 3 Partial view of a rule given by the experts (color figure available online).

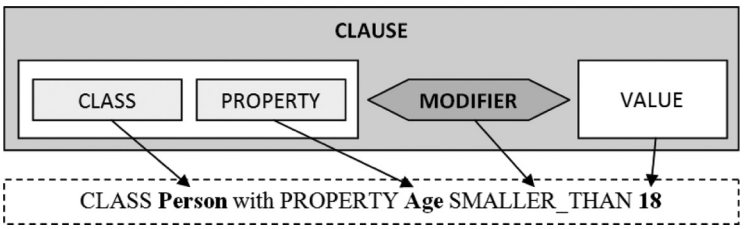

FIGURE 4 Structure of a clause.

The first two elements are used to identify the concept of the knowledge base that the clause refers to. The modifier is used to carry out a comparison between the value in the knowledge base and the value specified by the rule. A modifier may be either SMALLER_THAN, GREATER_THAN, or EQUAL_TO.

Our system uses the expert knowledge contained in the rules to infer a diagnosis for a certain patient. In order to do so, the system must launch the necessary queries and check whether the conditions stated by the rules are fulfilled for that patient.

Query System

The implementation of the query system supports logical operators, according to the fourth property stated in Toro et al. (2008). Logical operators (AND, OR, and NOT) provide a way to combine simple queries and construct complex queries using Boolean logic.

When a complex query is made over the ontology, the system splits the query into simple queries. The answers to the simple or atomic queries are retrieved and the answer to the complex query is inferred.

Figure 5 illustrates the query process that takes place in an RO-based system.

When a query is launched over the ontology, the system checks its complexity. If a complex query is made, the system splits the query into simple queries using a query parser. Then, the system will search for simple queries along with the original (complex) query through reflexivity instances in order

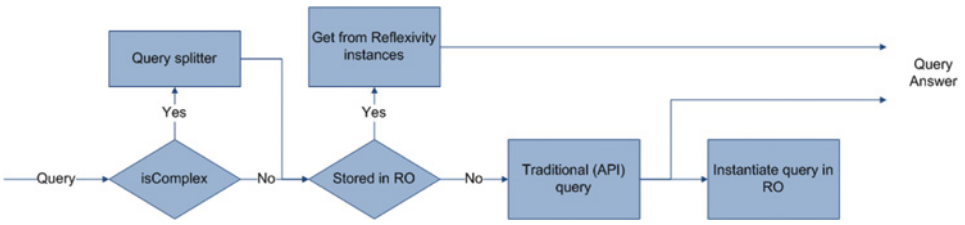

FIGURE 5 Flow of the query process in reflexive ontologies (color figure available online).

to check whether the queries have been made previously. If the system finds a reflexive instance for a certain query (or a similar one if syntactic similarity is used), its answer is retrieved directly from that instance. When the query is not present in the ontology, a traditional query is made via a Java-compliant API (Application Programming Interface). In addition, the query is instantiated in the reflexive structure, storing its answer within that instance.

A complex query formed by three simple queries will be

$$Q_c = Q_1 \text{ op } Q_2 \text{ op } Q_3, \text{ where op} = \{\text{AND, OR, NOT}\}$$

At the end of the querying process for Q_c, the reflexive structure will store four query instances: one for each simple query composing the query (Q_1, Q_2, and Q_3) and another one for the original query itself (Q_c).

Autopoiesis

As seen in the previous section, the autopoietic property of an RO refers to its ability to regenerate itself according to external perturbations. This property is closely related to the integrity update property, because any change in the ontology may be reflected in the reflexive structure in order to maintain data consistency.

When an external perturbation (modification of an individual of the ontology for instance) takes place, the system can act in different ways. During the design and implementation process of the system, different paradigms have been considered.

One of the paradigms is based on the detection of modified instances and then removing the RO queries related to the class of the modified instance. This method invalidates the whole list of related individuals that the RO query contains; that is, a single individual penalizes the entire set. In exchange, the computational cost of this method is relatively low.

Another approach is based on updating the list of RO queries and modifying the list of related individuals of the corresponding RO query. In the following pseudocode the implemented algorithm is shown (note that the code has been considerably simplified for easier understanding):

```
[Precondition: The modified individual is known]
function updateROQueries(modified_individual)
{
modified_class = class of the modified individual
RO_queries = all the queries in reflexive structure
    for each ROQuery in RO_queries
    {
    ROQuery_class = class of the ROQuery
if modified_class equals ROQuery_class
{
```

```
array_individuals = individuals pointed by ROQuery
    for each individual in array_individuals
    {
    if individual equals modified_individual
    {
        remove individual from array
        if individual fulfils the condition of the query
        add modified_individual to the array
    }
    }
}
}
}
```

This approach has a relatively high computational cost because it requires performing precaching work; therefore, its use is only recommended in an environment where data have limited variability.

Preliminary Work Regarding Self-Reasoning over the Query Set

In this work, a preliminary attempt was made to efficiently manage the RO query structure by means of the implicit knowledge underlying the query set. In this first approach we designed a method with which the query set can be modified depending on the use made of the various classes of the ontology.

Based on the assumption that queries are not made randomly and therefore follow certain patterns (recurrence in the domain), we have designed a simple system to measure the use of each class and assign a factor accordingly. For a set of queries $Q = \{q_1, q_2, \ldots, q_p\}$ and a list of existing classes $\{C_1, C_2, \ldots, C_n\}$, a factor F_i is assigned to each class C_i depending on the number of RO queries that refer to that class:

$$R_i = \{q \in Q \mid q \text{ is related to } C_i\}$$

$$F_i = \frac{1}{p} \mid R_i \mid,$$

where a query q_i expressed formally as $q_i = (\forall p \in O | p \# x)$is related to a class C_i if property p belongs to an instance of C_i, where O is an ontology, $\#$ is an operator, and x is a value.

Once the factors of each class have been calculated (referred as an *occurrence factor* or $\hat{\boldsymbol{o}}$), a threshold value can be arbitrarily established. In case the modified individual belongs to a class with a factor lower than the threshold, the RO query will be removed. On the other hand, if the modified individual belongs to a class with a factor that is equal to or higher than the

threshold (i.e., the query has been made more often), the RO query will be updated.

However, this is still a work in progress and thereby it should be considered as a first approach to the use of implicit knowledge in the reflexive structure. However, future work is expected to explore these techniques in depth in order to provide more refined solutions.

EVALUATION

Methodology

For the evaluation of the results the execution time of the diagnosis process was measured for a given number of patients. The goal was to compare the differences in execution time between a system using a conventional ontology and a system using an ontology enhanced with reflexivity.

The system was implemented using three different rule sets as shown in Table 1. For each rule set the number of contained rules is shown, along with the number of queries and the number of RO queries that were generated inside the reflexive structure. The number of generating query instances depends on the number of simple queries that each rule contains.

In order to compare the differences in performance, the system was implemented in two different ways: (1) using ROs and (2) not using ROs (referred as no-RO).

Execution time was gathered following a methodology that is similar to that used in knowledge base system benchmarking (Guo et al. 2004; Ma et al. 2006; Bock et al. 2008). Nevertheless, our evaluation process differs from other approaches because our measures are not taken from the execution of individual queries but from the entire diagnosis process. This fact, however, does not detract from the validity to our evaluation, because the diagnosis process itself can be regarded as a sequence of queries.

For testing purposes we randomly selected 10 patients who were subjected to diagnostic tests in each of the system configurations. Execution time was measured using built-in Java methods and every measurement was the average of 10 independent executions.

TABLE 1 Size and Characteristics of the Rule Sets

Rule set	Number of rules	Number of queries	Generating query instances
RuleSet 1	35	120	72
RuleSet 2	103	339	246
RuleSet 3	138	459	291

Test Environment

We performed the experiment on a desktop Intel Core 2 Quad CPU Q8300 at 2.5 GHz × 4, 2.9 GB RAM, and Ubuntu 11.10 64-bit. The system was developed and evaluated in Eclipse 3.7.0 with JDK version 1.6.0. Protégé (Protégé 2007) API was used for ontology access and management.

Data and Analysis

For each of the 10 patients, the time expended on the diagnostic process was measured in both systems: the one using reflexivity and the classical one. Table 2 shows a complete list of the test results (execution time is shown in milliseconds). In both systems the three rule sets shown in Table 1 were used.

Table 2 shows that, using the same rule set, execution times varied slightly between patients. This occurred with all of the sets of rules in both configurations; that is, using RO and no-RO. This was because the number of queries to be performed at diagnosis time was determined by the complexity of each rule in the rule set. The rule set remained constant for every patient; thus, the queries to be performed were equal for the whole group of patients. A preliminary analysis suggested that small variations were due to differences in the number of clinical tests between patients.

A reduction in the execution time needed to perform the diagnosis was evident when it comes to RO. This is shown in Figure 6 where the average execution times are compared.

TABLE 2 Execution Times in Milliseconds

		RuleSet 1	RuleSet 2	RuleSet 3
Patient 1	RO	825	1,813	2,541
	no-RO	2,715	4,139	6,343
Patient 2	RO	809	1,888	2,522
	no-RO	2,688	4,107	6,329
Patient. 3	RO	771	1,882	2,488
	no-RO	2,788	4,100	6,290
Patient 4	RO	858	1,923	2,538
	no-RO	2,752	4,165	6,298
Patient. 5	RO	972	1,932	2,553
	no-RO	2,729	4,207	6,281
Patient 6	RO	772	1,869	2,637
	no-RO	2,705	4,166	6,229
Patient 7	RO	827	1,831	2,615
	no-RO	2,726	4,097	6,238
Patient 8	RO	878	1,989	2,524
	no-RO	2,720	4,130	6,241
Patient 9	RO	767	1,976	2,436
	no-RO	2,781	4,181	6,240
Patient 10	RO	785	1,886	2,537
	no-RO	2,802	4,213	6,146

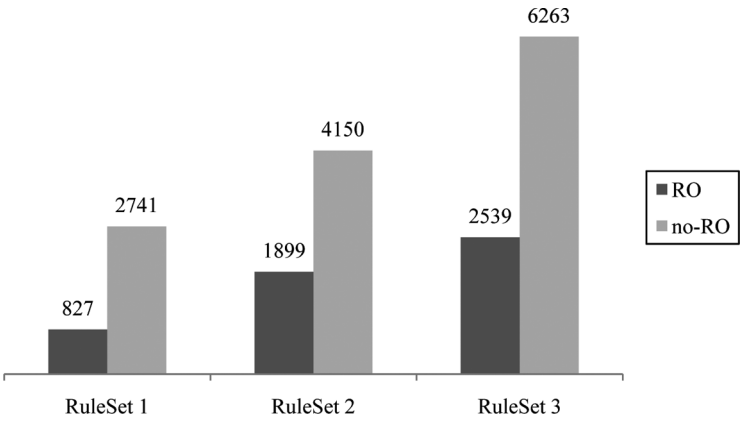

FIGURE 6 Average execution times.

Figure 6 shows the execution times corresponding to each of the rule sets. These values were obtained by calculating the arithmetic mean of the execution times measured for the 10 patients under consideration. The results showed that the use of RO significantly reduced the execution times (69.8% for RuleSet1, 54.2% for RuleSet2, and 59.4% for RuleSet3).

Comparing the execution times of both systems in relation to the number of queries may provide valuable information. Figure 7 shows the

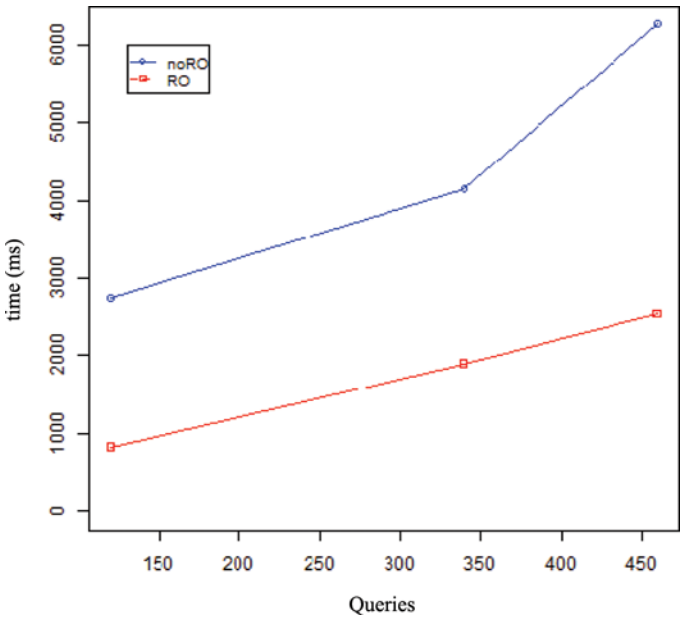

FIGURE 7 Execution time in relation to the number of queries in the rule set (color figure available online).

evolution of the execution times as the number of queries to be performed increased. The number of queries contained in the set of rules had a major impact on the performance of both systems, because this number determined the time required to perform the whole diagnostic process.

The chart in Figure 7 shows that the execution times for RO grew linearly. However, the execution times for the conventional ontology grew, if not exponentially (three points are not enough to accurately extrapolate this data), then still faster than RO. This means that, compared to a conventional ontology, an RO is more robust in terms of scalability, because the execution time grew significantly slower in relation to the number of queries.

Figure 8 shows the evolution of the execution times in relation to the number of rules in the rule set. Although the trend was similar to that presented in Figure 7, the number of rules was not as significant as the number of clauses they contained, because the complexity of the rules (in terms of number of clauses) was varied. This means, for instance, that a rule containing ten clauses is equal to a rule that contains just two when the computational cost of their processing is clearly uneven.

Both charts show that the reduction in execution time was fairly pronounced when reflexivity was used. The difference in growth between the two smallest sets of rules was similar, even if the growth of the conventional ontology was about 40% greater than that of the RO. Nevertheless, the difference was more

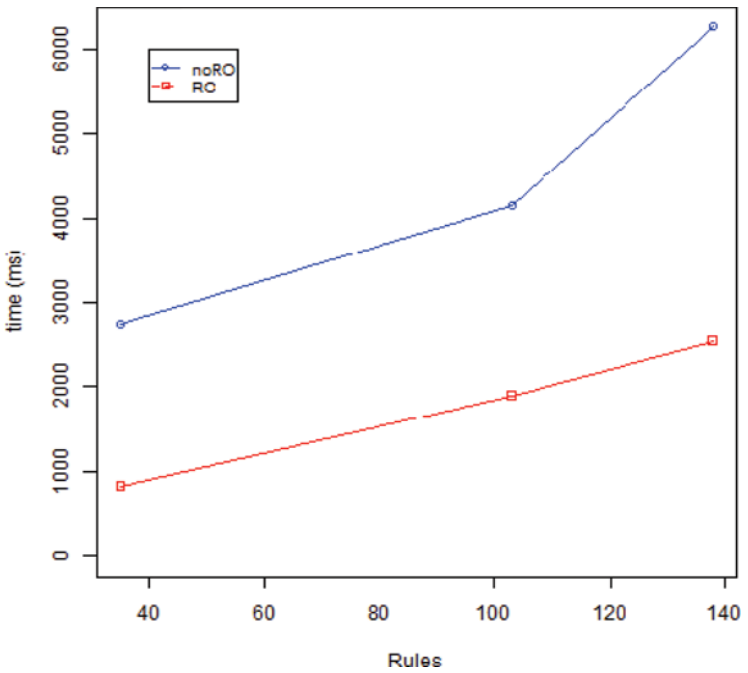

FIGURE 8 Execution time in relation to the number of rules in the rule set (color figure available online).

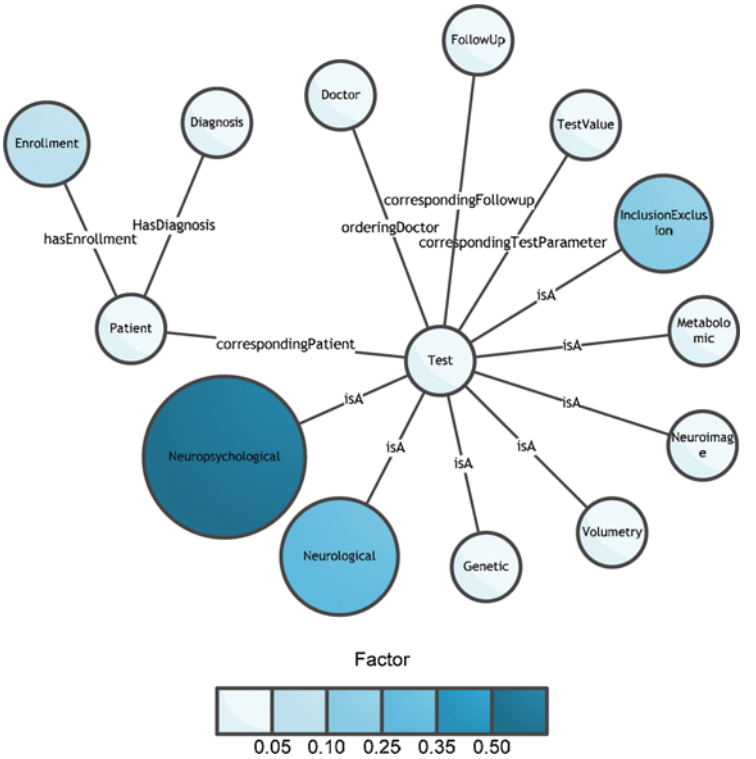

FIGURE 9 Graph representing MIND ontology (color figure available online).

evident when the biggest rule set was involved. Figures 7 and 8 show an increase of more than 2,100 ms when the size of the rule set grew to 35 rules (containing 120 queries). Under the same conditions, the execution time increased 640 ms if the ontology was enhanced with reflexivity; that is, 69.7% less.

From the obtained results, in addition to the exposed conclusions, further information can be extracted. As presented in previous sections, the analysis of RO queries provides valuable information such as the occurrence factor ($\hat{\boldsymbol{o}}$) of the classes. Figure 9 shows the graph of our ontology, where both the size and color nodes refer to the occurrence factor of a class.

Taking into account that the occurrence factor is directly related to the number of times that a class appears in a query, the larger and darker a node is, the more often it was queried. Therefore, Figure 9 shows that the class representing neuropsychological tests was the most frequently queried, followed by the one representing neurological tests. This could lead to an interesting second-order analysis of a qualitative nature, meaning that for the example at hand the domain experts were strongly biased toward neurology. In addition, it could be used to make interesting queries and assumptions when one asks why some classes received fewer (or even none) queries. Should they be removed from the knowledge model?

CONCLUSIONS

In this article we presented an implementation of the RO in a knowledge-based CDSS for the diagnosis of Alzheimer's disease. In order to measure the impact of RO in the overall performance of the implemented system, we presented a benchmark that compares two systems, one using RO and the other using a conventional ontology.

Our comparative evaluation suggests that, in the worst-case scenario, the performance of an RO is comparable to that of traditional ontologies. It also shows that in our diagnosis system, the use of RO significantly improved efficiency, reducing the execution time by almost 70% in some cases.

We presented a first approach for implementation of the *reasoning over the query set* concept. It allows the extraction of knowledge about the use of different parts of the ontology, enabling more efficient management of the queries. In order to accomplish this, we defined an occurrence factor ($\hat{\boldsymbol{o}}$) as a quantitative measurement of the recurrence of queries of certain terms in the ontology (class).

Future work will explore the design of more refined algorithms in depth in order to extract implicit knowledge from the reflexive structure, so that the list of RO queries can be optimized by means of precaching techniques.

Additionally, we plan to extend this work in order to evaluate the performance of ROs in different domains and use cases. In future evaluations we want to measure the impact (in terms of computational cost) of autopoiesis, so that the efficiency of the above-mentioned algorithms can be estimated.

REFERENCES

Altinel, M., Q. Luo, S. Krishnamurthy, C. Mohan, and H. Pirahesh. "Dbcache: Database Caching for Web Application Servers." In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 612–612. ACM, 2002.

Amir, E. and S. McIlraith. "Partition-Based Logical Reasoning for First-Order and Propositional Theories." *Artificial Intelligence* 162, (2005): 49–88.

Amiri, K., S. Park, and R. Tewari. "DBProxy: A Dynamic Data Cache for Web Applications." Paper presented at 19th IEEE International Conference on Data Engineering, Bangalore, India, March 8, 2003.

Bock, J., P. Haase, Q. Ji, and R. Volz. "Benchmarking OWL Reasoners." Paper presented at the International Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (AREa 2008), Tenerife, Spain, June 1, 2008.

Chandrasekaran, B., R. Josephson, and V. R. Benjamins. "What are Ontologies, and Why Do We Need Them?" *Intelligent Systems and Their Applications* 14, no. 1 (1999): 20–26.

Cobos, Y., C. Toro, C. Sarasua, J. Vaquero, M. Linaza, and J. Posada. "An Architecture for Fast Semantic Retrieval in the Film Heritage Domain." Paper presented at the 6th

International Workshop on Content-Based Multimedia Indexing (CBMI), London, UK, June 2008.

Grau, B. C., B. Parsia, E. Sirin, and A. Kalyanpur. "Automatic Partitioning of OWL Ontologies Using E-Connections." Paper presented at the International Workshop on Description Logics (DL2005), Edinburgh, Scotland, July 2005.

Gruber, T. R. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing." *International Journal of Human–Computer Studies* 43, nos. 5–6 (1995): 907–28.

Guo, Y., Z. Pan, and J. Heflin. "An Evaluation of Knowledge Base Systems for Large OWL Datasets." Paper presented at the International Semantic Web Conference, Hiroshima, Japan, November 7–11, 2004.

Kollia, I., B. Glimm, and I. Horrocks. "SPARQL Query Answering over OWL Ontologies." Paper presented at the 8th Extended Semantics Web Conference (ESWC'11), Heraklion, Greece, May 29–June 2, 2011.

Ma, L., Y. Yang, Z. Qiu, G. Xie, Y. Pan, and S. Liu. "Towards a Complete OWL Ontology Benchmark." *Lecture Notes in Computer Science* 4011, (2006): 125–39.

Maturana, H. and F. Varela. "Autopoiesis and Cognition: The Realization of the Living." In *Autopoiesis and Cognition: The Realization of the Living*. pp. 78–79, Dordrecht, Holland: D. Reidel Publishing Company, 1980.

McGuinness, D. L. "Ontologies Come of Age." In *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, edited by D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, pp. 171–193, Boston, MA: MIT Press, 2003.

Protégé. 2007. "Using the Protégé-OWL Reasoner API." Available at: http://protege.stanford.1edu/plugins/owl/api/ReasonerAPIExamples.html (accessed May 5, 2012).

Sanchez, E., C. Toro, A. Artetxe, M. Graña, E. Carrasco, and F. Guijarro. "A Semantic Clinical Decision Support System: Conceptual Architecture and Implementation Guidelines." In *Advances in Knowledge-Based and Intelligent Information and Engineering Systems*, *Frontiers in Artificial Intelligence and Applications* 243, edited by M. Graña, C. Toro, J. Posada, R. J. Howlett, and L. C. Jain, 1390–99. Amsterdam, Netherlands: IOS Press, 2012.

Sanchez, E., C. Toro, E. Carrasco, P. Bonachela, C. Parra, G. Bueno, and F. Guijarro. "A Knowledge-Based Clinical Decision Support System for the Diagnosis of Alzheimer Disease." In *2011 IEEE 13th International Conference on e-Health Networking, Applications and Services*, Columbia, MO: Institute of Electrical and Electronics Engineers, 2011.

Tolia, N. and M. Satyanarayanan. "Consistency-Preserving Caching of Dynamic Database Content." In *Proceedings of the 16th International Conference on World Wide Web*, 311–20. Banff, Alberta: ACM, 2007.

Toro, C., E. Sanchez, E. Carrasco, L. Mancilla-Amaya, C. Sanín, E. Szczerbicki, M. Graña, P. Bonachela, C. Parra, G. Bueno, and F. Guijarro. 2012. "Using Set of Experience Knowledge Structure to Extend a Rule Set of Clinical Decision Support System for Alzheimer's Disease Diagnosis." *Cybernetics and Systems* 43, no. 2 (2012): 81–95.

Toro, C., C. Sanín, E. Szczerbicki, and J. Posada. "Reflexive Ontologies: Enhancing Ontologies with Self-Contained Queries." *Cybernetics and Systems* 39, (2008): 171–89.