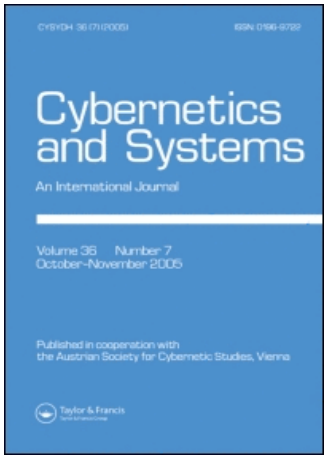

Cybernetics and Systems
An International Journal
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713722751

REFLEXIVE ONTOLOGIES: ENHANCING ONTOLOGIES WITH SELF-CONTAINED QUERIES

Carlos Toro [a]; Cesar Sanín [b]; Edward Szczerbicki [b]; Jorge Posada [a]
[a] VICOMTech Research Centre, Donostia, San Sebastían, Spain
[b] University of Newcastle, Newcastle, Australia

Online Publication Date: 01 February 2008
To cite this Article: Toro, Carlos, Sanín, Cesar, Szczerbicki, Edward and Posada, Jorge (2008) 'REFLEXIVE ONTOLOGIES: ENHANCING ONTOLOGIES WITH SELF-CONTAINED QUERIES', Cybernetics and Systems, 39:2, 171 - 189
To link to this article: DOI: 10.1080/01969720701853467
URL: http://dx.doi.org/10.1080/01969720701853467

Cybernetics and Systems: An International Journal, 39: 171–189

ISSN: 0196-9722 print/1087-6553 online
DOI: 10.1080/01969720701853467

REFLEXIVE ONTOLOGIES: ENHANCING ONTOLOGIES WITH SELF-CONTAINED QUERIES

CARLOS TORO[1], CESAR SANÍN[2], EDWARD SZCZERBICKI[2], and JORGE POSADA[1]

[1]VICOMTech Research Centre, Donostia, San Sebastían, Spain
[2]University of Newcastle, Newcastle, Australia

In this article, we introduce the concept of *reflexive ontologies*. A reflexive ontology is a description of the concepts and relations in a domain with self-contained queries. This approach presents several advantages; (1) the speeding of the query process; (2) the addition of extra knowledge about the domain extending it with queries and answers; and (3), the self-containment of the knowledge structure. We present a framework that can be used to extend any existing ontology with the reflexivity approach. Additionally, as case study, we test the architecture with a previously presented knowledge structure called Set of Experience Knowledge Structure (SOEKS).

INTRODUCTION

Semantic technologies constitute one of the most interesting technologies derived from the World Wide Web revolution. Constantly reviewed

This research has been partially financed by the Basque government under the INTEK 2006–2008 call (Bi2Hiru). A special mention is given to the Faculty of Engineering and Built Environment of the University of Newcastle (NSW, Australia) for their involvement in the development of the RO concept and the priceless ideas they shared concerning the Set of Experience Knowledge Structure and the decisional DNA paradigms.

Address correspondence to Carlos Toro, VICOMTech Research Centre, Paseo Mikeletegi 57, bajo, Donostia, San Sebastian, Spain. E-mail: ctoro@vicomtech.org

in different areas of knowledge (e.g., linguistics), their greatest improvements for information technologies could be still there to discover.

It is true that the whole concept of the semantic web presented by Tim Berners-Lee (Berners-Lee 2001) in his foundational article has not been reached yet, according to some members of the scientific community, but the improvements present in today's Web sites and search engines are not to be underestimated.

Within the myriads of semantic-based techniques available, great attention has been given to ontologies and how their implementation and use enhance real-world applications that are not directly related to the Web itself. One of the biggest advantages of ontologies is their flexibility and capability to model a domain and, hence, conceptualize the portion of reality to which such a domain refers.

It is not enough to have a good modeled ontology fed with real-world instances (individuals) from trustable sources of information. Nowadays, it is of the utmost importance to enhance the ontologies with the capability of querying their knowledge models in a fast and trustable way.

In this article, we introduce the concept of reflexive ontologies as a technique that can be used to add self-contained queries to an ontology.

The advantages of having self-contained queries include (1) the speeding up of the query process; (2) the possibility of the ontology itself adding new queries on individuals with the corresponding answers to such queries (a feature that adds knowledge about the domain); and (3) the self-containment of the knowledge structure in a single file, including the model, the relations between the elements of the model, the individuals (instances), and queries over such individuals.

This article is organized as follows: In the next section, we present some background concepts that will be referenced throughout the article. Next, we introduce the reflexive ontologies concept, discussing some of its characteristics, advantages, and implementation using a simple architecture that can be followed in order to add reflexivity to a traditional ontology. We present also, in this part, a query engine for reflexive ontologies. Following, we introduce a case study that uses a previously presented knowledge base (the SOEKS ontology) to exemplify the reflexive ontologies schema. Finally, we discuss our conclusions and future work.

BACKGROUND CONCEPTS

Reflexivity

The concept of reflexivity is used in multiple fields; in this work, we approach the reflexivity concept from the mathematics (logic) point of view and the sociological point of view.

Mathematical Concept of Reflexivity

Reflexivity, in mathematics, refers to the logic idea of $p \rightarrow p$ (it is read as "p implies p"), meaning that every proposition implies itself.

Any relation in mathematics is referred as a subset of a Cartesian product. For instance, a subset of $A \times B$, called a "binary relation from A to B," is a collection of ordered pairs (a, b) with first components from A and second components from B; and, in particular, a subset of $A \times A$ is called a "relation on A." For a binary relation R, one often writes aRb to mean that (a, b) is in R.

The reflexivity property can be defined as a relation R on a set S. Such a relation is reflexive provided that xRx exists for every $x \in S$.

In our approach, a mathematical simile corresponds well to the self-containing nature of the reflexive ontology as it will be explained later. The mathematical simile will permit us in the future to express in a structured way lemmas and demonstrations useful for the formalization of the concept.

Sociological Concept of Reflexivity

The mathematical idea is not distant from the definition used in sociology, where the concept of reflexivity takes on an epistemological flavor as it is used to identify the foundations of knowledge and the implications of any findings.

Reflexivity (2007), in sociology, is the action of self-referencing and refers and affects the object by means of examining or acting on the object itself.

It is a bidirectional relationship between cause and effect. In such a case, the reflections of the object about itself are not independent of its status quo as a reflexive object. Moreover, any object or agent in a real-world social system possesses characteristics of reflexivity and self-inquiry. Thus, the object or agent being reflected discovers or determines something about itself and abstracts out that aspect. It may reflect

on beliefs, memories, or knowledge, but this reflection does not produce that belief, memory, or knowledge.

Flanagan (1981) argues that reflective agents support the traditional roles played by classical science: control, explanation, and prediction. In order to help in the control, explanation, and future prediction of domains, our approach uses the capacity of self-interrogation and, hence, the capacity of obtaining conclusions, which are elements characterized by the sociological point of view.

Autopoiesis

Autopoiesis literally means "self-creation or auto-creation." It expresses a fundamental dialectic between structure and function. The term was originally introduced by Maturana and Varela (1980), and according to them, an autopoietic system represents a network of processes or operations that define the system and make it distinguishable from others.

Any autopoietic system is able to create and destroy elements of the system itself in response to environmental perturbations. Although the system changes at a structural level, the network is invariant along the system's existence, holding the inner system integrity.

The auto-production capacity of a system is autopoietic and constitutes the basic property of living creatures, as they are always determined by their structures; in other words, they are systems such that when an external factor affects them, the resulting effects depend solely on themselves, on their structure at that moment and not on the external factor itself. Living creatures are autonomous in the sense that they have an auto-referencing property as systems in continuous production of themselves. The most important thing for this theory is not the properties of the components of the system but the processes and relationships between processes made via their components.

Luhmann (1997) argues that autopoiesis is not a limited property of biological or physical systems, and he defines it as the "universal capacity of every system to produce self states well differenced between each other that are tied to the system's own operations due to the self-organization capacity of systems."

The structure and function relation of the autopoietic definition allows us to refer to our reflexive ontology as an autopoietic system that is in constant evolution in the sense that every new query being asked and stored will extend the knowledge base.

Ontologies

Following Tom Gruber's (1995) widely accepted definition of *ontology* in the computer science domain, an ontology is the explicit specification of a conceptualization—a description of the concepts and relationships in a domain. In the context of artificial intelligence (AI), we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate names of entities in the universe of discourse with human-readable text describing what the names mean and formal axioms that constrain the interpretation and well-formed use of these terms.

Computer programs can use ontologies for a variety of purposes including inductive reasoning, classification, and problem-solving techniques, as well as communication and sharing of information among different systems. In addition, emerging semantic web systems use ontologies for a better interaction and understanding between different agent Web-based systems. Ontologies can be modeled using several languages; the most widely used are RDF and recently OWL (Ontology Web Language).

User modeling, task, and experience are also possible scenarios for the exploitation of semantic data by ontology-based technology as it was addressed, for example, in the European IST-Project WIDE (Sevilmis et al. 2005).

In general terms, any knowledge is susceptible to being modeled as an ontology. However, no normative exists yet for modeling knowledge. This could be due to the inner nature of the object to be modeled, as it is different from one schema to another. Although there are interesting approaches for a universal ontology, this outcome has not been reached yet due to the difficulty of standardization and the fact that if a universal modeling of knowledge is reachable, it will lead to a high degree of conceptualization, which could create difficulty for an inexperienced user.

From the experience learned by different working groups in the ontology modeling of elements, a good starting point is to have a well-documented schema with the typical elements in the area of knowledge that is being described.

In order to model an ontology, different tools are available—for example, Protégé, OilEd, Ontololingua, and Swoop, among others (Toro et al. 2007).

In our implementation, we used the Protégé ontology editor and its APIs, but any other editor or API can be used as well.

Ontology Query

One of the advantages of a conceptual knowledge model expressed as an ontology is the capacity to semantically infer new derived queries. These queries relate concepts that are not explicitly specified by the user but are nevertheless relevant to the query. Modern inference engines and reasoners like Pellet and Racer deliver a highly specialized yet efficient way to perform such queries via a JAVA-compliant API. In the literature, data handling by ontology-based technology is reported by researchers in different fields (Toro et al. 2007; Sanin et al. 2005a; Sevilmis et al. 2005).

REFLEXIVE ONTOLOGIES

In this section, we introduce the reflexive ontologies (RO) concept. We begin with a definition of the RO concept; then we discuss the RO properties and introduce an architecture that allows an existent ontology to be extended with the RO schema. Lastly, we finish our line of thoughts with a reflection on the advantages of having a RO-compliant ontology.

Formal Definition

In our case, the de facto property of being reflexive addresses the property of an abstract structure of a knowledge base (in this case, an ontology and its instances) to "know about itself." When an abstract knowledge structure is able to maintain, in a persistent manner, every query performed on it and store those queries as individuals of a class that extends the original ontology, it is said that such an ontology is reflexive.

Thus, we propose the following definition for a reflexive ontology:

> 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.

Considering that any abstract knowledge structure of this kind is essentially a set of structured contents and relationships, the mathematical concept of a set and its properties can be applied to the knowledge structure for its formalization and handling.

Properties of the Reflexive Ontologies

A RO is, basically, an ontology that has been extended with the concept of reflexivity. This can be seen in Fig. 1, where C_i represents a class with r_j relations, I_k characterizes an instance of C_i (right side of the image),

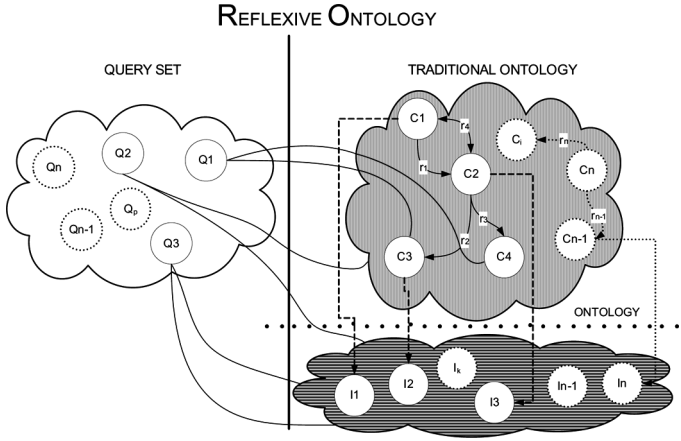

Figure 1. Simple structure of a RO.

and Q_p is a query represented as an extension of the base ontology (left side of the image). In order to be compliant with the RO concept, an extension of a base ontology must fulfill the following set of properties.

Property 1—Query Retrieval. This property refers to the RO faculty of storing every query performed in order to return it, when requeried. The query storage happens at two different levels: it could be in its atomized (simple query) or complex form (i.e., containing a Boolean operator between atoms).

Queries can refer to the structure of the ontology (data type) or to the instances (value type).

Property 2—Integrity Update. This property refers to the RO faculty of updating structural changes in the query retrieval system. In other words, when a new individual is added or removed from the ontology, the query system actualizes the queries that contain such an individual.

This property is similar to the database integrity property in a traditional Data Base Management System (DBMS).

Property—Autopoietic Behavior. This property refers to the autopoietic capacity of self-creation or auto-creation derived from the fact that for every new query launched, the knowledge structure will grow. The quality of the knowledge embedded in the system is increased as the system is in a constant auto-building process that can reflect the auto-production capacity of an autopoietic system. Moreover, it is

important to mention that this property allows the RO to store the history of the queries while, at the same time, it provides the schema with a repeat ability and integrity mechanism.

Property—Support for Logical Operators. This property provides the RO with the mechanisms of set handling from the logistic point of view—that is, the inclusion of AND, NOT, and OR logical operators.

For example, let us suppose that we have a simple ontology (O) that describes the genealogy of the Jones family with some properties (p) such as the age of every individual ($p.age$). After the instantiation with some elements, a query with two elements can be launched:

Elements whose age is greater than 5 expressed formally as Q_1
$= (\forall p \in O | p.age > 5)$and elements that belong to the Jones family Q_2
$= (\forall p \in O | p.family = ''jones'')$

A query like "retrieve the members of the Jones family whose age is greater than 5 years" on the ontology O will give as a result a set of individuals that match the abstract properties **p.age>5** and **p.family=jones**.

So, the final answer to the query called Q_t (indistinctively reflexivity) will be

$$Q_t = Q_1 \cap Q_2,$$

where Q_1 and Q_2 are defined as above, with p being the predicate and O the ontology.

A question like this will be instantiated in the ontology in its atomized form (Q_1, Q_2) and, of course, in the complex form Q_t. Therefore, next time a question is asked, the system will search through the reflexivity instances in order to retrieve the answer (being the computing time at the worse case linear). If the exact question or (as it will be seen later) a similar question cannot be found (similarity), a traditional ontology interrogation process is executed.

Property 5—Self-Reasoning over the Query Set. This property refers to the RO capacity of performing some logic operations over the query system in one of the following schemas:

1. To discover patterns of queries.
2. To suggest the need of ontology refinement, as some elements are more queried than others; this means that, probably, they should be

taken into special consideration as the queries performed are being focused on specific sections of the query system that cause a possible asymmetric behavior.

3. To discover new nonexplicit relationships, meaning that some queries could be different, but their sets of solutions are the same. Hence, this could imply a possible undercover relationship between sets of queries (a possible serendipity behavior).

Advantages of Reflexive Ontologies

The advantages of having self-contained queries include the following main aspects:

1. Speed (i.e., the speeding of the query process): Because every query is handled and stored in either its atomized form or its complex form, including the logical operators, the interrogation of the ontology is in the worst case time linear if the query exists (i.e., if it has been previously stored). If the query has not been asked before, then a typical ontology interrogation via an API takes place, and when the new set of answers is retrieved, it is added to the reflexivity class. This advantage is related to the RO Property 1.
2. Incremental nature (i.e., the possibility of the ontology itself adding new queries on individuals with the corresponding answers to such queries): This is in fact a feature that adds knowledge about the domain, because the more questions that are asked, the more knowledge that can be stored in the ontology. This advantage is related to the RO Property 5. The questions and answers are in fact a guideline to infer "things" about the ontology.
3. Self-containment of the knowledge structure in a single file: This feature includes the storage of the model, the relations between the elements of the model, the individuals (instances), and queries over such individuals. This advantage relies on fulfillment of the functional purpose of a RO.

Implementation of the Reflexive Ontologies Concept

We define the following architecture for the implementation of the RO concept.

This architecture is divided into three layers (see Fig. 2). The *repositories layer* contains the real-world elements that "feed" the ontology; the information contained can be structured or unstructured, and by means

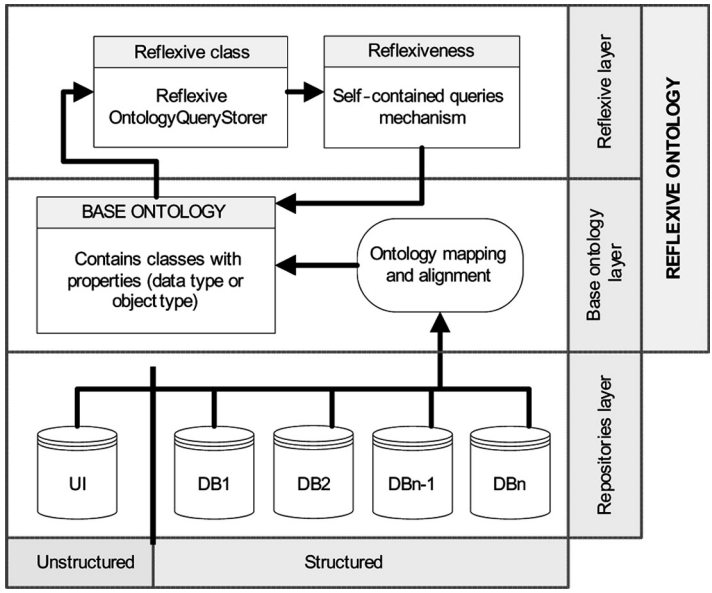

Figure 2. Reflexive ontology architecture.

of a process of ontology alignment and mapping, they become instances of the base ontology (traditional ontology approach).

In a typical ontology, the knowledge definition is represented as classes and properties of two possible types: (1) object type (i.e., mapping a class to a class) and (2) data type (i.e., mapping a class to a characteristic represented by a traditional computer type such as a string, integer, etc.).

The next layer contains two modules; the first one is the extension itself, which adds a new class to the base ontology with the needed schema for the reflexivity. In our implementation, we call this the ReflexiveOntologyQueryStorer class (see Fig. 3).

The extension hangs from the OWL: Thing superclass and has the following OWL properties (see Table 1).

The last module inside our architecture is the reflexiveness itself, and it provides the ontology (programmatically) with a mechanism to perform queries and some logic on the queries that allows the handling of the reflexiveness.

Finally, we would like to mention that in our implementation we used Protégé and its APIs and the OWL-DL subspecies of the OWL

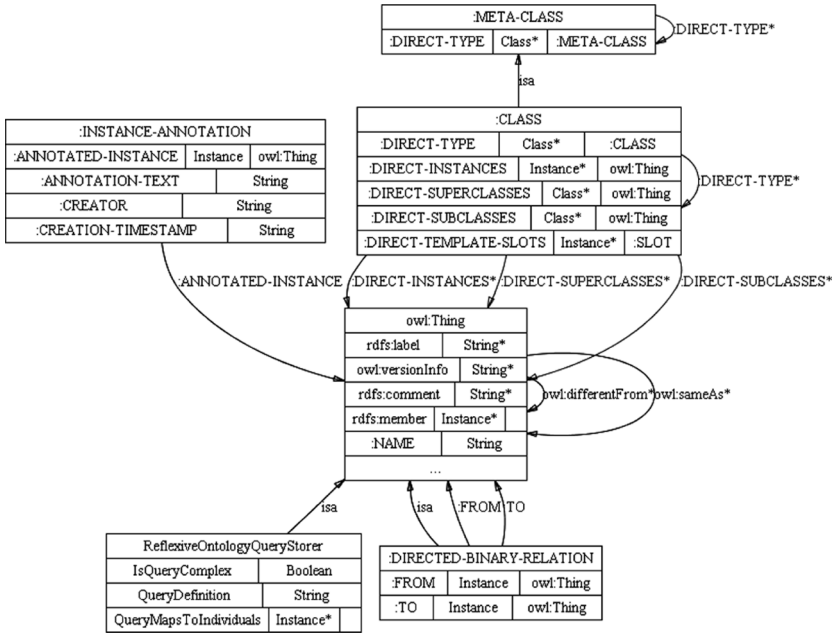

Figure 3. Reflexive ontology QueryStorer.

specification (Zhang 2005), but any other ontology modeling software that offers an open API could be used to extend an ontology programmatically with the RO concept.

CASE STUDY

In this section, we introduce a case study that uses a previously presented ontology to exemplify the reflexive ontologies schema. Such an ontology is the Set of Experience Knowledge Structure-OWL (Sanin, Toro, and Szczerbicki 2007) or SOEKS-OWL for short.

Table 1. Reflexive class properties

Property	Type	Comment
isQueryComplex	Data	Boolean
QueryDefinition	Data	String
QueryMapsToIndividuals	Object	Collection of individuals

Set of Experience Knowledge Structure

Arnold and Bowie (1985) argue that "the mind's mechanism for storing and retrieving knowledge is transparent to us. When we 'memorize' an orange, we simply examine it, think about it for a while, and perhaps eat it. Somehow, during this process, all the essential qualities of the orange are stored (experience). Later, when someone mentions the word 'orange,' our senses are activated from within (query), and we see, smell, touch, and taste the orange all over again" (p. 46). The Set of Experience Knowledge Structure (SOEKS or SOE for short) has been developed to keep formal decision events in an explicit way (Sanin and Szczerbicki 2005a). It is a model based on existing and available knowledge, which must adjust to the decision event from which it is built (i.e., it is a dynamic structure that relies on the information offered by a formal decision event); besides, it can be expressed in XML or OWL to make it shareable and transportable (Sanin and Szczerbicki 2005b, 2006a; Sanin et al. 2007). Four basic components surround decision-making events, and they are stored in a combined dynamic structure that comprises the SOE. These four components are variables, functions, constraints, and rules.

Additionally, the SOEKS is organized in a way that takes into account some important features of DNA. First, the combination of the four nucleotides of DNA gives uniqueness to itself, just as the combination of the four components of the SOE offers distinctiveness. Moreover, the elements of the structure are connected among themselves, imitating part of a long strand of DNA, that is, a gene. Thus, a gene can be assimilated to a SOE, and, in the same way as a gene produces a phenotype, a SOE produces a value of decision in terms of its objective functions. Such value of decision can be called the efficiency or the phenotype value of the SOE (Sanin and Szczerbicki 2005a); in other words, the SOEKS itself stores an answer to a query presented.

A unique SOE cannot rule a whole system, even in a specific area or category. Therefore, more SOEs should be acquired and constructed. The day-to-day operation provides many decisions, and the result of this is a collection of many different SOEs. A group of SOEs of the same category comprise a kind of decisional chromosome, as DNA does with genes. These decisional chromosomes of SOE could make a "strategy" for a category. In this case, each module of chromosomes forms an entire inference tool and provides a schematic view for knowledge inside an

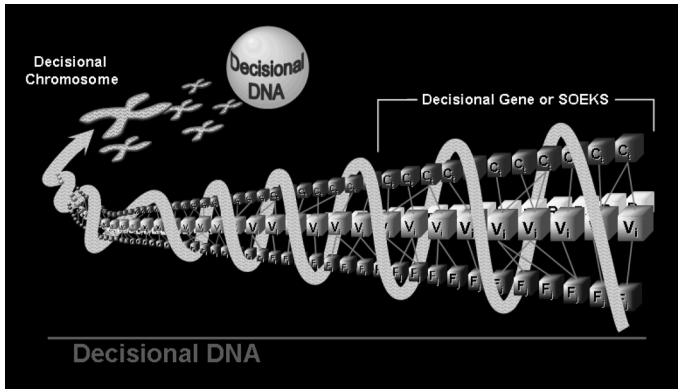

Figure 4. Decisional DNA.

organization. Subsequently, having a diverse group of SOE chromosomes is like having the decisional DNA of an organization, because what has been collected is a series of inference strategies related to such enterprise (see Fig. 4).

In conclusion, the SOEKS is a compound of variables, functions, constraints, and rules, which are uniquely combined to represent a formal decision event. Multiple SOEs can be collected, classified, and organized according to their efficiency, grouping them into decisional chromosomes. Chromosomes are groups of SOEs that can comprise a decision strategy for a specific area of an organization. Finally, sets of chromosomes comprise what is called the decisional DNA of the organization. Furthermore, the decisional DNA can be used in platforms to support decision making, and new decisions can be made based on it. In this text, a concise idea of the SOEKS and the decisional DNA was offered; for further information, the work of Sanin and Szczerbicki (2005a, 2006b) should be reviewed.

After having instantiated the SOEKS-OWL with real values, the purpose of the case study is to exemplify how the SOEKS-OWL is converted into a reflexive ontology by the use of the ReflexiveQueryStorer class and the changes it generates in such an ontology.

Initially, Fig. 5 shows the visualization of the SOEKS-OWL instantiated (base ontology as presented in the architecture). The next step comprises the adaptation of the ReflexiveQueryStorer class with some initial values such as the path of the ontology, options of saving the

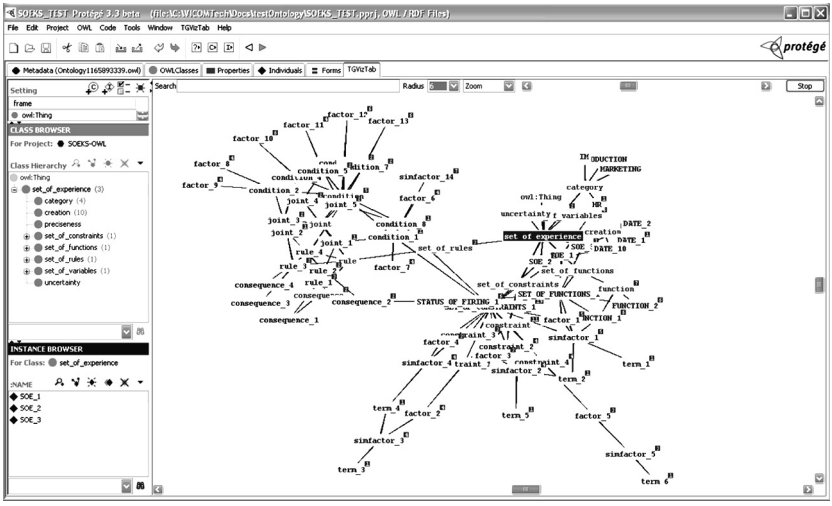

Figure 5. SOEKS-OWL instantiated.

reflexive structure and the query instances, and the type of query to be performed. This is expressed in Java code as follows:

```
public static String
    ONTOLOGY_AND_PATH="C://VICOMTech//testOntology//
      SOEKS_TEST.owl";
public static boolean
  SAVE_ONTOLOGY_WHEN_CREATE_REFLEXIVE_STRUCTURE=true;
public static boolean
  SAVE_ONTOLOGY_WHEN_INSTANTIATE_REFLEXIVE_STRUCTURE=true;
public static boolean PERFORM_SIMPLE_CHECK=true;
```

For explanation purposes, three queries will be exemplified in this case study. The first query is defined in the code as **public static** `String` *SIMPLE_RFLEXIVE_QUERY*="`CLASS variable with the PROPERTY var_name EQUALS to X1`"; notice that this is a value-type query. Such a query is written in a human-readable form, but in other terms, it means "retrieve all the variables of the ontology that have the variable name X1."

The execution of the code offers information about the type of query executed and the successful saving of the reflexive ontology structure (created for the first time) as well as the query executed with results. Following, the results can be seen as a query successfully executed with the

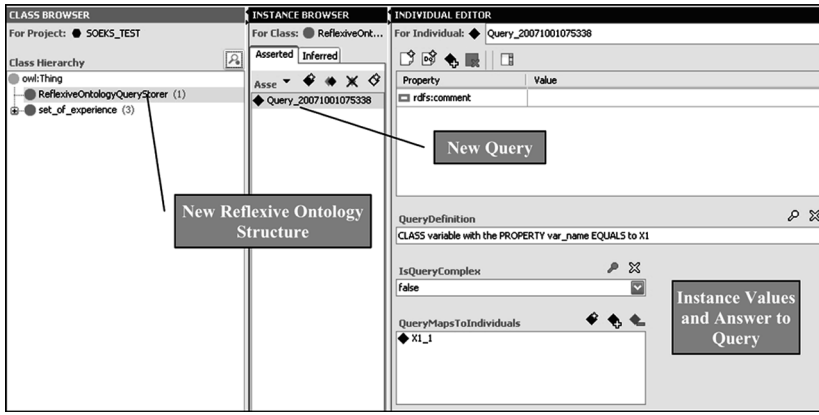

Figure 6. SOEKS-OWL transformed into a reflexive ontology with its new elements.

new instance in the SOEKS-OWL transformed into a reflexive ontology (see Fig. 6):

```
-----------------------------------------START-----------------------------------
Testing Simple query : CLASS variable with the PROPERTY
var_name EQUALS to X1
-----------------------------------------------------------------------------------
... saving successful.
File modification saved with 0 errors.
-----------------------------------------END-------------------------------------
```

The next example includes a data-type query: **public static** String *SIMPLE_RFLEXIVE_QUERY*="CLASS term with the PROPERTY withVariable EQUALS to X2_1"; in other words, it means "retrieve all the terms in the ontology that involve the variable with name X2_1." Its results are as follows (Fig. 7):

```
-----------------------------------------START-----------------------------------
Testing Simple query : CLASS term with the PROPERTY
with Variable EQUALS to X2_1
-----------------------------------------------------------------------------------
.... saving successful.
File modification saved with 0 errors.
-----------------------------------------END-------------------------------------
```

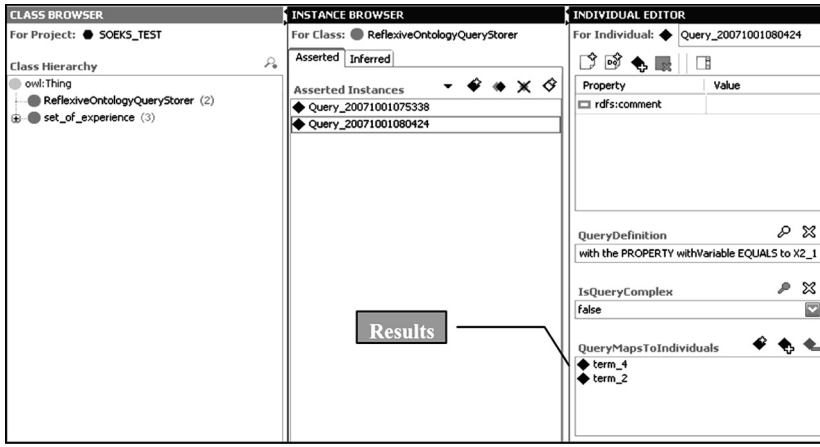

Figure 7. SOEKS-OWL with a data-type query executed.

In these results, the term_2 and term_4 are valid for the query; in other words, those terms contain the X2_1 variable. As an example, the term_2 is shown in Fig. 8.

One of the features of reflexive ontologies is that when a new query comes, it is not necessary to perform it again, if it was already done. In such a case, the reflexive ontology class will return the following answer:

```
-----------------------------------------START-----------------------------------
Testing Simple query : CLASS term with the PROPERTY
with Variable EQUALS to X2_1
---------------------------------------------------------------------------------
query has been made before
-----------------------------------------END-------------------------------------
```

The queries are stored according to the architecture presented in the query storer class that has been created. Finally, a complex query is performed comprising Property 3 (autopoietic behavior) and Property 4 (support for logical operations) (see Fig. 9):

```
-----------------------------------------START-----------------------------------
Testing Complex query : CLASS simfactor with the PROPERTY
hasSterm EQUALS to term_1 AND CLASS
simfactor with the PROPERTY hasSterm EQUALS to term_2
```

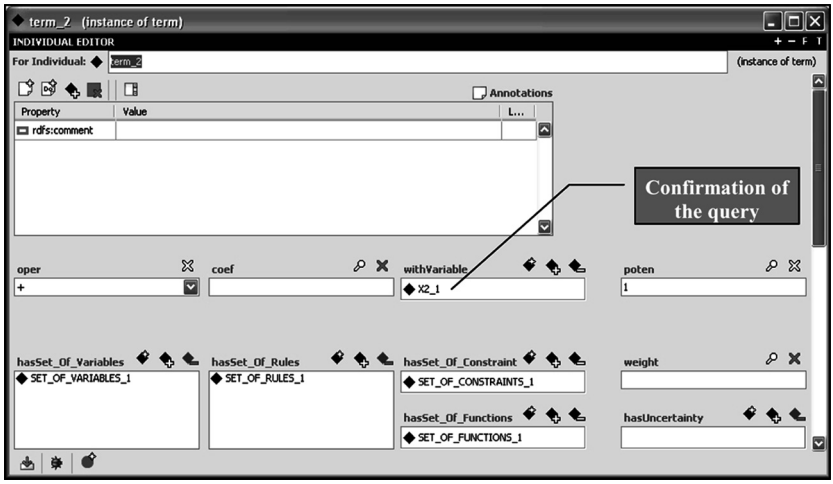

Figure 8. term_2 with confirmation of query results.

```
CLASS simfactor with the PROPERTY hasSterm EQUALS to term_1 AND
CLASS simfactor with the PROPERTY hasSterm EQUALS to term_2
Sub queries in query 2
--------------------------------------------------------------------------------
... saving successful.
File modification saved with 0 errors.
----------------------------------------END-----------------------------------------
```

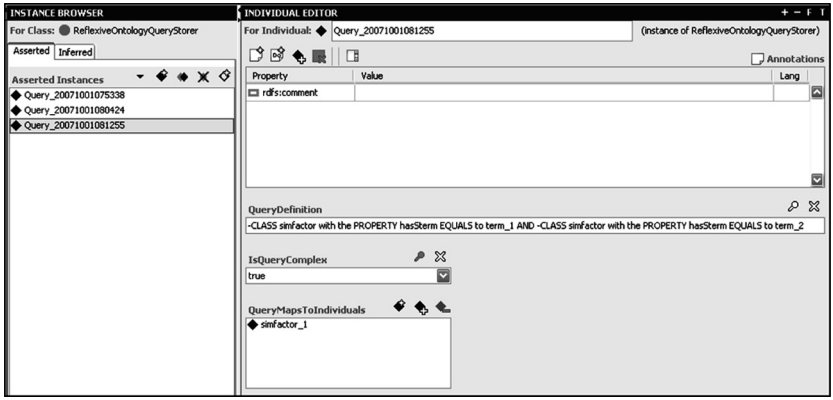

Figure 9. Complex query executed on the SOEKS-OWL.

As previously shown, the reflexive ontology transformation includes the creation of a new class inside the ontology—in this case, the SOEKS-OWL. Additionally, when different simple or complex queries (data type or value type) are executed, they are inserted as instances in the new reflexive ontology; this change facilitates the application of similarity elements among the SOEs and will allow extended logic handling over the SOEKS, as it comprises Property 5 of the RO (self-reasoning over the query set).

This step can also be seen in the architecture, as the RO is strongly attached to the base ontology in order to extend it.

CONCLUSIONS

In this article, we introduced the concept of reflexive ontologies (RO). The presented schema can be applied to an existing ontology to improve its query capabilities. We described the RO properties and benefits of having self-contained queries, and within the architecture of the RO we presented a case study that extends a previously presented OWL ontology called SOEKS.

In future work, some additional features must be developed in order to offer a more user-friendly environment: implementation of a graphical user interface (GUI) by the means of an API, which should help in the transformation of an ontology into a RO; implementation of a GUI for the query engine; and extension of the query logic elements into a more humanlike language.

In addition, a formalization from a mathematical point of view will be presented in a future work, taking advantage of the possibility to define possible theorems and lemmas that model the RO behavior.

Finally, similarity features for the decisional DNA in conjunction with the RO will improve its implementation. This will constitute a very important line of future work based on the SOEKS paradigm.

REFERENCES

Arnold, W. and Bowie, J. 1985. *Artificial intelligence: A personal commonsense journey.* New Jersey: Prentice Hall.

Berners-Lee, T., Hendler, J., and Lassila, O. 2001, May. The Semantic Web—A new form of Web content that is meaningful to computers will unleash a

revolution of new possibilities. *Scientific American*, May 2001, No. 284, pp.34–43.

Flanagan, O. J. 1981. Psychology, progress, and the problem of reflexivity: A study in the epistemological foundations of psychology. *Journal of the History of the Behavioral Sciences*, 17: 375–386.

Gruber, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* 43(5–6): 907–928.

Luhmann, Niklas R. 1997. *Organización y Decisión, Autopoiesis y Entendimiento Comunicativo.* Barcelona: Anthropos.

Maturana, H. and Varela, F. 1980. *Autopoiesis and cognition.* Boston: Reidel.

Reflexivity. 2007. In Wikipedia, The Free Encyclopedia. Retrieved September 6, 2007, from http://en.wikipedia.org/w/index.php?title=Reflexivity_%28social_theory%29&oldid=150538764

Sanin, C. and Szczerbicki, E. 2005a. Set of experience: A knowledge structure for formal decision events. *Foundations of Control and Management Sciences* 3: 95–113.

Sanin, C. and Szczerbicki, E. 2005b. Using XML for implementing set of experience knowledge structure. In *Proceedings on International Conference on Knowledge-Base and Intelligent Information and Engineering Systems—KES*, edited by R. Koshla, R. Howlett, and L. Jain., Berlin, Heidelberg: Springer, pp. 946–952.

Sanin, C. and Szczerbicki, E. 2006a. Extending set of experience knowledge structure into a transportable language extensible markup language. *Cybernetics and Systems* 37(2–3): 97–117.

Sanin, C. and Szczerbicki, E. 2006b. Using set of experience in the process of transforming information into knowledge. *International Journal of Enterprise Information Systems* 2: 45–62.

Sanin, C., Toro, C., and Szczerbicki, E. 2007. An OWL ontology of set of experience knowledge structure. *Journal of Universal Computer Science* 13(2): 209–223.

Sevilmis, N., Stork, A., Smithers, T., Posada, J., Pianciamore, M., Castro, R., et al. 2005. Knowledge sharing by information retrieval in the semantic web. In *Lecture notes in computer science.* Germany: Springer Berlin, Heidelberg, pp. 471–485.

Toro, C., Termenón, M., et al. 2007. Ontology supported adaptive user interfaces for structural CAD design. In *CIRP Conference on Digital Enterprise Technology (DET)*, edited by Cunha, Pedro Felipe, u.a.: Proceedings. Berlin; Heidelberg: Springer, p. 8.

Zhang, Z. (2005). *Ontology query languages for the semantic Web* p. 57. Master's thesis, University of Georgia, Athens.