ROZDZIAŁ 7

CESAR SANÍN, CARLOS TORO, EDWARD SZCZERBICKI
AND JORGE POSADA

ENHANCING THE SET OF EXPERIENCE KNOWLEDGE STRUCTURE AS A REFLEXIVE ONTOLOGY

1. INTRODUCTION

Semantic technologies constitute one of the most interesting technologies derived from the World Wide Web revolution. It is a field constantly reviewed in different areas of knowledge and its greatest improvements for information and knowledge technologies are still there to be discovered.

According to some members of the scientific community, it is true that the whole concept of the semantic web presented by Tim Berners-Lee in his foundational article [Berners-Lee et al. 2001] is not reached yet; however, the improvements present in today's Web sites and search engines are not to be underestimated.

Within the myriads of semantic based techniques available, a 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. Ontologies offer great flexibility and capability to model specific domains, and hence, conceptualize the portion of reality to which such domain refers to. Nevertheless, it is not enough to have a good modelled 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. On this regard, the introduction of the concept of Reflexive Ontologies (RO) by Toro et al. [2007b] offers alternatives to improve ontology querying.

The RO technique can be used to add self contained queries to an ontology and improves: *(i)* the speeding of the query process *(ii)* the possibility of the ontology itself to add new queries on individuals with the correspondent answers to such queries (a feature that adds knowledge about the domain); and *(iii)* 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.

Furthermore, the Set of Experience Knowledge Structure (SOEKS or shortly SOE) is a knowledge structure that allows the acquisition and storage of formal decision events in a knowledge-explicit. It comprises variables, functions, constraints and rules associated in a DNA shape allowing the construction of the Decisional DNA of an organization.

Having a powerful knowledge structure such as the SOEKS enhanced with the RO technique can be considered as an important advance in the development of knowledge systems.

This paper is organized as follows: in the next Section, we present some background concepts that are included along the article. Next, we discuss the Reflexive Ontologies concept and its characteristics, advantages and implementation using a simple architecture that can be followed in order to add reflexivity to a traditional ontology. Following, we introduce the implementation of the RO into the SOEKS ontology. Finally, we discuss our conclusions and future work.

2. BACKGROUND CONCEPTS

Reflexivity

The concept of Reflexivity is used in multiple fields; in regards to Reflexive Ontologies, it is approached from the mathematics (logic) point of view and the sociologic 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. Moreover, 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. Thus, the reflexivity property can be defined as a relation R on a set S. Such relation is reflexive provided that xRx exists for every $x \in S$.

In this approach, a mathematical simile is consistent with the self-contained nature of the reflexive ontology as it will be explained later.

Sociological Concept of Reflexivity

The mathematical idea is not distant from definition used in sociology, where the concept of reflexivity takes an epistemological flavour 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 the 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-enquiry. Thus, the object or agent being reflective 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 the classical science: control, explanation and prediction. In order to help in the control, explanation and future prediction of domains, the RO 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 dialect 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 among others. In general terms, any autopoietic system is able to create and destroy elements of its system itself in response to the environment perturbations. Although the system changes at a structural level, the network is invariant along the system's existence, holding the inner system integrity. Such auto-production capacity of a system 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. Furthermore, 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 among the processes made via their components.

Additionally, 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 own operations due to the self-organization capacity of systems".

The structure and function relations of the autopoietic definition allows referring to RO as autopoietic systems that are in constant evolution in the sense that every new query being asked and stored will extend the knowledge base.

Ontologies

Tom Gruber's [1995] widespread accepted definition in the Computer Science field of Ontology sates that 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 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 knowledge among different systems. In addition, emerging semantic systems use ontologies for a better interaction and understanding between different agent-based systems. Ontologies can be modelled using several languages, being the most widely used RDF and recently OWL (Ontology Web Language).

User modelling, task, knowledge 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] and Toro et al. [2007b; 2007c].

In general terms, any knowledge is susceptible to be modelled as an ontology; however, no normative exists yet in order to model knowledge. This could be due to the inner nature of the object to be modelled 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 modelling of knowledge is reachable, it will lead to a high degree of conceptualization turning into a difficulty for an inexperienced user. From the experience acquired by different working groups in ontology modelling 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.

One of the advantages of a conceptual knowledge model expressed as an ontology is the capacity of inferring semantically new derived queries. These queries relate concepts that are not explicitly specified by the user; nevertheless the concepts are 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. 2007a, Sanin et al. 2007, Sevilmis et al. 2005].

In order to model an ontology, different tools are available such as Protégé, OilEd, Ontololingua, Swoop, among others [Toro et al. 2007a]. In our implementation, we used the protégé ontology editor and its APIs, but any other editor or API can be used as well.

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 shortly) has been developed to keep formal decision events in an explicit way [Sanin & Szczerbicki 2005a]. It is a model based upon existing and available knowledge, which must adjust to the decision event is built from (i.e. it is a dynamic structure that relies on the

information offered by a formal decision event); besides, it can be expressed in OWL and has been implemented as an ontology as way to make it shareable and transportable [Sanin & Szczerbicki 2005b; 2006a; Sanin et al. 2007]. Four basic components surround decision-making events, and 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 taking into account some important features of DNA. Firstly, the combination of the four nucleotides of DNA gives uniqueness to itself, just as the combination of the four components of the SOE offer 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 & 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 Sets of Experience should be acquired and constructed. The day-to-day operation provides many decisions, and the result of this is a collection of many different Sets of Experience. A group of SOE 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 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 Figure 1).

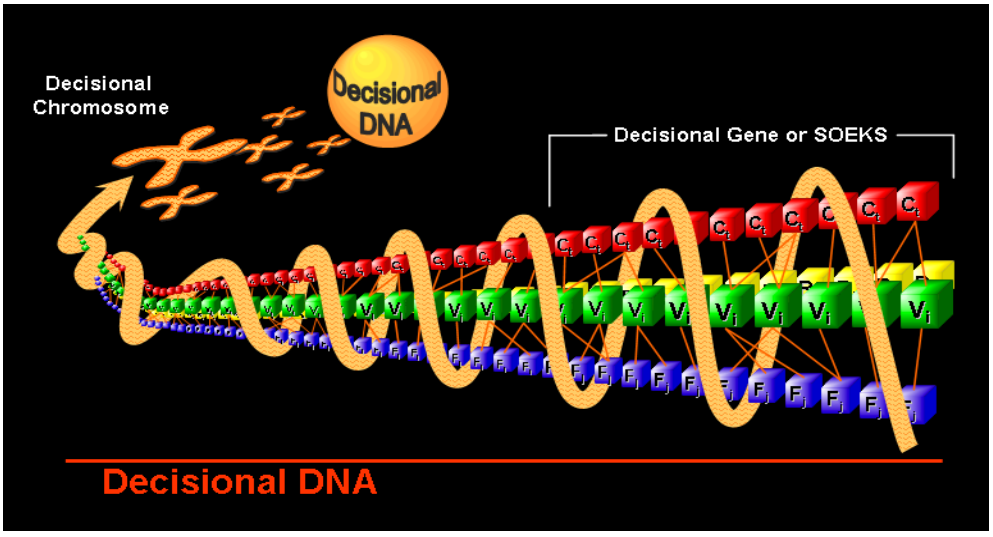

Figure 1. SOEKS and Decisional DNA[1]

In conclusion, the SOEKS is a compound of variables, functions, constraint and rules, which are uniquely combined to represent a formal decision event. Multiple Sets of experience can be collected, classified, and organized according to their efficiency, grouping them into decisional chromosomes. Chromosomes are groups of Sets of Experience 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 [Sanin and Szczerbicki 2005a; 2006b] should be reviewed.

3. REFLEXIVE ONTOLOGIES (RO)

Reflexivity 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 ontology is reflexive.

Thus, Toro et al. [2007b] 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 RO as abstract knowledge structure is essentially a set of structured contents and relationships, all the mathematical concepts of a set can be applied to it as a way of formalization and handling.

Properties of the Reflexive Ontologies

A RO is, basically, an ontology that has been extended with the concept of reflexivity. In order to be compliant with the RO concept, an extension of a base ontology must fulfil the next set of properties:

- *Property 1 – Query retrieval:* It refers to the faculty of storing every query performed in order to offer faster answers when re-queried. Such queries can refer to the structure of the ontology (data type) or to the instances (value type).
- *Property 2 – Integrity update:* It refers to the 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 individual.

[1] All figures: source own.

- *Property 3 – Autopoietic behaviour:* This property refers to the autopoietic capacity of self creation or auto-creation derived from the fact that for every new performed query improves the knowledge structure. The quality of the knowledge embedded in the system is increased as the system is in a constant self-nurture process that reflects the auto production capacity of any autopoietic system.
- *Property 4 – Support for logical operators:* This property provides any RO with mechanisms of set handling, from a logistic point of view; that is, it includes AND, NOT, and OR as logical operators.
- *Property 5– Self reasoning over the query set:* This property refers to the capacity of performing some logic operations over the query system in one of the following schemas:
  - *(i)* To discover patterns of queries,
  - *(ii)* To suggest the need of ontology refinement, as some elements are more queried than others; meaning that, probably, they should be taken into special consideration as the queries performed can cause a possible asymmetric behaviour, and
  - *(iii)* To discover new non-explicit relationships, meaning that some queries could be different, but their set of solutions are the same. Hence, this could imply a possible undercover relationship between sets of queries (possible serendipity behaviour).

Advantages of Implementing Reflexive Ontologies

The enhancement of having self contained queries relies in the following main aspects:

(i) *Speed on the query process:* As 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 has been performed previously. In case it has not been performed, then a typical ontology interrogation via an API takes place and when the new answer is retrieved, the query is added to the reflexivity class.

(ii) *Incremental nature:* the possibility of the ontology itself to add new queries on individuals with the correspondent answers to such queries: This is in fact a feature that adds knowledge about the domain as the more questions are asked, the more knowledge can be stored in the ontology. Questions and answers act as guidelines to infer "things" about the domain.

(iii) *Self containment of the Knowledge Structure in a single file:* This feature includes the storage of the model, the relations among the elements of the model, the individuals (instances) and queries over such individuals.

Implementation of the Reflexive Ontologies Concept

The following architecture has been defined in order to implement the RO concept.

This architecture is divided in three layers (see Figure 2) as follows: the repositories layer contains the real world elements that "feed" the ontology in the second layer, the base ontology layer. The reflexive layer, this is the extension itself, which contains two modules.

The first module adds a new class to the base ontology with the needed schema for the reflexivity; this is called the "ReflexiveOntologyQueryStorer class". Such extension hangs from the OWL:Thing super class and it has the OWL properties presented in Table 1. And the second module is the reflexiveness itself providing the ontology (programmatically) with a mechanism to perform queries and some logic on the queries that allows the handling of the reflexiveness.

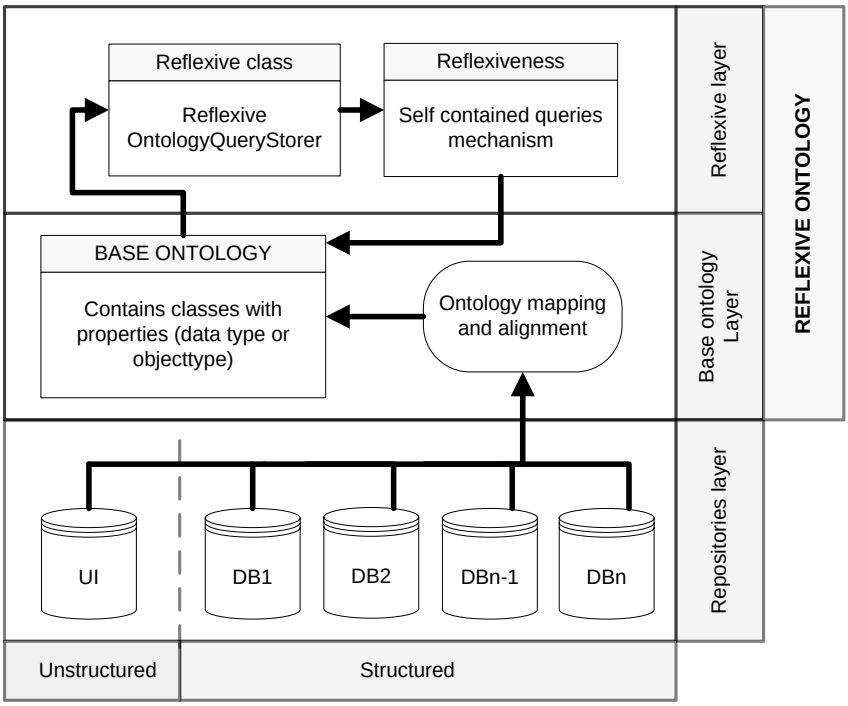

Figure 2. Reflexive Ontology architecture

This architecture is divided in three layers (see Figure 2) as follows: the repositories layer contains the real world elements that "feed" the ontology in the second layer, the base ontology layer. The reflexive layer, this is the extension itself, which contains two modules. The first module adds a new class to the base ontology with the needed schema for the reflexivity; this is called the "ReflexiveOntologyQueryStorer class". Such extension hangs from the OWL:Thing super class and it has the OWL properties presented in Table 1. And the second module is the reflexiveness itself providing the ontology (programmatically) with a mechanism to perform queries and some logic on the queries that allows the handling of the reflexiveness.

Table 1. Reflexive class properties

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

4. ENHANCING THE SOEKS WITH REFLEXIVENESS

Ontology-based technology was applied to the SOEKS by Sanin et al (2007) as a way to facilitate distributing and sharing companies' decisional DNA. Adding heavier semantics, logic, and expressiveness to the SOE resulted in an OWL decisional Ontology called the SOEKS-OWL. In this Section, such previously presented ontology is extended with the reflexive ontologies schema.

The initial step consisted on instantiating the SOEKS-OWL with real values. Figure 3 shows the visualization of the SOEKS-OWL instantiated (base ontology as presented in the architecture).

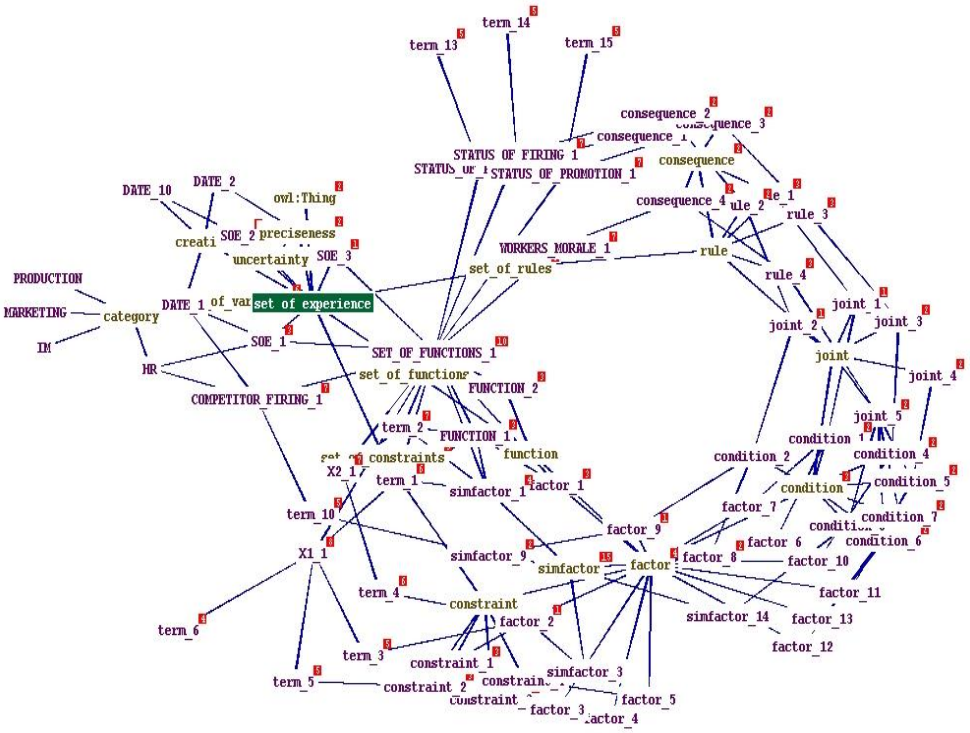

Figure 3. SOEKS-OWL instantiated

Next step comprises the adaptation of the Reflexive Query Storer class in order to make it operate with the SOEKS-OWL; that is, setting the class with some initial values such as the path of the used ontology, creation options of the reflexive structure and 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://testOntology//SOEKS_OWL.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;
```

Once this step is done, the querying process starts, and with the first executed query, the RO elements, as explained in the architecture, are created.

Any typical ontology has the knowledge definition represented as classes and properties, being the last ones of two possible types: *(i)* object type, mapping a class to a class and *(ii)* data type, mapping a class to a characteristic (a traditional fundamental data type such as string or integer); thus, for explanation purposes, three queries will be exemplified running over both kind of properties, object types and data types.

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 data type query. Such query is written in a human readable form, which 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 first time) as well as the query executed with results. Following, the results can be seen as a query successfully executed with the new instance in the SOEKS-OWL transformed into a Reflexive Ontology:

```
-----------------------------------START-------------------------------------
-
Testing Simple query : CLASS variable with the PROPERTY var_name EQUALS to X1
```

```
------------------------------------------------------------------------------
-
... saving successful.
File modification saved with 0 errors.
-------------------------------------END-------------------------------------
-
```

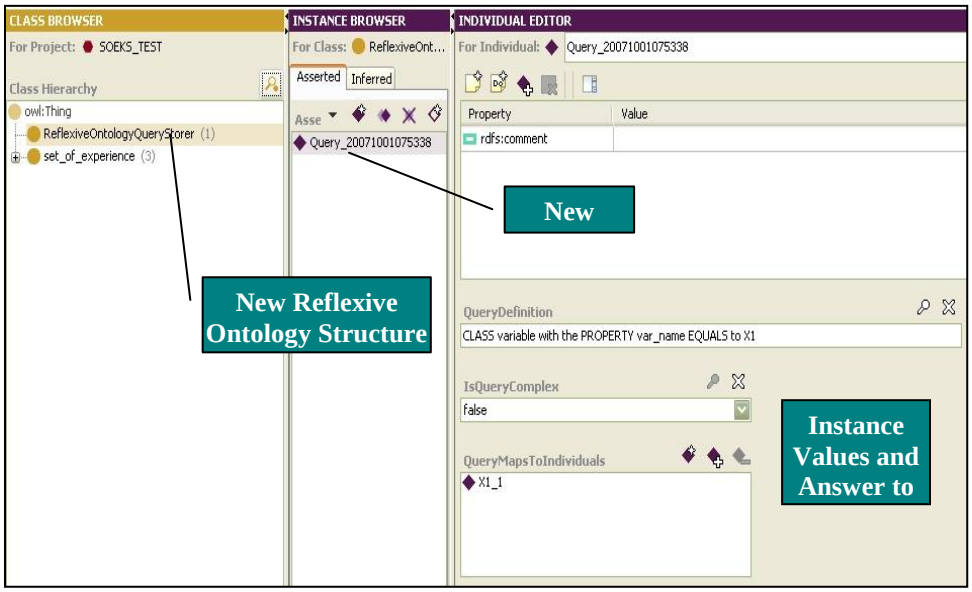

Figure 4. SOEKS-OWL transformed into a Reflexive Ontology with its new elements

The new class ReflexiveOntologyQueryStorer is created with one instance, such instance contains the three property values presented in table 2:

Table 2. Reflexive class with its property values

Property	Type	Value
isQueryComplex	Boolean	False
QueryDefinition	String	CLASS variable with the PROPERTY var_name EQUALS to X1String
QueryMapsToIndividuals	Object	X1_1

The next example is an object 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 the name X2_1". Its results are as follows (Figure 5):

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

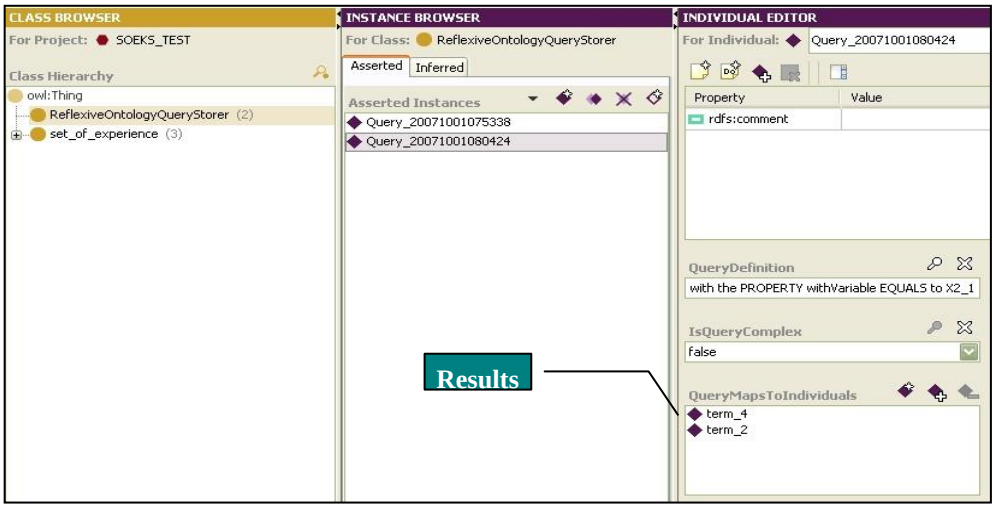

Figure 5. SOEKS-OWL with a data type query executed

In these results can be seen that 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 Figure 6.

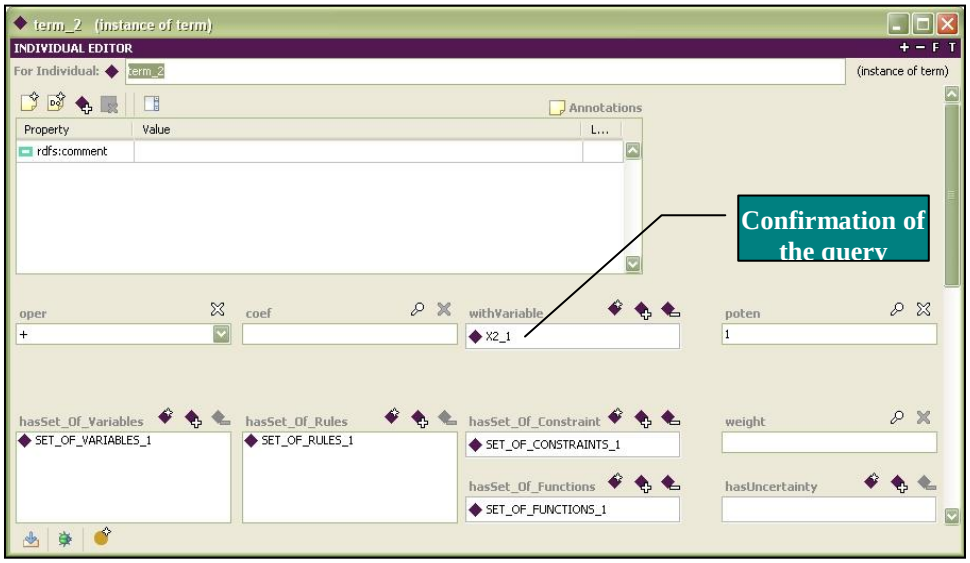

Figure 6. Term with confirming the query results

One of the features of the RO is that when a new query comes, if it was already executed, there is no need to perform it again. In such a case, the RO class will return an answer as follows:

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

The queries are stored according to the architecture presented in the query storer class, and repeated queries can be retrieved without osculating the ontology again. Finally, a complex query is performed:

```
public static String SIMPLE_RFLEXIVE_QUERY="CLASS
simfactor with the PROPERTY hasSterm EQUALS to term_1 AND
CLASS simfactor with the PROPERTY hasSterm EQUALS to
term_2";
```

This query mixes two object type queries showing properties 3 (autopoietic behaviour) and 4 (support for logical operations) (see Figure 7):

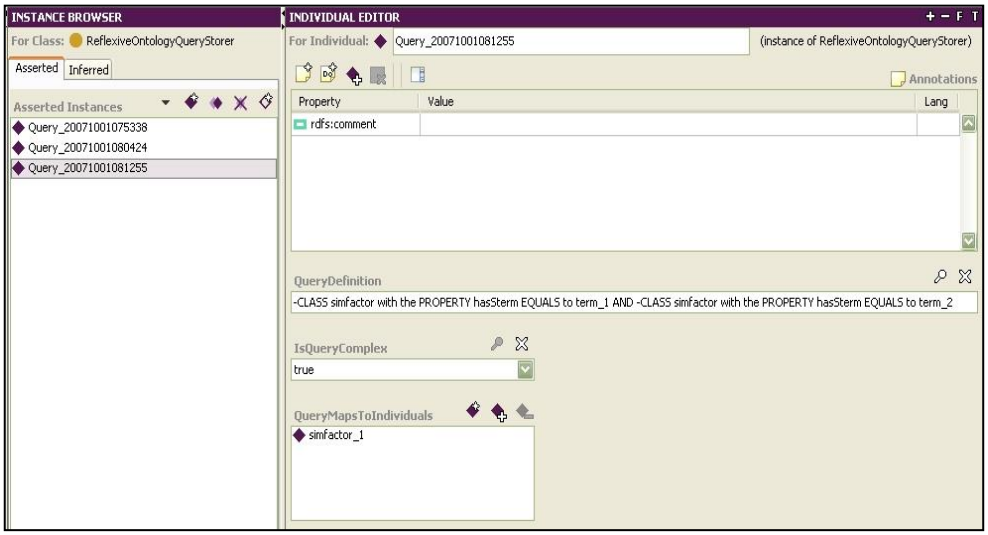

Figure 7. Complex query executed on the SOEKS-OWL

```
---------------------------------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
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-------------------------------------------
----------------
```

As it has been 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 object type) are executed, they are inserted as instances in the new Reflexive Ontology; this change facilitates the application of similarity elements among the Sets of Experience and it will allow an extended logic handling over the SOEKS as it comprises the 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.

Finally, we would like to mention that in our implementation we used protégé and its API's and the OWL-DL subspecies of the OWL specification [Zhang 2005], but any other ontology modelling software that offers an open API could be used to extend an ontology programmatically with the RO concept.

5. CONCLUSIONS

In this paper, the concept of Reflexive Ontologies (RO) was applied onto a knowledge Structure, the Set of Experience Knowledge Structure. The presented schema can be applied to an existing ontology in order to improve its query capabilities. The RO properties and benefits of having self contained queries were obtained as the SOEKS-OWL was extended.

Among the future work, we can mention that some additional features must be developed in order to offer an user friendly environment such as the implementation of a Graphical User Interface (GUI) by the means of an API which should help in the transformation of any ontology into a RO, implementation of a GUI for the query engine, and extension of the query 'logic' elements into a more human like language.

Moreover, a formalization of the concept 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 behaviour.

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 in the SOEKS paradigm.

Acknowledgements

This research was developed with in collaboration with the VicomTech Institute (partially financed by the Basque Government under the INTEK 2006-2008 call - Bi2Hiru).

REFERENCES

[1] Arnold W. and Bowie J.: *Artificial Intelligence*, A Personal Commonsense Journey, Prentice Hall, New Jersey, 1985.

[2] Berners-Lee T., Hendler J. and Lassila O.: *The Semantic Web - A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*, Scientific American, Vol. 5(284) May, pp. 28-31, 2001.

[3] Flanagan O.J.: *Psychology, progress, and the problem of reflexivity: a study in the epistemological foundations of psychology*, Journal of the History of the Behavioral Sciences, Vol. 17, pp. 375-386, 1981.

[4] Gruber T.R.: *Toward Principles for the Design of Ontologies Used for Knowledge Sharing*, International Journal of Human-Computer Studies, Vol. 43(5-6), pp. 907-928, 1995.

[5] Luhmann N.R.: *Organización y Decisión, Autopoiesis y Entendimiento Comunicativo*, Anthropos, Barcelona, 1997.
[6] Maturana H. and Varela F.: *Autopoiesis and Cognition*, Reidel, Boston, 1980.
[7] Reflexivity, 'Wikipedia, The Free Encyclopedia', 6 September 2007, http://en.wikipedia.org/w/index.php?title=Reflexivity_%28social_theory%29&oldid=150538764, 2007.
[8] Sanin C. and Szczerbicki E.: *Set of Experience: A Knowledge Structure for Formal Decision Events*, Foundations of Control and Management Sciences Journal, Poznan University of Technology, Vol. 3, pp. 95-113, 2005a.
[9] Sanin C. and Szczerbicki E.: *Using XML for Implementing Set of Experience Knowledge Structure*, Proceedings of 9th International Conference on Knowledge-based and Intelligent Information and Engineering Systems, KES, Melbourne, Part I LNAI 3681, Springer, pp. 946-952, 2005b.
[10] Sanin C. and Szczerbicki E.: *Extending Set of Experience Knowledge Structure into a Transportable Language XML (eXtensible Markup Language)*, Cybernetics and Systems: An International Journal, Taylor and Francis, Vol. 37, No. 2-3, pp. 97-117, 2006a.
[11] Sanin C. and Szczerbicki E.: *Using Set of Experience in the Process of Transforming Information into Knowledge*, International Journal of Enterprise Information Systems, Idea Group Publishing, Vol. 2, pp. 45-62, 2006b.
[12] Sanin C., Toro C. and Szczerbicki E.: *An OWL ontology of set of experience knowledge structure*, Journal of Universal Computer Science, Vol. 13(2), pp. 209-223, 2007.
[13] Sevilmis N., Stork A., Smithers T.: *Knowledge Sharing by Information Retrieval in the Semantic Web*, Proceedings of 2nd European Semantic Web Conference, ESWC, Greece, LNAI 3532, Springer, pp. 471-485, 2005.
[14] Toro C., Sanín C., Szczerbicki E. and Posada J.: *Reflexive Ontologies: Enhancing Ontologies with Self- Contained Queries*, International Journal of Cybernetics and Systems, 2007b (in print).
[15] Toro C., Sanín C., Vaquero J., Posada J. and Szczerbicki E.: *Knowledge Based Industrial Maintenance Using Portable Devices and Augmented Reality*, Proceedings of 11th International Conference on Knowledge-based and Intelligent Information and Engineering Systems, KES, Italy, Part I LNAI 4692, Springer, pp. 295-302, 2007c.
[16] Zhang Z.: *Ontology Query Languages for the semantic Web*, Athens, The University of Georgia, Master of Science, p. 57, 2005.