Semantic Capabilities for the Metrics and Indicators Cataloging Web System

Hernán Molina, Fernanda Papa, M. de los Angeles Martín, Luis Olsina
GIDIS, Department of Informatics, Engineering School at UNLPam
Calle 9 y 110, (6360) General Pico, La Pampa. Argentina
{olsinal, martinma, pmfer, hmolina}@ing.unlpam.edu.ar

Abstract. In this paper we thoroughly discuss design and implementation issues for semantic search and navigation to ontology-based metrics and indicators metadata. These semantic search and navigation capabilities are basic functionalities offered by the metrics and indicators cataloging web system in order to accede to related information by registered users. Moreover, this environment can finally allow tools, evaluators and other stakeholders to have service and consultation mechanisms based on a sound specification of the entity type, the attribute and metric definition, related indicators, and calculable concepts, among other metadata. In fact, distributed cataloging systems and interoperable repositories of metrics and indicators can effectively be used to support different quality assurance processes. Particularly, in this work we show the design of the navigational model for semantic search and navigation by the registered user of the cataloging system. The navigational map is a view defined over the metrics and indicators conceptual model [8]. In addition, aspects of the formalization in RDF/S specification and the implementation in the Sesame architecture are shown. Effective and full-fledged quality assurance processes require not only methodological but also up-to-date technological support.

Keywords: Metrics and Indicators, Semantic Web, OOWS, Web Cataloging Tool.

1 Introduction

As we know, quality assurance is one of the challenging processes both to the Software Engineering (SE) and to the Web Engineering (WE) –as a new discipline [4]. A common building block shared for many assessment methods and techniques that give support to the quality assurance process is the specification of nonfunctional requirements stemmed from a sound definition and documentation of attributes and calculable concepts (e.g., quality, quality in use, productivity) and their metrics and indicators that quantify and evaluate them.

In fact, great amounts of information about attributes, metrics, and indicators for different purposes and domains have been published in diverse fora and media. However, observing the rapid and chaotic growth and heterogeneity of information sources related to metrics and indicators, in addition to some lack of consensus in the terminology, it urges to provide mechanisms for agreeing, structuring, and retrieving that information in order to be useful to diverse software and Web assurance activities, methods, and tools. For this end, we are building a cataloging web system that basically provides different stakeholders with consultation, retrieval, and reuse mechanisms starting from a sound specification of the diverse metadata of software metrics and indicators [8, 10].

Some efforts have been made to classify metrics for some entity type as for example metrics for software products. It is worthwhile remarking the initiative of the ISO 9126 standard [5] in the 9126-2 and 3 draft documents. However, the paper-based template of information used to describe those metrics is not sufficiently complete (as was discussed in [8]), and it is not intended being automated by retrieval tools. Maybe the closer works to our initiative are the ZD-MIS (Zuse/Drabe Measure Information System) CD-ROM delivered with the Zuse's book [13]. But we argue that our system can be more robust in delivering the information, in agreeing metrics and indicators, and in the completeness of metadata used to model metrics and indicators. It is also worthy of mention the initiative carried out by Kitchenham *et al.* [7] in the definition of a conceptual framework and infrastructure (based on the Entity-Relationship model) to specify entities, attributes and relationships for measuring and instantiating software projects, with the purpose of analyzing metrics and datasets in a consistent way. As result of this project, the MiniSQUID prototype [7] has been reported as a useful tool to support metadata and data set maintenance -as we know, this system was intended just for storing metrics data and metadata but not for indicators.

This last framework has served as a starting point for our proposal [10], which we have strengthened not only from the conceptual modeling point of view (using O-O approaches), but also from the cataloging technologies and offered services (using Semantic Web approaches).

In this paper we discuss design and implementation issues for semantic search and navigation to ontology-based metrics and indicators metadata. Particularly, in this work we show the design of the navigational model by using the OOWS (*Object-Oriented Web Solutions* [11]) approach. The navigational map is a view defined over the metrics and indicators conceptual model [8]. In addition, for semantic search and navigation by the registered user of the cataloging system, aspects of the formalization in RDF/S specification and the implementation in the Sesame architecture are shown as well. The semantic web approach, bridge the gap between the enormous amount of semi-structured and heterogeneous data available in the Web and the ability to cope with all this with machine-processable semantics.

The rest of this article proceeds as follows. In Section 2, we give an overview of the cataloging web system's architecture stressing the semantic web components. In Section 3, we specify the conceptual model for the metrics and indicators domain, which is the cornerstone to develop the cataloging web system with semantic web power. In Sections 4 and 5, the semantic search and navigation capabilities from both the design and implementation aspects are discussed. Finally, concluding remarks and future works are drawn.

2 Metrics and Indicators Cataloging Web System

2.1 Overview of the Cataloging Web System's Architecture.

The metrics and indicators cataloging system will provide a Web-based collaborative mechanism for discussing, agreeing, and adding approved metrics and indicators to the repository on one hand, and semantic search and navigation functionalities for consultation and reuse, on the other hand. These subsystems are outlined in Fig. 1.

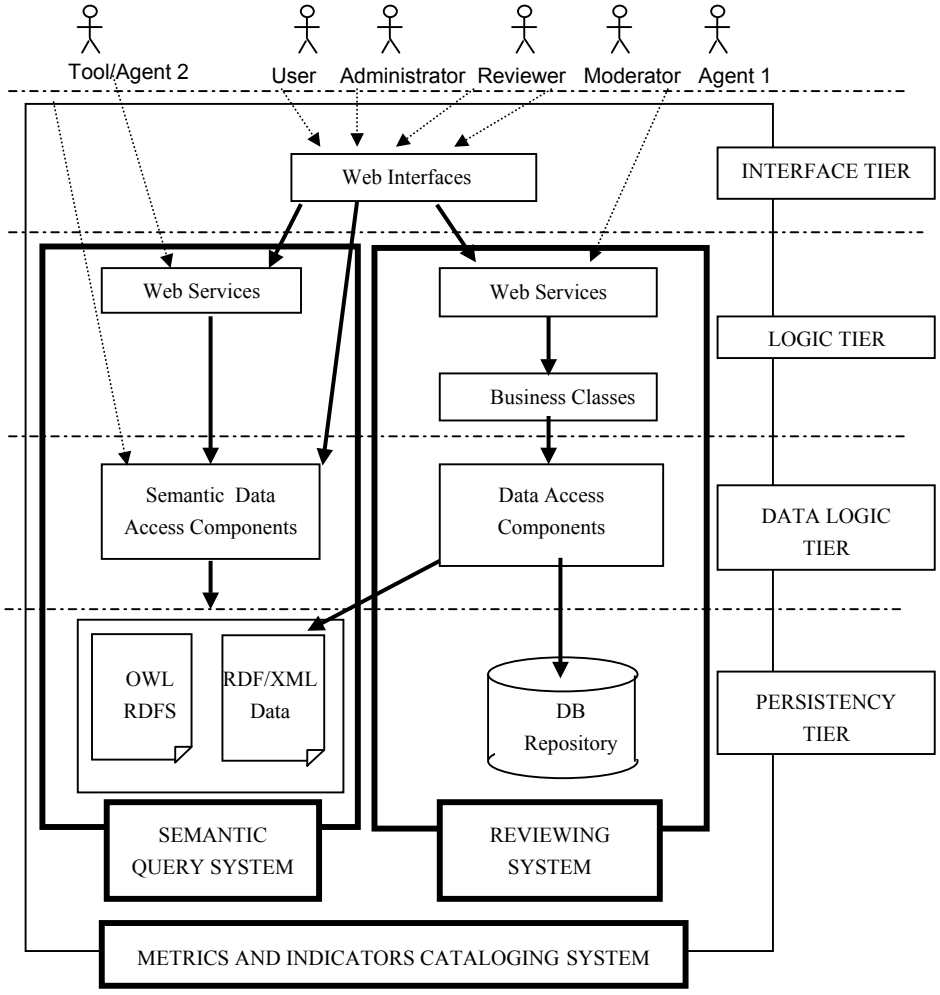

Fig.1 An Architectural view of the Metrics and Indicators Cataloging Web System.

From the design of users' point of view, five user's role types with different responsibilities and access privileges were considered, namely: *Administrators, Moderators*, *Reviewers*, *Registered Users* and *Tools/Agents*. The user's role types were discussed in [10] and previous works. The *Final User* can be a human being that accesses the catalog of metrics and indicators by means of searching and browsing functionalities with read-only access permissions. Applications will also be able to access the repository in a similar way, i.e., by means of Web services and the SOAP (*Simple Object Access Protocol*) interface.

To design the cataloging architecture, we have chosen a *multi-level* architectural style or, the so-called *n-tier* architecture. Basically, the system is composed of two subsystems, namely: the *Metrics and Indicators Reviewing System*, and the *Metrics and Indicators Semantic Query System*. The former subsystem is the responsible for the management and manipulation of the metrics and indicators in the catalog. It provides the functionality to users in order to perform the metrics reviewing process through the web, and to extract data about metrics and indicators. The latter subsystem is the responsible for the publication of cataloged metrics and indicators making use of the semantic web principles. It does not implement any management functionality; however, it must be capable of searching and navigating on-line semantic documents and repositories. It is made up of three core tiers:

1. *Logic tier*. It implements the querying, searching, and browsing capabilities via web services. It can be used by the interface layer and by other software applications (represented by *Tools/Agents*).
2. *Data Access tier*. It contains a set of components based on the Sesame architecture [3] to accede, for example, to different on-line repositories and documents.
3. *Persistency tier*. A set of web pages and documents with semantic information about metrics and indicators specified in OWL (*Ontology Web Language*) [1], RDFS (*Resource Description Framework Schema*) [12], and RDF/XML).

To design the *Metrics and Indicators Semantic Query System* we wanted to use *semantic web* principles in order to facilitate reaching the system functionality through the web, with information processing capability. Thus, to fulfill this objective we based our system in a very known architecture for storing and querying RDF data and schema information: *The SESAME architecture*. In the next subsection we introduce the Sesame architecture, and how this was adapted for browsing and searching the cataloging system.

2.2 The Adopted Sesame Architecture.

Sesame is a Web-based architecture, which allows persistent storage of RDF data and schema information and subsequent on-line querying of that information [3]. An adaptation of this architecture to our system is illustrated in Fig. 2.

For persistent storage of RDF data and schema, Sesame needs a scalable repository. Because the Sesame's authors wanted to keep Sesame DBMS-independent, all DBMS-specific code was concentrated in a single architectural layer of Sesame: the *Repository Abstraction Layer* (RAL).

This RAL offers RDF-specific methods to its clients and translates these methods to calls to its specific DBMS. An important advantage of the introduction of such separate layer is that it makes it possible to implement Sesame on top of a wide variety of repositories without changing any of Sesame's other components [3]. Sesame's core functional modules are clients of the RAL. Currently, there are three such modules:

- *The RQL query module.* This module evaluates RQL (*RDF Query Language*) [6] queries -see section 5.1 for a pair of examples of RQL queries.
- *The RDF administration module.* This module allows incremental uploading of RDF data and schema information, as well as the deleting of information.
- *The RDF export module.* This module allows the extraction of the complete schema and/or data from a model in RDF format.

As the authors indicate in [3], depending on the environment in which it is deployed, different ways to communicate with the Sesame modules may be desirable. For example, communication over HTTP may be preferable in a Web context, but in other contexts protocols such as RMI (*Remote Method Invocation*) or SOAP (*Simple Object Access Protocol*) may be more suited. As a matter of fact, the Sesame environment is an open software that provides storage and querying functions modules for RDF data and schemas, allowing additional protocol handlers if necessary. We viewed the Sesame's architecture as an appealing possible solution in terms of reusing existing APIs and tools that also provided flexibility and interoperability.

The Semantic Query System module of our cataloguing web system (see Fig. 1) must be capable of querying on-line semantic documents containing the metrics and indicators information. The architecture shown in Fig. 2 allows transparent manipulation of the repository for Agents and Tools. Each application can either work with the repository through the existing Sesame modules or by calling web services for more specialized functions of searching and querying. On the other hand, the repository contains a compact body of knowledge related to

metrics and indicators that could be used, manipulated, and referred as a whole. Such repository contains both ontological assertions and instance data. The querying of ontology's schema and instances are handled by the Sesame system by using the RQL language. One of the main RQL features that distinguish it from some other RDF query languages is its capability of querying both RDF schemas and data. Currently, we are using the SeRQL (Sesame RQL) interface, because it supports all the RQL language capabilities but is fitted to the implementation features of the Sesame architecture.

If more expressive reasoning is necessary (e.g. by means of the OWL language [1]) then the corresponding information should be sent to an external reasoner that processes the query and returns the answer back.

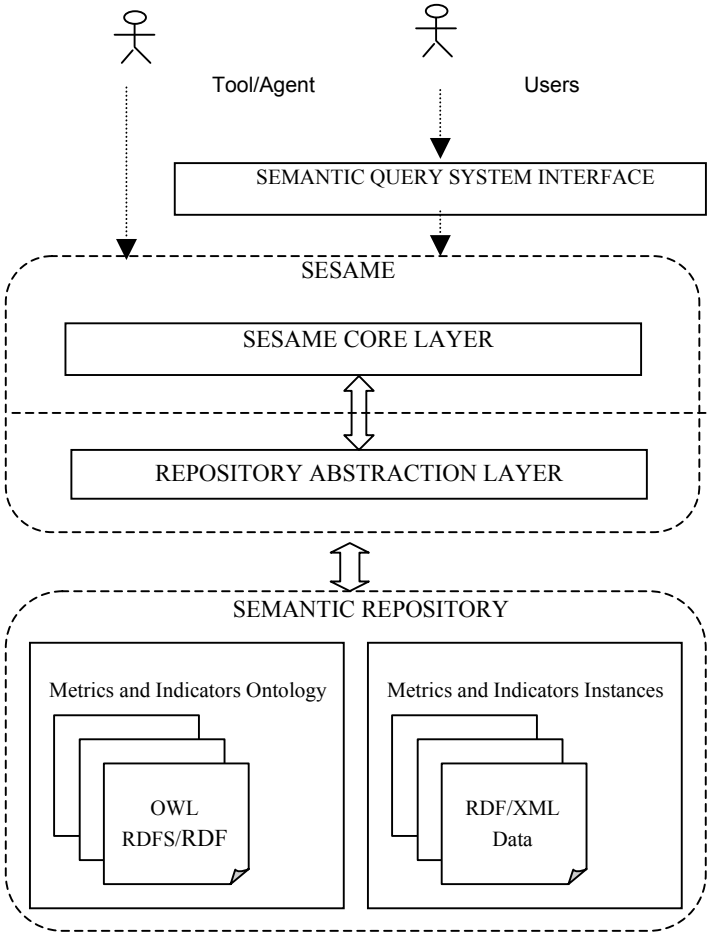

Fig. 2 Metrics and Indicators Semantic Query System architecture.

3 Conceptual Model for the Metrics and Indicators Domain

In this section we analyze the conceptual model for metrics and indicators, which is the cornerstone for designing and implementing the cataloging web systems with semantic web power. The conceptual model specifies the main classes, attributes and relationships for the metrics and indicators domain useful to define and exploit all the information for cataloging purposes.

Fig. 3 shows the UML-based conceptual model for this domain (notice this model is called Object Model in OOWS [11]). Classes such as *Entity*, *Attribute*, *Calculable Concept*, *Metric*, *Scale, Method*, *Measure*, *Unit*, *Indicator*, among others, and their relationships can be observed. A thorough discussion from the ontological standpoint considering the meaning of concepts, properties, and relationships was made in [8]. For the shake of brevity some descriptions of them are given next.

From the measurement process viewpoint, in the empirical domain we have mainly *Entity*, *Attribute* and *Calculable Concept* classes (see Fig. 3). The selection and/or definition of appropriate attributes and indicators to address an *information need* starts with the specification of a calculable concept to be evaluated or estimated. A

calculable concept is an abstract relationship between attributes of entities and information needs. Examples of calculable concepts include reliability, quality in use, productivity, etc.

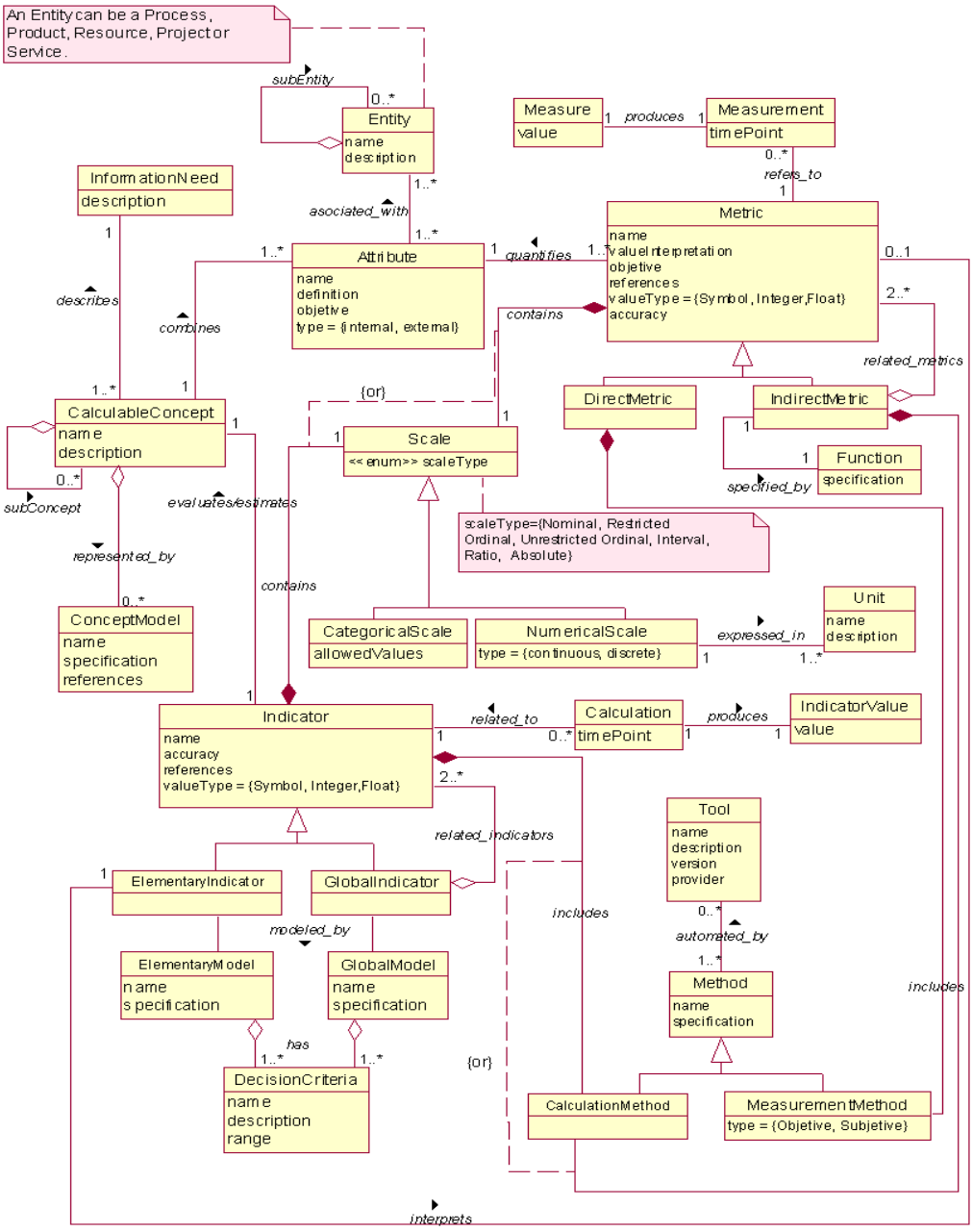

Fig. 3 Conceptual model for the domain of metrics and indicators.

Considering the level of abstraction a calculable concept can be composed of other subconcepts, which could be represented by a *concept model* (e.g. ISO 9126-1 [5] specifies a quality model based on characteristics and subcharacteristics). A calculable concept is associated to one or more attributes of entities.

The *Entity* class represents tangible or intangible objects we observe in the empirical (real) world, which is characterised by measuring its attributes. Types of entities of interest to software and web engineering are: Project, Product, Service, Process, and Resource. In addition, the *Attribute* class represents a measurable physical or abstract property of an entity. For a given attribute, there is always at least an empirical relationship of interest that can be captured and represented in the formal domain (by means of a *Metric*), enabling us to explore the relationship mathematically. Attribute is an object at the lowest level of abstraction that can be quantified directly or indirectly by a metric; meanwhile a *Calculable Concept* class represents objects at the highest level of abstraction.

The *Metric* class represents the specific mapping of an empirical attribute (that belongs to an entity) to the formal world. For instance, the metric m represents the mapping m: A -> X, where A is an empirical attribute of one or more entities (the empirical world), X the variable to which categorical or numerical values can be assigned (the formal world), and the arrow denotes a mapping. In order to perform this mapping a sound and precise measurement (activity) definition is needed by specifying explicitly the *method* and *scale*. On the other hand, a semantic distinction between metric and indicator concepts should be raised. The indicator represents a new mapping coming from the interpretation of the metric's value (formal world) into the new variable to which categorical or numerical values can be assigned (the new formal world). In order to do this mapping a *model* and *decision criteria* for a specific user information need is considered.

Ultimately, all the information observed in the conceptual model is necessary in order to have a sound specification of the domain for cataloging and consultation purposes (see [8] for a complete definition of terms and examples. In addition this conceptual model was formalized in RDF and RDFS languages).

4 Semantic Navigation Capabilities for the Cataloging System

We argue that the conceptual modeling process is necessary in order to develop correct web applications, as well as to support the documentation of complex ones. Moreover, we consider navigational modeling as a critical feature as well. In this section we analyze the navigational model introduced by the OOWS [11] approach to specify navigational requirements of a web application at conceptual level. This approach uses UML-compliant diagrams to represent its abstraction primitives. The next subsections present this navigational model applied to the definition of the navigational requirements of the metrics and indicators cataloging system for the *Registered User* in addition to the use of the Sesame server API to browse semantic information from the target system.

4.1 Navigational Model for the Registered User.

OOWS is a conceptual modeling method that uses a *Navigational Model* to properly capture the navigation semantics of a web application by providing a set of conceptual constructors to define navigation capabilities (structure, navigability, etc.) of a web system. The navigational model is made up of a set of navigational maps, one for each type of user of the system.

A *Navigational Map* (see Fig. 4) represents the personalized view that each kind of user has over the system and defines how users would access the system information and functionality. It is represented by means of a directed graph where nodes represent user interaction units (navigational contexts), and arcs (navigational links) represent node reachability (defining valid navigational paths). Thus, a *navigational context* is a node in this

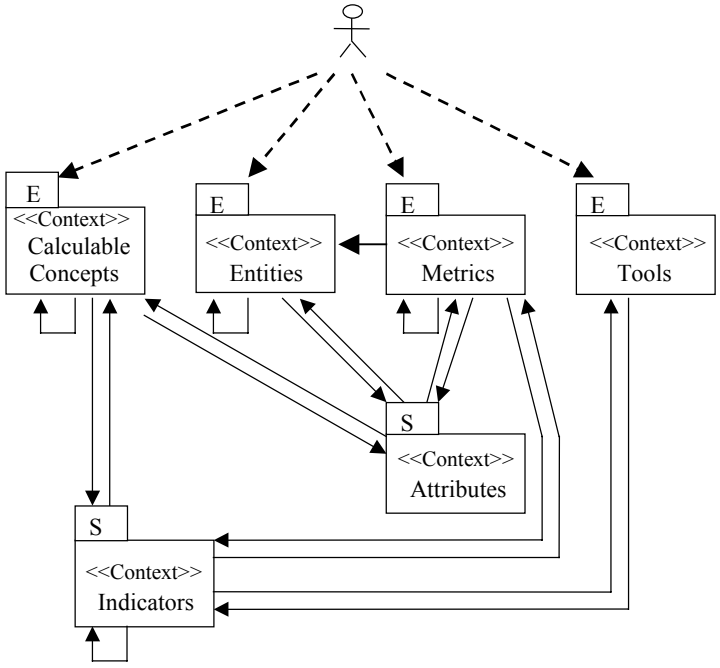

Fig. 4. Navigational Map for the *Registered User* of the Cataloging System

graph (depicted as UML packages stereotyped with the «context» reserved word). It represents a view over a set of class attributes and services, and also class relationships, from the object model (shown in Fig. 3).

There exist contexts of two types: a) *exploration contexts* (labeled with an "E"), which represent navigational nodes that can be reached at any moment independently of the current context (e.g. landmarks); and b) *sequence contexts* (labeled with an "S") that can only be reached following a predefined navigational path (sequence of navigational contexts).

Fig. 4 presents the navigational map for the Registered User of the metrics and indicators cataloging web system. It defines six navigational contexts: Calculable Concepts, Entities, Metrics and Tools -that are exploration contexts which have a dashed arrow starting from the user, and Indicators and Attributes -that are sequence contexts, only reachable following predefined navigational paths. For instance, Indicators context could not be reachable from Entities context. Each one of these contexts defines a view over the object model providing access to the relevant information (class attributes), and functionality (class services) for the Registered User.

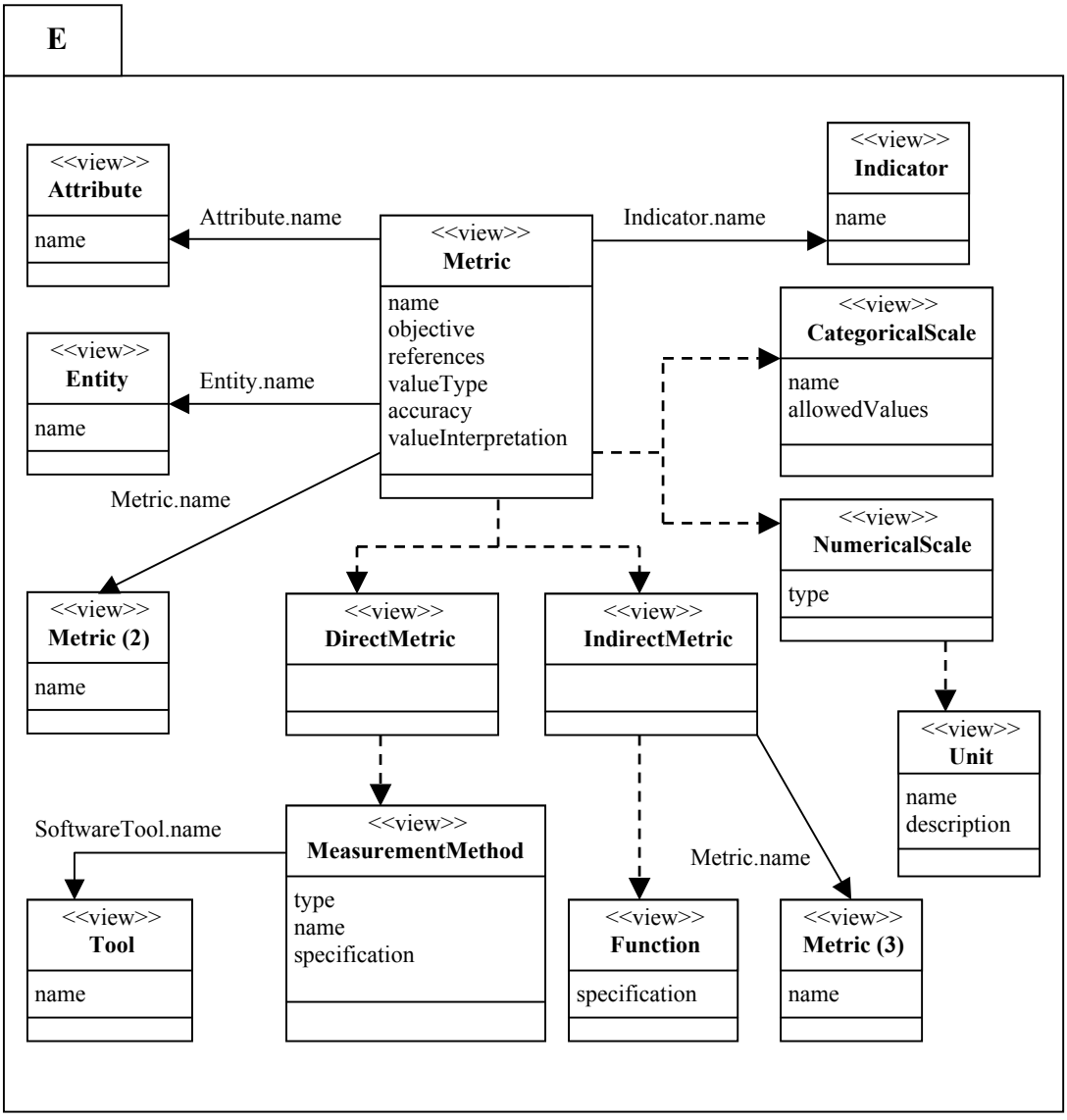

Fig. 5. Metrics context for the *Registered User*

The following step is to define with detail each specified context. A navigational context is made up of a set of *navigational classes* and *navigational relationships*. A *navigational class* represents a view of a class in the object model, defining a subset of visible attributes and services. They are graphically depicted as UML classes stereotyped with the «view» reserved word (see Fig. 5). A navigational class represents a graphical query to

retrieve all the instances of the related class, specifying the attributes to retrieve. A *population filter* can be specified related to a navigational class defining a filtering condition for the instances to be retrieved.

Figure 5 shows the definition of the Metrics context. This context is responsible of provide relevant information about metrics for the Registered User. Metrics context has fourteen navigational classes (e.g. Metric, Attribute, Entity, among others) with its attributes (name, objective, references, etc).

All navigational classes must be related by a unidirectional binary relationship, called *navigational relationship* (depicted as an arrow). It is defined over an existing aggregation, association, composition or specialization relationship in the Object Model and specifies a retrieval of all the instances of the target navigational class related to the origin class. For instance, all the related Attributes (name), Entities (name), Indicators (name), etc… for each retrieved Metric will be shown.

There exist navigational relationships of two types: 1) *context dependency relationships* (depicted as dashed arrows), and 2) *context relationships* (depicted as solid arrows). The former only defines a population retrieval. The latter also defines a navigation to a *[target context]* using as the *anchor* an attribute of the target class. For instance, when retrieving all the Entities for a given Metric, it will be possible to navigate to the target context [Entities] by selecting one of them using the name of that Entity as the anchor (defining the navigational link between the contexts Metrics → Entities specified in the navigational map – see Fig. 4).

Finally, the other contexts for the Registered User were specified in the same way.

4.2 Semantic Browsing Implementation for the Registered User.

As previously commented, RDF and RDFS languages were used in order to implement the ontological information of metrics and indicators (represented in the conceptual model of Fig. 3). To gain modeling power for semantic navigation some primitives of the OOWS navigational model were formalized. For instance, the elements of a navigational map for a given user were implemented in RDF. For this end, three RDF schema resources were defined, namely: *View*, *ExplorationContext*, and *Navigational Link*. In the next excerpt of code, the specification of these resources is shown.

```
<rdfs:Class rdf:ID="View">
        <rdfs:label xml:lang="en">View</rdfs:label>
        <rdfs:comment>Entry point for a navigational map</rdfs:comment>
</rdfs:Class>
<rdfs:Class rdf:ID="ExplorationContext">
        <rdfs:label xml:lang="en">Exploration Context</rdfs:label>
        <rdfs:comment>Destination to an Entry point</rdfs:comment>
</rdfs:Class>
<rdf:Property rdf:ID="NavigationalLink">
        <rdfs:label xml:lang="en">Navigational Link</rdfs:label>
        <rdfs:comment>Navigational Link from a given user View to an Exploration
                Context</rdfs:comment>
        <rdfs:domain rdf:resource="#View"/>
        <rdfs:range rdf:resource="#ExplorationContext"/>
</rdf:Property>
```

In the following excerpt of code the instances of the above resources are specified for the *Metrics* navigational context.

```
<rdf:RDF ... xmlns:metnav="http://gidiss/rdf/m_nav_v1.rdf#">
        ...
        <rdf:Description rdf:about="http://gidiss/rdf/m_i_v4.rdf#Metrics">
                <rdf:type rdf:resource="http://gidiss/rdf/m_nav_v1.rdf#ExplorationContext"/>
        </rdf:Description>
        <rdf:Description rdf:ID="RegisteredUserView">
                <rdf:type rdf:resource="http://gidiss/rdf/m_nav_v1.rdf#View"/>
        </rdf:Description>
        <rdf:Description rdf:about="#RegisteredUserView">
                <metnav:NavigationalLink rdf:resource="http://gidiss/rdf/m_i_v4.rdf#Metrics"/>
                ...
        </rdf:Description>
...
<rdf:RDF>
```

On the other hand, we used the Sesame Server's API from a JSP page in order to get the different items of exploration contexts. The used method is getStatements of the RdfSchemaSource class, and the method call has the following form:

```
StatementIterator iter;
iter=rdfSource.getStatements(
     new Resource("http://gidiss/rdf/datosNav.rdf#RegisteredUserView"),
     new Resource("http://gidiss/rdf/m_nav_v1.rdf#NavigationalLink"),
     null);
```

In addition, to implement the navigational paths considered in navigational contexts the defined properties of RDF schema for the metrics and indicators conceptual model were used for context relationships (see Section 4.1). For context dependency relationships a new property was specified. In the following excerpt of code the RDF schema for this latter relationship is shown.

```
<rdfs:Property rdf:ID="ContextDependencyRelationship">
       <rdfs:label xml:lang="en">Context Dependency Relationship</rdfs:label>
       <rdfs:comment>It allows to show the information in the same view</rdfs:comment>
</rdfs:Property>
```

An example of instance of the above relationship is the *indicatorIncludes* as shown bellow:

```
<rdf:Description rdf:about="http://gidiss/rdf/m_i_v4.rdf#indicatorIncludes">
       <rdf:type rdf:resource="http://gidiss/rdf/m_nav_v1.rdf#ContextDependencyRelationship"/>
</rdf:Description>
```

To implement in a JSP page this relationship, each property is checked for membership to the ContextDependencyRelationship type. If it checks, the appropriated data is retrieved and shown.

5 Semantic Search Capabilities for the Cataloging System

In this section we briefly address the power of the semantic web approach focusing us particularly on queries for the metrics and indicators cataloging system. From a general point of view, it is well known the ability of the semantic web languages and technologies to cope with enormous amount of semi-structured and heterogeneous data available in different repositories and documents with machine-processable semantics.

5.1 A Semantic Query Language: RQL.

As mentioned in Section 2, taking into account the sate-of-the-art of semantic web technologies and languages, a semantic query language as RQL (*RDF Query Language*) was used in order to retrieve the semantic information from the cataloging system.

RQL is a typed declarative query language -developed in the ICS-FORTH (http://139.91.183.30:9090/RDF/) institute, which is based on the evaluation of path expressions on RDF graphs, featuring variables on labels for both nodes (i.e., classes) and edges (i.e., properties). One of the main RQL features that distinguishes from some other RDF query languages, it is its capability of querying both RDF schemas and descriptions (i.e., instances).

For the M&I Cataloging Web System this is a powerful feature for exploiting semantic documents and repositories of metrics, by allowing us not only to retrieve metrics and indicators data and its relations but also descriptive information (metadata) of available resources and services in the web, without human-processing intervention. Moreover, RQL has inference capabilities on hierarchies of classes and properties. This feature allows us exploiting additional information, even information that is not explicitly modeled in schemas.

In Table 1, we specify a pair of queries for the metrics and indicators cataloging system in order to illustrate some RQL features and powerness. In the first case, the RQL query acts only on RDF descriptions (instances) without the need of schemas. In the second case, the query retrieves all the properties information related to the Metric class, i.e. the labels of all edges.

Table 1. RQL query examples for the metrics and indicators domain.

Description	**Query Example**
1) Retrieves all instances of attributes and metrics for a subentity (named Webpage)	Select X, Y, Z from http://gidis.ing.unlpam.edu.ar/rdf/RDF-Metricas1#Entity{X}. http://gidis.ing.unlpam.edu.ar/rdf/RDF-Metricas1#Possess{Y}. http://gidis.ing.unlpam.edu.ar/rdf/RDF-Metricas1#IsQuantified{Z} Where X="Webpage"
2) Retrieves all properties names and their ranges to the Metric class	Select @P, range(@P) from {$C}@P Where $C ="http://gidis.ing.unlpam.edu.ar/rdf/RDF-Metricas1#Metric"

5.2 Semantic Searching Implementation for the Cataloging System.

Two types of semantic search were implemented: A quick search, available from every page of the site, and an advanced search, as shown in the screenshot of Fig. 6. In both cases, the string of the query is generated by a servlet and sent to the SeRQL query engine, which is accessed by means of the Sesame Server's API.

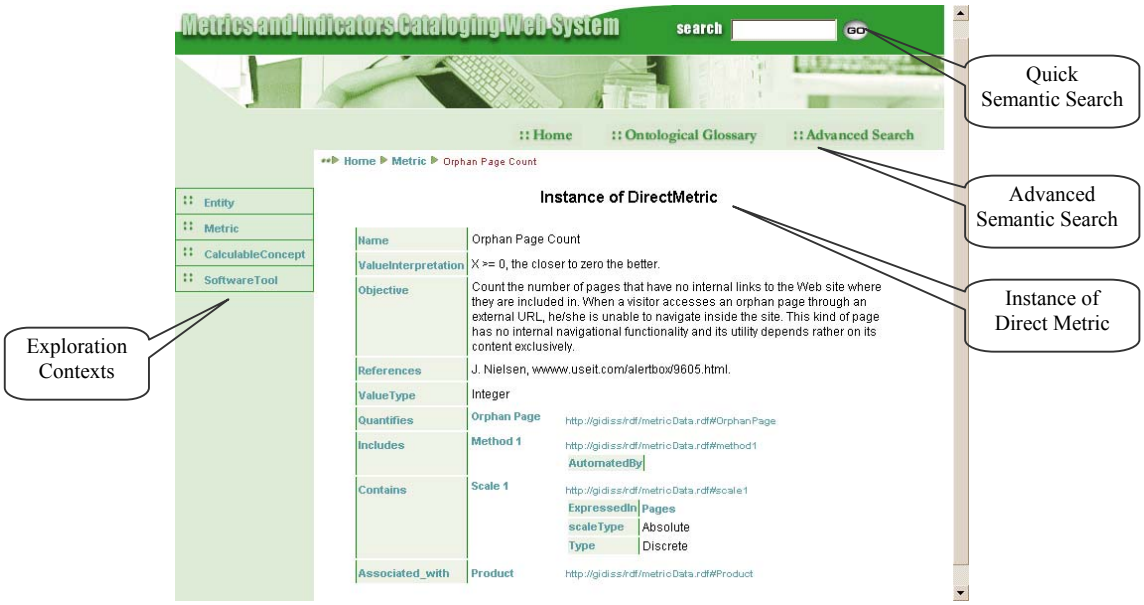

Fig. 6. Semantic browsing and searching capabilities of the M&I Cataloging System accessed by the register user.

For the quick search, a set of blank-separated words or optionally a string in quotation marks is accepted. These words are searched in the `rfds:label` and `rdfs:comment` properties. For instance, considering the `Enlace "Sitio Web"` string search:

```
SELECT Item, Name, Description
FROM {Item} <rdfs:label> {Name}; <rdfs:comment> {Description}
WHERE (Description like "*Enlace*" OR Name like "*Enlace*") AND
      (Description like "*Sitio Web*" OR Name like "*Sitio Web*")
```

The result is a list of instances and their descriptions. It is necessary that all RDF instances have the `rfds:label` and `rdfs:comment` properties defined. In this way the query does not disregard none instance.

For the advanced search, these key elements were considered:

- What to look for (i.e. the set of words to search)
- Where to look for (it can be the *name*, *description* or *all the properties* of instances)
- What to retrieve (e.g. any of the defined classes in the metrics and indicators schema)

For example, for the search: *all the Metrics and Attributes that contain the word "Enlace" both in the name or description*, the specification in RQL is as follows:

```
SELECT Item, Name, Description
FROM {Item} <rdf:type> {Class};
            <rdfs:label> {Name};
            <rdfs:comment> {Description}
WHERE (Name like "*Enlace*"
       OR Description like "*Enlace*") AND
      (Class = <metr:Metric> OR Class = <metr:Attribute>)
USING NAMESPACE
    metr = <!http://gidiss/rdf/m_i_v4.rdf#>
```

6 Concluding Remarks and Future Work

Sound metrics and indicators specification, flexible documentation, consultation, and retrieval mechanisms are needed in order to contribute to the comprehension and selection process whether metrics and indicators can be useful, easy to collect, and understand. Regarding the aim of our research, we argued that both a well-designed repository of metrics and indicators and a cataloging web system with semantic web power can be effectively used to support quality assurance processes such as nonfunctional requirement definition and specification, metrics and indicators understanding and selection, quality testing definition, amongst others. It is recognized that effective and full-fledged quality assurance processes require not only methodological but also up-to-date technological support.

Unfortunately, in recent research initiatives of the SE and WE communities, sound metrics and indicators specifications and cataloging environments as technological support for quality assurance processes have often been neglected. As a way to contribute to fill this gap, our recent and current research concern is manifold. Firstly, we have defined a sound conceptualization to the metrics and indicators domain formalized by the ontology [8]. This ontology has been the cornerstone for designing and implementing the cataloging web system. Secondly, we have implemented a prototypical application (the cataloging web system) with semantic search and navigation capabilities, as discussed in this paper. Thirdly, we are exploring the inclusion of OOWS-centered modeling primitives with the principles and models of the semantic web. This concern is in initial stages as result of current collaborations with the Oscar Pastor's research group (UPV, Spain). Finally, one line of research is the design and implementation of different tools to support quality assurance processes, being the cataloging system of metrics and indicators the core source for these tools. For instance, WebQEM_Tool must be able to use the metrics and indicators metadata to design and perform evaluations. WebQEM_Tool is the supporting tool for the WebQEM [9] methodology, which allows defining nonfunctional requirements, designing and performing elementary and global evaluation, analysis and recommendation tasks.

Acknowledgments. This research is supported partially by the UNLPam-09/F022 project.

References

1. Bechhofer S., van Harmelen F., Hendler J., Horrocks I., McGuinness D., Patel-Schneider P., and Stein L., OWL Web Ontology Language Reference, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/
2. Berners-Lee, T.; Hendler, J.; Lassila, O., 2001, *The Semantic Web.* Scientific American, http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html
3. Davies, J., Fensel, D. and Van Harmelen, F., "*Towards the Semantic Web: Ontology-driven Knowledge Management*", John Willey & Sons (2003).
4. Deshpande, Y.; Murugesan, S., Ginige, A., Hansen, S., Schwabe, D., Gaedke, M., White, B., "Web Engineering", *Journal of Web Engineering*, Rinton Press, US, 1(1), pp. 61-73 (2002).
5. ISO/IEC 9126-1 "International Standard, Software Engineering - Product Quality - Part 1: Quality Model" (2001).
6. Karvounarakis, G.; Alexaki, S.; et al, RQL: A Declarative Query Language for RDF. *In Procced. of 11th WWW12*, Honolulu, Hawaii, US (2002)
7. Kitchenham B.A., Hughes R.T., Linkman S.G., "Modeling Software Measurement Data", *IEEE Transactions on Software Engineering*, 27(9), pp. 788-804 (2001).
8. Martín, M.; Olsina, L., "Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System", *In proceed. of IEEE Computer Society (1st Latin American Web Congress),* Santiago de Chile, pp 103-113, ISBN 0-7695-2058-8, (2003).
9. Olsina L., Rossi G., "Measuring Web Application Quality with WebQEM", *IEEE Multimedia*, 9(4), pp. 20-29 (2002).
10. Olsina, L.; Martin, M. A.; Fons, J.; Abrahao, S.; Pastor, O., "Towards the Design of a Metrics Cataloging System by Exploiting Conceptual and Semantic Web Approaches"*, In Lecture Notes in Computer Science of Springer, Int'l Conference on Web Engineering (ICWE'03)*, Oviedo, Spain, LNCS 2722, 2003, pp. 324-333 (2003).
11. Pastor, O.; Abrahão, S. M.; Fons, J. J., Object-Oriented Approach to Automate Web Applications Development, *2nd Int'l Conference on Electronic Commerce and Web Technologies (EC-Web'01)*, Munich, Germany, Springer Verlag, 16-28, (2001)
12. W3C, WWW Consortium, 2002, "RDF Primer", W3C Recommendation 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
13. Zuse, H., *A Framework of Software Measurement*, Walter de Gruyter, Berlín-NY, (1998).