The *OnBrowser* Ontology Manager: Managing Ontologies on the Grid

Mario Cannataro, Antonio Massara, and Pierangelo Veltri

University "Magna Græcia" of Catanzaro, 88100 Catanzaro, Italy
`{cannataro, veltri}@unicz.it`

Abstract. Although many tools and methodologies exist for ontology editing and management, complete, integrated ontology management systems working on the Grid are needed. Moreover, such ontology management systems should allow the use of knowledge coded into ontologies to different applications in a standard way. In this paper we consider the role of ontology management systems that using metadata spread on the Grid can help Grid programming. The paper presents the design and a first implementation of *OnBrowser*, an ontology manager that provides access to "knowledge objects" coding ontology portions and referring to Grid metadata, through both user interfaces and application programming interfaces.

1 Introduction

Ontology – «The study of being since being», told Aristotle – was born as a philosophic discipline, far from world of technology. Nevertheless, in the last years the explosion of net communications favored an unthinkable phenomenon: ontological aspects of information gained a strategic value. These aspects are independent from information coding, so they can be insulated, recovered, organized and integrated on the basis of *information contents*. Today, the standardization of these contents is crucial for many scientific applications and is necessary to simplify their communications processes.

"An ontology is an explicit specification of a conceptualization" [1]. In other words, an ontology is a shared understanding of some domain of interest, which is often realized as a set of classes (concepts), relations, functions, axioms and instances. Concepts in the ontology are usually organised in taxonomies. An ontology in computer related terms is a hierarchical structured set that describe a domain and that can be used as a schema for knowledge bases. Through ontologies the concepts of a particular domain can be shared between different application domains enriching with semantic the knowledge to model. Ontologies try to capture the semantics of domain expertise by deploying knowledge representation primitives, enabling a machine to understand the relationships between concepts in a domain. Additional knowledge can be captured by logical axioms or rules which derive new facts from the existing ones. An inference engine can draw conclusions based on the rules or axioms to create new knowledge and eventually to solve problems.

At the beginning ontologies were developed in Artificial Intelligence to facilitate sharing and reuse of knowledge; now many research communities are interested in them: knowledge engineering, processing of natural speaking, cooperative informative systems, knowledge management, and more recently Grid Computing.

Although many tools and methodologies exist for ontology editing and management, complete, integrated ontology management systems are needed. In these last years many application domains are building their own domain ontologies, that more and more will be used in metadata management, in resource scheduling and management, in dynamic application composition. The existence of different ontologies poses two big challenges: (i) how to allow the use of such knowledge to different applications in a standard way, and (ii) how to use knowledge coded in different application domains. The former problem could be faced, at a programming level, allowing to access and transfer "knowledge objects", that code in a standard way a portion of an ontology, whereas the latter could be faced using meta-ontologies that cover and connect basic domain ontologies. The access to such "knowledge objects" could be provided both through a GUI or, more importantly, through program-accessible functions, such as APIs or Web Services.

In this paper we will talk about ontologies, their life cycle, and discuss requirements of ontology management on the Grid. When considering a Grid environment, the knowledge coded into ontologies must be linked to metadata spread over Grid nodes. Moreover, changes in Grid state, i.e. changes in resources availability, Grid node state, etc., must be reflected into ontologies. This poses new challenges for ontology life cycle management on the Grid, e.g. adding new resources to the Grid could require either refreshment or a complete updating of an ontology, like adding a new concept.

The presented work aims to face main ontology life cycle phases in a Grid environment. *OnBrowser* is an Ontology Manager whose main goal is to allow the access through programming interfaces to knowledge objects describing Grid resources. The rest of the paper is organized as follows. Section 2 gives a brief background on ontology and ontology life cycle. Section 3 discusses requirements of ontology management on the Grid. Section 4 presents *OnBrowser* overall design and a first prototype implementation. Section 5 discusses some related works. Finally, Section 6 concludes the paper and illustrates future work.

2 Background on Ontology and Ontology Life Cycle

With the increasing of communications and the growth of costs for knowledge acquisition, the integrated access to heterogeneous information and the sharing and reusing of knowledge have assumed an ever increasing importance. In this prospect, the *value added* of a knowledge base is not more only connected to the particular application for that the knowledge base is made, but rises in function of its *reuse* and possibility to be integrated with other knowledge source [2].

Whatever domain or application we consider, some reasons to use ontologies are:

- to share common knowledge between peoples or software agents;
- to allow reuse of knowledge about a domain;
- to make explicit the specific of a knowledge domain;
- to analyze a knowledge domain;
- to categorize items (e.g. sale products or web sites) and their characteristics.

However, main sectors where the application of ontological technique is rising are *Semantic Web, Knowledge Management* and more recently *Semantic Grid*.

2.1 Ontology modelling and engineering

Ontology modelling and management is not an easy task, so engineers can use some tools to automate some of these operations. For example they can automate graphics visualization of ontology to see interlacement of class's relations, avoiding writing code. In the last years, a lot of tools for ontologies have been developed; they can be represented by these groups:

- **Ontology development tools**. These include tools, environments and suites that can be used for building new ontologies from scratch or reusing existing ones.
- **Ontology merge and integration tools**. These tools have appeared to solve the problem of merging or integrating different ontologies on the same domain, e.g. when two companies are merged together, or when it is necessary to obtain a better quality ontology from existing ontologies in the same domain.
- **Ontology evaluation tools**. They appear as support tools which ensure that both ontologies and their related technologies have a given level of quality.
- **Ontology-based annotation tools**. These tools have been designed to allow users inserting and maintaining (semi) automatically ontology-based mark-ups in Web pages.
- **Ontology storage and querying tools**. These tools have been created to allow managing and querying ontologies easily.
- **Ontology learning tools.** These tools are used to (semi) automatically derive ontologies from natural language texts.

Ontological engineering is concerned with the design, modification, application, and evaluation of ontologies [3, 4]. There are different approaches for ontological engineering and in particular for the design phase, among them: *Inspiration, Induction, Deduction, Synthesis,* and *Collaboration*. Hybrids of approaches are also possible. The characteristics of these approaches are summarized in the following.

- **Inspirational approach:** in this case a developer designs an ontology starting from reason about why an ontology is needed, then he/she proceeds to design an ontology that satisfies the specifications.
- **Inductive approach:** in this case the ontology is developed by observing, examining, and analyzing specific case(s) in the domain of interest.
- **Deductive approach:** conversely, a deductive approach to ontology design is concerned with adopting some general principles and adaptively applying them to construct an ontology geared toward a specific case.

- **Synthetic approach:** in this case a developer identifies a base set of ontologies, no one of which subsumes any other.
- **Collaborative approach.** In this case the development is a joint effort reflecting experiences and viewpoints of persons who intentionally cooperate to produce it. Such approach could be useful in Grid environment.

There is another type of approach for ontological engineering; it's called "**from the scratch"**. The following are some examples of this approach: TOVE Methodology; Enterprise Methodology; Sensus Methodology; Bernaras, Laresgoiti, Corera Methodology; Methontology; CyC Methodology; Uschold and King Methodology; Gruninger and Fox Methodology; Kactus Methodology; Onto-To-Knowledge Methodology [8, 9, 10].

Each methodology has a particular approach to develop an ontology, but all have a common aspect: all scratch methodologies obey the following *ontology life cycle* phases: *(i)* specification, *(ii)* conceptualization, *(iii)* formalization, *(iv)* implementation, (v*i)* maintenance. As an example, the specific phases of the commonly used Enterprise methodology that we employ in our system are:

1. Ontology Capture (i.e. identify key concepts and relations, produce unambiguous text definitions, identify terms to refer to such concepts and relations).
2. Ontology Coding (i.e. commit to a meta-ontology, choose a representation language, write the code).
3. Evaluation.
4. Documentation.
5. Guidelines for each phase.

To date, there is not a unique integrated tool that supports all phases of ontology life cycle in a Grid environment, although some projects (see Section 5), such as WebODE, KAON, AKT, Protege, attempt to fulfil them. So people often need to use a lot of tools with proliferations of various formats and an increased development complexity. A complete *ontology manager*, instead, should be able to handle all the different phases of ontology life cycle, facing the complexity and distribution of Grids.

3 Requirements for Ontology Management on the Grid

An important aspect of Ontology Management Systems is how ontologies (e.g. semantic description of resources) are related to described resources and their metadata (e.g. installed software tools, data sources, etc.). For example: should ontology be updated whenever the state of Grid resources changes? I.e., should ontology reflect Grid status? In the following we first recall overall requirements of ontology management, and then we discuss requirements in Grid environment. Main requirements for an Ontology Management System are:

- complete management of the entire life cycle phases, such as ontology definition, creation, consistency checking, updating, importing/exporting;
- ontology querying, it may be provided through a data-oriented query language SQL-like, e.g. RDQL, or through a navigational language, e.g. path expressions;

- ontology reasoning, i.e. the process of knowledge extraction on the knowledge base realized by ontology;
- ontology browsing through interactive GUIs;

Such functions can be provided through APIs, for example to be used by external programs, eventually through Web services technology and usually, through graphical user interfaces. From an architectural point of view, an ontology manager should comprise:

Ontology editor: allowing to define, edit, create, and store ontologies using different languages (e.g. OWL, DAML+OIL, RDF, RDFS) and data formats (N3, N Triplets). Storing should be provided on permanent stores, such as RDBMS;

Ontology browser: allowing to navigate inside concepts of ontology and to follow their relations, using different point of access, through the different implemented taxonomies;

Ontology query engine: allowing to retrieve data describing nodes on ontology, in a format usable by humans or programs. It should provide mechanisms to narrow or enlarge the scope of query on the basis of user needs and query results.

Ontology storage manager: in a distributed environment, ontologies can be partitioned among nodes of the system, or ontologies can refer to metadata stored on distributed nodes (e.g. an ontology of software components could refer to metadata about installed software tools that are spread among the nodes). So, ontology storage manager should be able both to face ontology partitioning and replication, and distributed ontology updating (e.g. when a new software tool is added to the system this could require an update of the ontology). Such update functions could be offered through distributed daemons or through Web Services, whereas local applications could access them through message passing, or Web Services invocation, or by means of APIs.

To introduce ontology management requirements on the Grid, we consider a Grid information system scenario where ontologies are used to describe resources and refer to their metadata to allow resource access. In particular, in a Grid programming environment, resources could be data sources and software components. A reference architecture for such Grid information system is depicted in Fig. 1. In particular we have the following layers:

Resource Repository, containing concrete Grid resources, such as data sources and software components.

Resource Description. Such layer contains both metadata of resources, describing details of each concrete resource on each Grid node, and ontologies that describe semantic properties of resources and semantic relations.

Resource Access. This layer should allow general primitives to access heterogeneous resources (i.e. the same functions to access a text document or a relational database). Considering data sources, the emerging GridDB concept goes along this direction [25]. For the scope of our work, the access can be directly implemented by applications by using metadata.

Information Services. This layer offers a set of functions to query and browse resources, a framework to describe resources and to publish such information in both metadata and ontologies, and finally functions to infer new knowledge by reasoning or mining.

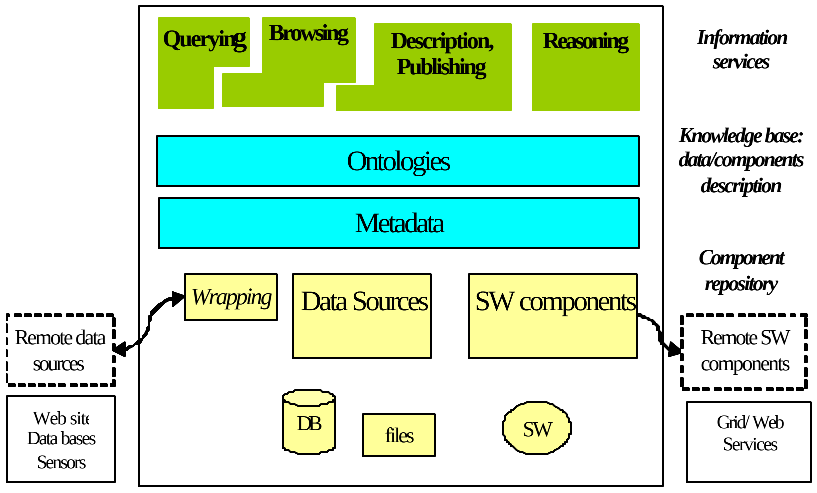


Fig. 1. Conceptual layers in a Grid information system

Main requirements for ontology management on the Grid are:

Reflecting resources state changes in ontologies. In such a scenario, both ontology and metadata management has to be considered in a distributed environment, where Grid nodes can or cannot be connected, and resources can or cannot be available. For example, if we use ontology to enhance Grid programming (i.e. composition of tools), probably the ontology should show all resources whatever they are available or not, since an application should be designed without worry if a resource is currently available. On the other hand, when using the same ontology to dynamically schedule Grid jobs through a request/resource matchmaking (e.g. start a classification job by using C4.5 software), it should clearly refer to only available C4.5 software. In general, the status of concrete resources should be reflected, in some cases, by ontology. In general, ontology should be integrated with Grid metadata and information systems.

Management in a distributed setting. Implementing an ontology as a centralized resource could produce a bottleneck in a Grid environment. So supporting the storing, retrieving and updating of ontologies in a decentralized way is a main requirement in Grids. Even if ontology is centrally stored, functions to replicate it and scale-up access time should be employed.

Cooperative design and editing. In a Grid environment different communities should be able to cooperate during ontology definition and editing, and functions to import and integrate autonomously designed ontologies should be provided.

The *OnBrowser Project* is a first attempt aiming to satisfy main requirements of ontology engineering and ontology use in Grid environment.

4 *OnBrowser*: an Extensible Ontology Manager

Today, the research community establishes OWL as the reference language for ontology and many tools are available for ontology management on the Semantic Web. On the other hand, to manage ontologies on the Semantic Grid [24], it is necessary to develop Grid-aware tools that will manage ontologies and integrate them with application-level and Grid-level metadata management systems and information systems. The *OnBrowser* project aims to develop an Ontology Manager suitable for the Grid environment that integrates different ontology management functionalities currently often spread among different tools.

To create an ontology, usually the developer needs to have specific competences so it is understandable that used tools may have a medium/high level of complexity; on the contrary ontology browsing or querying usually is made from inexpert users so it must be as much easy and intuitive as possible. A main goal of *OnBrowser* is to allow to generic users to concentrate on interested information avoiding they need to know working formalism or learn particular command to use ontologies.

To reach these goals, *OnBrowser* provides a simple and user friendly graphic interface to show an ontology. It provides the possibility to browse ontology levels as users like, leaving them decisions on deep of visualized relations. Browsing is made through an interactive GUI that drives users by simply mouse clicking across visual representation of ontology model.

4.1 Architecture

The current *OnBrowser* release is implemented in Java language and permits to browse DAML+OIL or OWL/RDF ontologies. Current version is implemented as a standalone application, but we plan to release it as a set of distributed services on the Grid.

The *OnBrowser* design employs a layered architecture which main goals are modularity and the possibility to add more functions in an easy way. The architecture comprises four main layers: *Storage*, *Manipulation*, *API*, and *GUI Layer* respectively. Each layer exchanges information with adjacent layers in order to obtain objects of an ontology that a user load usually from the file system or from a relational database. In this release ontology data are handled by a GUI that allows graphic browsing; in future releases we are going to create a set of APIs to permit third parts tools to handle ontology data.

In the following (see Fig. 2), starting from the bottom, we show details of each *OnBrowser* layer to better illustrate functionalities of the system.

The *Storage Layer* manages data structures where ontology model data are stored. In the current release ontologies provided by the user are loaded from the file system. Ontology data are then stored in an optional relational database (using MySQL RDBMS) in N-Triple format description. The database is created when the tool creates ontology model and can be accessed during browsing operations or during ontology updating. In next releases it will be used as persistent knowledge base on which to develop querying APIs.

The *Manipulation Layer* is the core of the system. It allows manipulating elements of ontology included in storage level. In Manipulation layer we identify two sublevels: a Jena API level and a set of methods used to make graphics interface. Jena project [6] is a toolkit of Open Source Java APIs conceived to write Semantic Web applications. The Jena APIs execute the low level ontology operations, in particular: parsing of storage level data, building of the ontology model, manipulation of this model and its possible storing in MySQL DBMS. The high level capacity of Jena APIs as querying and personalized reasoning will be used to develop high level ontology management services as shown in the API layer of Fig. 2.

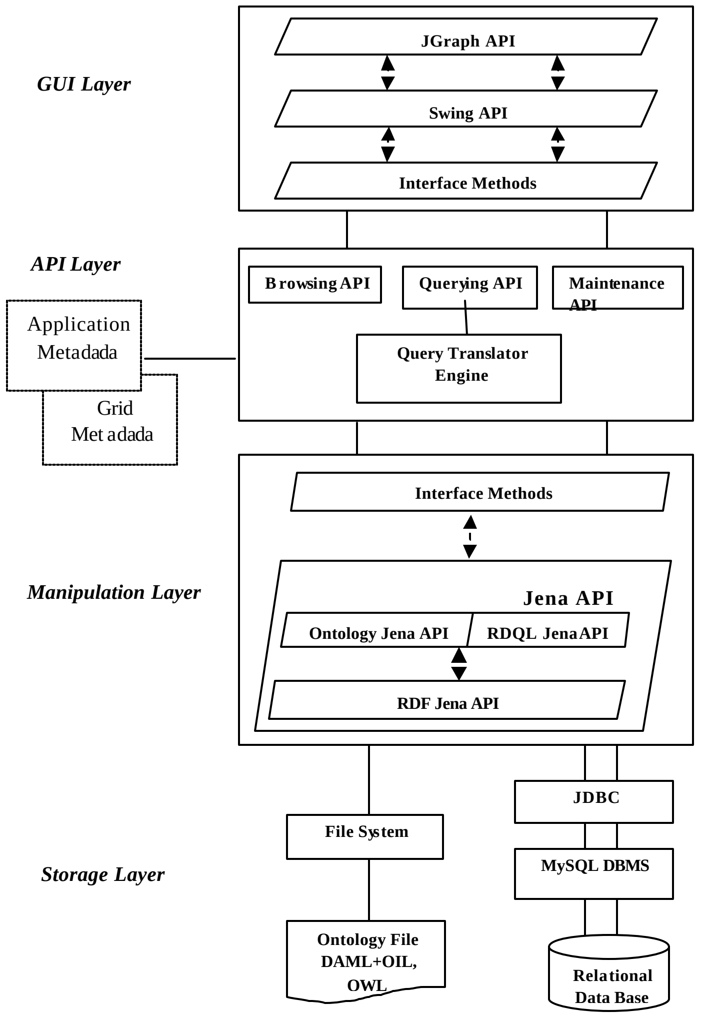


Fig. 2. *OnBrowser* detailed architecture

In the *API layer* we are implementing a set of APIs that permit to third parts software tools to reach the knowledge base and made operations on it. The API implementation is realized for accessing, querying and updating the ontology: the API will provide a set of object-oriented abstractions of ontology elements such as Concept, Relation, Properties, and Instance objects. In particular we designed and partially implemented three classes of APIs: Browsing (already implemented), Querying and Maintenance.

- *Browsing APIs* permit to users or tools to take information about specific ontology class. We have used this method to interpret the questions from GUI level and to ask, through a manipulation language, the knowledge base. In this way we could be able to browse the entire ontology model.
- *Querying APIs* should permit to users or software tools to ask the ontology model, using a quite natural language or directly by using RDQL. This will be possible trough a query translator engine used as interface between query APIs and Manip ulation layer.
- *Maintenance APIs*, instead, should permit to do operations in the maintenance phase of ontology life cycle. For example to add or remove a class from the o ntology model. Maintenance APIs are used also to reflect the state of concrete resources on ontology, as explained in Section 3.

Each set of API can be designed and implemented as Web Services using message passing or as a Java library to be included in a third part project.

The *GUI layer* is the graphics interface of the system that permits users to access ontologies. In this layer we can identify three sublevels: (1) Java Swing API level that manages frames and their elements; (2) JGraph API level, a third part interface that contains methods to elaborate users' and manipulation layer questions. In the GUI layer users actions (e.g. mouse click) are caught, explained and processed to call methods of API layer; so we can obtain answers containing searched information. In particular JGraph [7] is an open-source project totally compatible with Java Swing API to represent however complex graph structures. To represent nodes and arcs it uses concepts as simple as powerful. All arcs and nodes have a particular property called *port*. When we want connect two nodes, simply we associate to the nodes two new ports that will be departs and arrivals ports of the arc that connects the two nodes; in this way we automatically manage orientations arcs. In *OnBrowser* project we utilized basic characteristics of JGraph APIs, which allowed us to represents ontology graph, where actions such as objects drag&drop, nodes and arcs events capture, personalized algorithm to visualize ontology graph, have been implemented. In next release we could utilize advanced features of JGraph API to manage insert and remove ontology nodes directly in visual mode.

4.2 A First *OnBrowser* Prototype

In this section we show some snapshots of the *OnBrowser* prototype when browsing DAMON, an ontology developed for the Data Mining domain describing data mining software tools, data sources and knowledge discovery processes [11].

OnBrowser can read DAML+OIL or OWL [5] ontology files. After specifying ontology file, the tool executes a set of synchronized processes to make ontology parsing of input file. The result of this phase is the building of the first level taxonomy ontology tree and its visualization in the user interface.

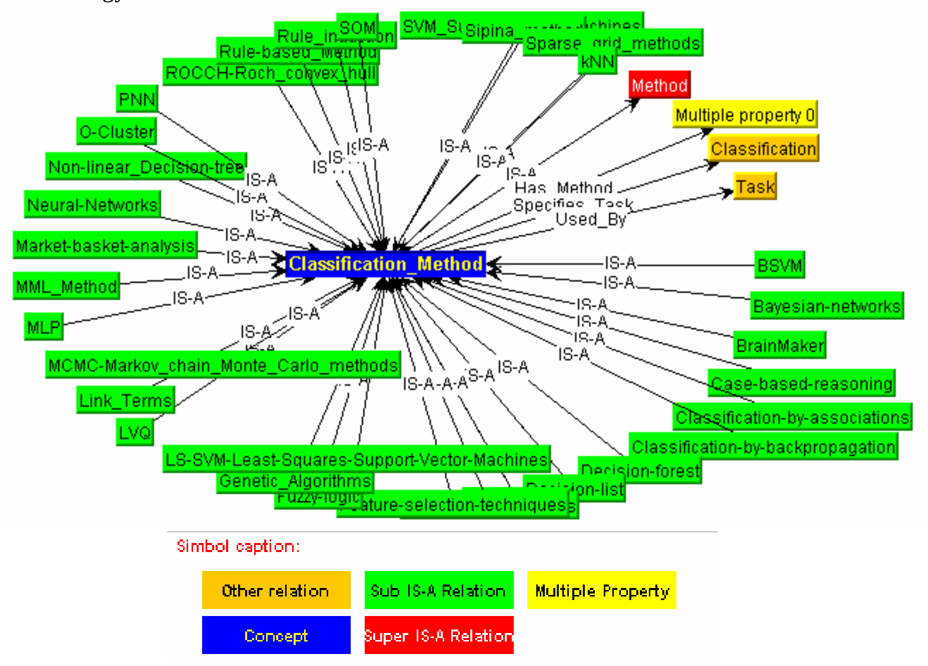
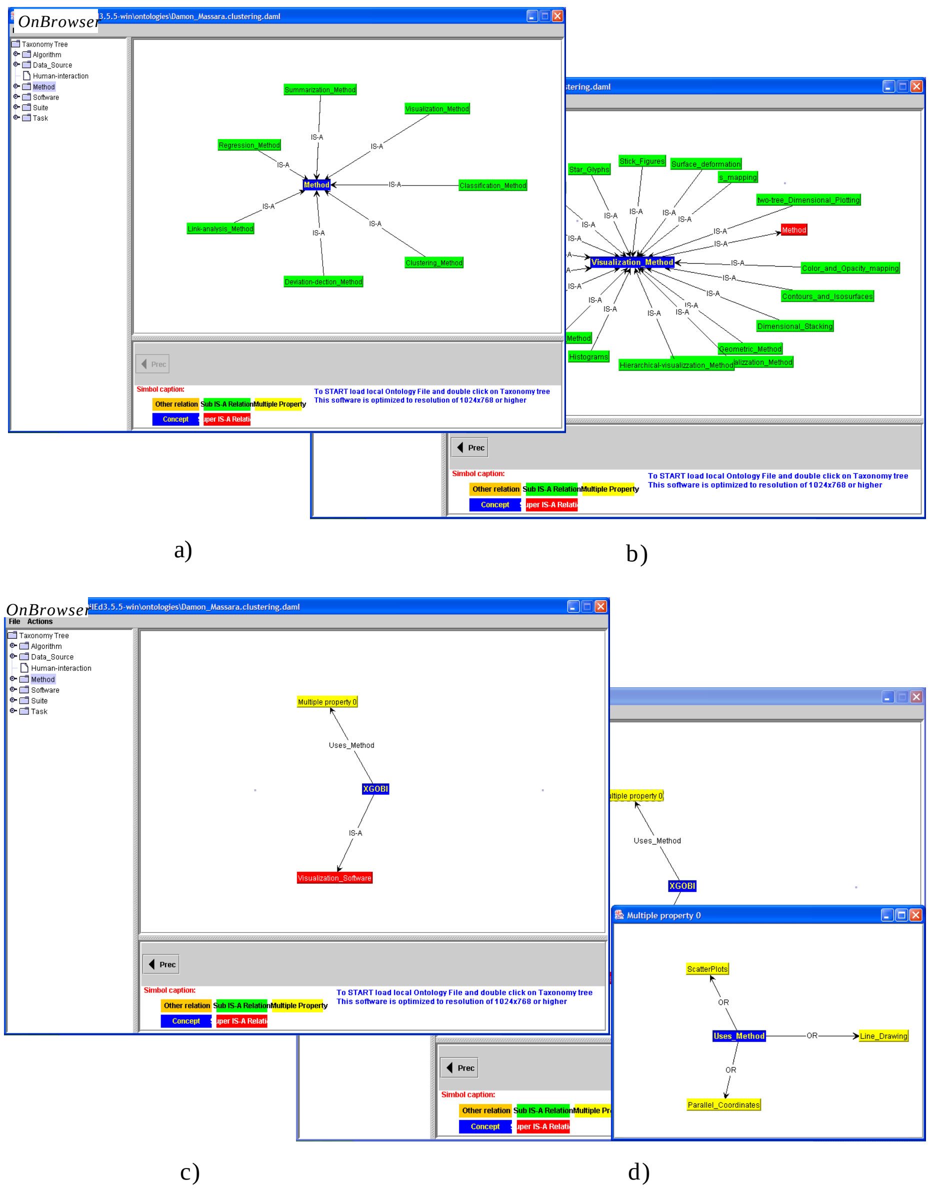


Fig. 3. "Classification_Method" concept and its relations with other ontology concepts

The ontology graph of a particular ontology class is represented as a set of oriented and labelled arcs that show ontology properties and various colour rectangles that show ontology concepts (e.g. ontology class). An example of this kind of representation and symbols chosen to represent ontology concepts are shown in Fig. 3. In particular the label of each arc represents the name of property that connects the two adjacent concepts.

In the centre of ontology graph there is the last clicked concepts, depicted with blue colour. All around there are various relations: *is-a* relations are represented in red (those indicating super-class relations) or in green (those indicating subclass relations), whereas non taxonomic relations are represented in orange. In particular if a non is -a property refers to a set of concepts (e.g. ontology restriction declared on a set of classes) this is represented with yellow colour and label *Multiple_Property_x.*

Trough simple mouse clicks, user can browse ontology concepts as he/she likes. Starting from a taxonomic concept we can reach and browse all ontology levels. When a user clicks on a taxonomic node the corresponding ontology graph is shown in the right frame of the tool (see Fig. 4). In particular Fig. 4a shows the ontology graph of the "Method" concept, and, after clicking the "Visualization_Method" ontology object, its related graph is shown in Fig. 4b. In this pane we choose to focus on the “Line_Drawing” object of the class “Visualization_Method”: this method has a “is-a” relation with “Visualization_Method” object and is related with “Xgobi” object through “Used_by” property.



To START load local Ontology File and double click on Taxonomy tree
This software is optimized to resolution of 1024x768 or higher

Fig. 4. Different snapshots of *OnBrowser* during browsing of DAMON ontology

To obtain information about "Xgobi" object user clicks it and obtain ontology graph stating that Xgobi is a Visualization_Software and uses a set of methods identified with "*Multiple_property_0*" object (see Fig. 4c). "*Multiple_property_0*" is a particular object that has not direct correspondence in ontology model. It represents a set of objects that are connected from the same relations with single other concepts. If user clicks it, the corresponding representation will be showed in a new frame (see Fig. 4d). In this secondary frame user can chose to click on a particular concept to represent it in the main frame or to do nothing and simply close secondary frame. The previous visited ontology graph is always recoverable through the "Prec" button in main frame.

4.3 Ontology-Based Grid Programming

PROTEUS is a Grid-based Problem Solving Environment [19] for composing and running bioinformatics applications on the Grid. We use ontologies for modelling bioinformatics processes and Grid resources, and workflow techniques for designing and scheduling bioinformatics applications. In such environment we use ontology to enhance application composition. In particular, the design and execution of an application on PROTEUS comprises the following steps:

1. *Ontology-based component selection*. Search, location and selection of the resources used in applications are conducted on the domain ontology of PROTEUS through the browsing and querying functions of *OnBrowser*. Such ontology extends DAMON to describe bioinformatics concepts.
2. *Workflow design*. Selected components are combined producing a workflow schema that can be translated into a standard language, such as Business Process Modelling Language (www.bpmi.org). To produce and validate the workflow schema, metadata about tools and data sources, returned by *OnBrowser*, are exploited. In particular PROTEUS uses a metadata/ontology schema as depicted in Fig. 1.
3. *Application execution on the Grid*. The workflow is scheduled by a workflow engine on the Grid. In the current version, workflow schema is static, whereas we plan to implement an ontology-based scheduler that uses both domain ontology (to find available components), and resource ontology (to find available resources where to execute them.
4. *Results visualization and storing*. After application execution and result collection, the user can enrich and extend the domain ontology of PROTEUS.

5 Related Work

There are many research and industrial efforts in the ontology field. Whereas the development of standard languages and technologies for ontology representation and manipulation has reached a large agreement, (e.g. with OWL language), currently there are many independent tools that face different phases of ontology life cycle. As an example, the Ontology Editor Survey (www.xml.com) reports over 50 ontology editors. In the following we briefly describe complete ontology platforms and specific tools.

WebODE (http://delicias.dia.fi.upm.es/webODE/) is an advanced ontological engineering platform that covers and gives support to most of the activities involved in the ontology development process [20].

KAON (http://kaon.semanticweb.org/) is an open-source ontology management infrastructure. It includes a comprehensive tool suite allowing easy ontology creation and management and provides a framework for building ontology-based applications. An important focus of KAON is scalable and efficient reasoning with ontologies [21, 22].

OntoEdit (http://www.ontoprise.de/) is an ontology engineering environment that relies on W3C standards. In OntoEdit ontology development comprises three main phases: Requirements Specification phase produces an ontology requirements specification document describing what an ontology should support; Refinement phase produces a mature and application-oriented ontology; Evaluation phase is as a proof for the usefulness of developed ontologies [23].

SNOBASE (Semantic Network Ontology Base), the IBM Ontology Management System, is a framework for creating, modifying, querying, and storing ontologies (http://www.alphaworks.ibm.com/tech/snobase). SNOBASE provides good support for many phases of the ontology lifecycle. Moreover, it provides a mechanism for querying ontologies and an easy-to-use programming interface for interacting with vocabularies of standard ontology specification languages.

AKT (www.aktors.org) is an ambitious project of Advanced Knowledge Technologies consortium, an interdisciplinary research collaboration of UK EPSRC [17]. The goal of this project is to identify new technologies to create, manage and extract value from knowledge bases, and integrate these technologies to obtain a complete approach to the knowledge life cycle. AKT identifies six main challenges: *knowledge acquisition, modelling, reuse, retrieval, publishing, and maintenance*.

TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) is a system that allows a user to access information coming from heterogeneous bioinformatics data sources using an ontological description of key bioinformatics tasks and concepts and of information sources. The TAMBIS Ontology Server allows the concept-based browsing of managed bioinformatics ontologies [18]

Moreover, some tools that face specific phases on ontology life cycle are:

Ontology editing tools allowing to define and edit ontologies. **OilEd** [12] is a simple graphical tool that supports the construction of OIL/DAML+OIL/OWL-based ontologies. Basic OilEd functionalities allow the definition and description of classes, properties, individuals and axioms through graphical means. OilEd uses FaCT reasoner which allows the user to produce classification hierarchies and check classes for inconsistency. **Protégé** [13] is an ontology editor that provides an extensible architecture for the creation of customized knowledge-based applications. This tool allows the user to construct domain ontology, customize data entry forms and enter data. It has a graphical user interface which enables ontology developers to concentrate on conceptual modelling without knowing about syntax of ontology output language.

Ontology manipulation tools allowing navigating, querying and manipulating ontologies. **Jena** is an open source Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDF Schema and OWL, including a rule-based inference engine [6].

Ontology-based annotation tools, for annotating web resources according to an ontology. For example, the UML Based Ontology Toolset (**UBOT**) [14] supports translation from UML class diagrams to DAML ontologies.

Ontology learning tools, for learning ontologies from natural language documents. **CORPORUM** is a document and information management system [15]. The CORPORUM technology focuses on meaningful content rather than odd data or standardized document parameters. The **Text-To-Onto** system provides an integrated environment for the task of learning ontologies from text [16].

6 Conclusions and Future Work

The paper presented the design and partial implementation of an Ontology Manager, whose main goal is to allow the access to knowledge objects through programming interfaces. *OnBrowser* project aims to face main ontology life cycle phases on the Grid. The *OnBrowser* design provides three different classes of APIs for ontology browsing (i.e. reaching a knowledge object), ontology querying (i.e. collecting a set of knowledge objects), and ontology maintenance (e.g. ontology updating and importing/exporting ontology portions). Its layered architecture leverages Jena open source ontology manipulation functions.

Future work will regard the full implementation of *OnBrowser* functions and its integration in the metadata management system of PROTEUS, a Grid-based Problem Solving Environment for bioinformatics applications [19].

Acknowledgements

This work has been partially supported by Project “FIRB GRID.IT” funded by MIUR. Authors thank Carmela Comito for her collaboration on the *OnBrowser* design.

References

1. T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993. (http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html).
2. Enrico Franconi, Using Ontologies, *IEEE Intelligent Systems* - Special Issue on E-Science, Volume 19, Number 1, January/February 2004. p. 76.
3. Ontology Applications and Design, (Special Issue on) – Communications of ACM, Vol. 45, No.2, February 2002, pp. 39.

4. S. Staab and R. Studer (Eds.), *Handbook of Ontologies*, Springer Verlag, 2004.
5. W3C - OWL Web Ontology Language Reference - http://www.w3.org/TR/owl-ref/
6. Jena Project - http://www.hpl.hp.com/semweb/jena
7. JGraph Project - http://www.jgraph.com
8. Farquhar, Fikes, Rice, The Ontolingua Server: A Tool for Collaborative Ontology Construction, Proceedings of KAW96. Banff, Canada, 1996.
9. M. Fernández-López, A. Gómez-Pérez, Methontology – 3/1.1.4 System Administration Guide, http://www.ontoweb.org/workshop /ontoweb2/slides/methontosig3.pdf
10. Michael Gruninger, Mark S. Fox, Methodology for the Design and Evaluation of Ontologies, http://www.ontoweb.org/workshop/ontoweb2/slides/methontosig3.pdf
11. M. Cannataro, C. Comito. A Data Mining Ontology for Grid Programming. 1st Int. Woprkshop on Semantics in Peer-to Peer and Grid Computing (SemPGrid2003), pp. 113-134, 2003.
12. S. Bechhofer, I. Horrocks, C. Goble, R. Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396—408, 2001.
13. *Protégé system*, http://protege.stanford.edu
14. UML Based Ontology Toolset, http://ubot.lockheedmartin.com/ubot/intro/index.html
15. B.A. Bremdal, F. Johansen. CORPORUM Technology and Applications, CognIT a.s, June 2000. http://www.ontoknowledge.org/downl/CorporumTechApp.pdf
16. The Text-To-Onto System. http://ontoserver.aifb.uni-karlsruhe.de/texttoonto/
17. Nigel Shadbolt (Ed.), Advanced Knowledge Technologies Selected Papers, http://www.aktors.org/publications/selected-papers/
18. R. Stevens et al., TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources, *Bioinformatics*, Vol. 16 (2), 2000, pp.184-185.
19. Cannataro M., Comito C., Lo Schiavo F., Veltri P., Proteus, a Grid based Problem Solving Environment for Bioinformatics: Architecture and Experiments, *IEEE Computational Intelligence Bulletin*, Vol. 3, No. 1, pp. 7-18, February 2004.
20. Corcho O, Fernández-López M, Gómez-Pérez A, Vicente O., WebODE: an integrated workbench for ontology representation, reasoning and exchange. Lecture Notes on Artificial Intelligence Vol 2473. 13th Int. Conf. on Knowledge Engineering and Knowledge Management (EKAW'02). Springer-Verlag. pp: 138-153. October 2002.
21. D. Oberle, R. Volz, B. Motik, S. Staab, An extensible ontology software environment, In *Handbook on Ontologies*, chap. III, pp. 311-333. S. Staab and R. Studer, Eds., Springer, 2004
22. R. Volz, D. Oberle, S. Staab, B. Motik, KAON SERVER - A Semantic Web Management System, *WWW2003, Budapest, Hungary, 20-24 May 2003*. ACM, 2003.
23. Y. Sure, J. Angele, and S. Staab, OntoEdit: Guiding Ontology Development by Methodology and Inferencing, ODBASE 2002.
24. D. De Roure, N.R. Jennings, and N. Shadbolt, “The Semantic Grid: A Future e-Science Infrastructure,” *Grid Computing: Making the Global Infrastructure a Reality*, F. Berman, A.J.G. Hey, and G. Fox, eds., John Wiley & Sons, 2003, pp. 437–470.
25. S. Malaika, Standards for Databases on the Grid. *ACM SIGMOD Record*, Vol. 32, No. 3, Sept. 2003.