A Data Mining Ontology for Grid Programming

Mario Cannataro[1] and Carmela Comito[2]

[1]University "Magna Græcia" of Catanzaro, Via T. Campanella 115,
88100 Catanzaro, Italy
cannataro@unicz.it

[2] University of Calabria, Via P. Bucci 41/c,
87036 Rende, Italy,
comito@si.deis.unical.it

Abstract. The Grid is an integrated infrastructure for coordinated resource sharing and problem solving in distributed environments. The effective and efficient use of stored data and its transformation into information and knowledge will be a main driver in Grid evolution. The use of ontologies to describe Grid resources will simplify and structure the systematic building of Grid applications through the composition and reuse of software components and the development of knowledge-based services and tools. The paper presents an ontology for the Data Mining domain that can be used to simplify the development of distributed knowledge discovery applications on the Grid, offering to a domain expert a reference model for the different kind of data mining tasks, methodologies and software available to solve a given problem, helping a user in finding the most appropriate solution. How the DAMON ontology is used to enhance the design of distributed data mining applications on the KNOWLEDGE GRID is also shown.

1 Introduction

The word "ontology" has a long history in philosophy, in which it refers to the subject of existence. According to the Gruber [6] definition, "an ontology is a specification of a conceptualization". What is important is what an ontology is *for*. Usually ontologies are defined for the purpose of enabling knowledge sharing and reuse, in a way that semantic is independent of reader and context.

Ontologies are used for [27]:

- **communication** between implemented computational systems, between humans or between humans and implemented computational systems;

- **computational inference**, e.g. for internally representing and manipulating plans and planning information, and for analyzing the internal structures, algorithms, inputs and outputs of implemented systems in theoretical and conceptual terms;
- **reuse** and **organization of knowledge**, e.g. for structuring and organizing libraries or repositories of plans and planning and domain information.

The emerging discipline of "ontological engineering" has the goal to support ontology development and use throughout its entire life cycle – the design, evaluation, validation, maintenance, deployment, mapping, integration, sharing and reuse of ontologies within intelligent systems.

The *Grid* is an integrated infrastructure for coordinated resource sharing and problem solving in distributed environments. Grid applications often involve large amounts of data and/or computing, and are not easily handled by today's Internet and Web infrastructures [7]. Since their birth, Grids have traversed different phases or generations [8]. In the early 1990s, first-generation Grids (Computational Grids) allowed to interconnect large supercomputing centers. Second-generation Grids are characterized by their capability to link more than just few regional or nation-wide supercomputing centers, and by the adoption of standards (such as HTTP, LDAP, PKI, etc.) that enable the deployment of global-scale computing infrastructure. The motivation for third-generation or next-generation Grids [15] is to simplify and structure the systematic building of Grid applications through the composition and reuse of software components and the development on knowledge-based services and tools.

The *Open Grid Services Architecture*, the Semantic Grid and Knowledge Grids are the most promising approaches towards next-generation Grids.

Following the trend emerged in the Web community, the Open Grid Services Architecture (OGSA) introduced *service orientation* in Grids, leveraging the results of Web Services [10, 11]. A Grid Service is a Web Service that conforms to a set of conventions for the controlled, fault resilient and secure management of stateful services and exposes capabilities via standard interfaces.

The *Semantic Grid* is an initiative of the UK EPSRC/DTI Core e-Science Program that aims to incorporate the Semantic Web approach (systematic description of resources through metadata and ontologies, and provision for basic services about reasoning and knowledge extraction), into the Grid [9]. The research issues of the Semantic Grid covers many aspects of the next-generation Grids: i) full support of the three recognized layers composing a Grid, i.e. computation/data layer, information layer (where information id obtained from data), and knowledge layer (where knowledge can be used to take decision); and ii) provision of seamless, pervasive and secure use of resources.

Knowledge Grids offer high-level tools and techniques for the distributed mining and extraction of knowledge from data repositories available on the Grid, thus they realize the higher layer of the Grid architecture [12, 13]. Main issues for the development of the knowledge layer in Grids are: i) synthesizing useful and usable knowledge from data, and ii) leveraging the Grid infrastructure to perform sophisticated data-intensive large-scale computation. The integration of knowledge discovery techniques (mainly based on Data Mining) in Grid environments must pass through a unambiguous representation of the knowledge base (through metadata and ontologies) needed

to translate moderately abstract domain-specific queries into computations and data analysis operations able to answer such queries by operating on the underlying systems. The KNOWLEDGE GRID is a joint research of ICAR-CNR, University of Calabria, and University of Catanzaro, Italy, aiming at the development of an environment for geographically distributed high-performance knowledge discovery applications [14].

Along these recent trends in the development of the Grid, this paper presents the design and development of an ontology for the Data Mining domain whose main goal is to easy the development of distributed knowledge discovery applications on the Grid. The development of KDD (Knowledge Discovery in Databases) applications usually requires the joint work of technology (data mining) and domain (application) experts. The main goal of the proposed ontology is to offer to a domain user a reference model for the different kind of data mining tasks, methodologies and software available to solve a given problem, helping him/her in finding the most appropriate technological solution.

The paper presents DAMON (DAta Mining ONtology), an ontology for Data Mining domain, and its DAML+OIL implementation. Moreover, the architecture and main functions of DAMON-MAP, a Java-based tool for ontology browsing and querying, is also described. The use of DAMON is exemplified in the design of distributed KDD applications on the KNOWLEDGE GRID.

The rest of the paper is organized as follows. Section 2 introduces ontology basic definitions. Section 3 describes the design and implementation of DAMON. Section 4 describes architecture and main functionalities of DAMON-MAP, a tool for ontology browsing and querying. Section 5 shows how the DAMON ontology and its tool can be used to enhance the component-based design of KDD applications on the KNOWLEDGE GRID. Finally, Section 6 draws conclusions and future work.

2 What is an Ontology?

The word **ontology** came from philosophy. From a philosophical viewpoint, "Ontology" (without the indeterminate article and with the uppercase initial) is the branch of philosophy which deals with the nature and the organization of reality [19].

At the computer science domain, ontologies aim at capturing domain knowledge in a generic way and provide a commonly agreed understanding of a domain, which may be reused and shared across applications and groups [20]. Ontologies provide a common vocabulary of an area and define the meaning of the terms and the relations between them.

We refer to term Ontology as the shared understanding of some domains of interest, which is often conceived as a set of classes (concepts), relations, functions, axioms and instances. Concepts in the ontology are usually organised in taxonomies.

3 DAMON: a DAML+OIL Ontology for Data Mining

This Section presents the design and development of DAMON (*DAta Mining ONtology*), an ontology for Data Mining domain. After a brief introduction of main tasks and goals of Data Mining applications, we show an abstract specification of the ontology and its implementation in DAML+OIL, a commonly used ontology language.

3.1 DAML+OIL Ontology Language

DAML+OIL [22, 3] is an ontology language designed for the Web built upon XML and RDF.

DAML+OIL is modelled through an object-oriented approach, and the structure of the domain is described in terms of classes and properties. DAML+OIL classes can be names (URIs) or expressions and a variety of constructors are provided for building class expressions. The axioms supported by DAML+OIL makes it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, the equivalence or non-equivalence of individuals, and various properties of properties. Classes can be combined using conjunction, disjunction and negation. Within properties both universal and existential quantification are allowed, as well as more exact cardinality constraints. Range and domain restrictions are allowed in the definition of properties, which themselves can be arranged in hierarchies.

DAML+OIL supports two kinds of reasoning tasks:

- The automatic determination of subsumption between compositional descrip-tions. Given two conceptual definitions A and B, we can determine whether A subsumes B, in other words whether every instance of B is necessarily an instance of A. Consequently, a collection of conceptual definitions can be organized into a multi-axial classification based on the subsumption relation inferred by reasoning about the definitions of the concept expressions.
- A concept satisfiability test on an arbitrary lass expression to test for its logical coherency with respect to the concepts in the ontology.

We used the OilEd [5] graphical tool for developing the ontology. OilEd is a simple ontology editor that supports the construction of OIL/DAML+OIL-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.

3.2 Data Mining and Knowledge Discovery in Databases

Knowledge Discovery in Databases (KDD) is *"the non trivial process of identifying novel, potentially useful, and ultimately understandable patterns in data"* [29]. KDD is an iterative and interactive process involving many steps, where Data Mining is an

essential phase of the overall process. Data Mining (DM) refers to the algorithms and methodologies used to extracts patterns or models from observed data.

The main steps of (distributed) KDD are the following [23]:

1. **Data cleaning** is the removal of irrelevant data from data sources.
2. **Data integration**, is the combination of multiple, often heterogeneous data sources, that hides their heterogeneity and availability. The data cleaning and integration often results in a central data warehouse.
3. **Data transformation**, i.e. the data sources are optionally transformed and consolidated for mining, for example through aggregation and summarization.
4. **Data selection** is the extraction of the data relevant to the analysis from the data sources. It depends by the type of analysis being conducted and by the data mining algorithm used.
5. **Data mining**, i.e. the application of data mining algorithms on the cleaned, filtered and optionally integrated data sources, to discover new patterns and models. In distributed data mining, models and patterns can be built by combining partial results coming from different DM processes.
6. **Pattern evaluation**, where the truly interesting and useful patterns/models are selected on the basis of some interestingness measure.
7. **Knowledge presentation**, the extracted knowledge is represented and visualized to the user in an understandable manner, using visualization and representation techniques.

The steps 1 to 4 can be viewed as a *pre-processing phase* and usually in a traditional environment they are conducted once, prior to start analysis, and need not to be repeated. Step 5 is the core *execution phase*, and usually is conducted in a iterative and interactive way, finally, steps 6 and 7 constitute the *analysis and visualization phase.* In this paper we focus on the modelling of the core execution phase (data mining), in the future we plan to model the entire KDD process.

3.3 DAMON Design

A *domain ontology* is a conceptualization of an application domain described through a set of classes (concepts) and their hierarchical relationships. Such kind of ontology is frequently used as a classification system. Several methodologies for developing ontologies have been defined [24], here we have adopted the "Enterprise Methodology" which comprises the following steps.

The **first step** in the ontology development is the definition of the domain and scope of the ontology itself: in our scenario the ontology will cover the Data Mining domain. To build a consistent ontology model it is necessary to establish for **what** we are going to use the ontology and for **what types of questions** the information in the ontology should provide answer. The choice of how to structure an ontology determines what a system can know and reason about. We have built our ontology through a characterization of data mining software that is classified on the basis of some parameters useful to select the more ones software to solve a KDD problem. The need to characterize the data mining domain arises in the context of our work on the KNOWL-

KNOWLEDGE GRID: we would like to help and guide the KNOWLEDGE GRID user in the visual composition of data mining applications.

The main goals of the proposed ontology are (see later):

- to allow the semantic search (concept-based) of data mining software and others data mining resources;
- to assist the user suggesting him/her the software to use on the basis of the user's requirements/needs.

The **second step** identifies key concepts and properties. To this end it is useful to write down a list of all terms we would like either to make statements about or to explain to a user, establishing what we would like to say about those terms and what properties those terms should have. A top-down development process that starts with the definition of the most general concepts in the domain and subsequent specialization of the concepts has been adopted. In fact, we first defined classes for the general data mining concepts, and then we specialised them.

As we just said the ontology should represent the features of the available data mining software, classifying their main components and evidencing the relationships and the constraints among them. The categorization of the data mining software has been made on the basis of the following classification parameters:

- the data mining task performed by the software, e.g. the results produced or the knowledge discovered;
- the type of methodologies that the software uses in the data mining process;
- the kind of data sources the software works on;
- the degree of required interaction with the user, e.g. if the mining process is completely autonomous, or if it requires the user intervention.

On the basis of those parameters important terms related to data mining will include: *Task, Method, Algorithm, Software, Suite, Data Source, Human Interaction.*

- A (data mining discovery) *Task* represents a data mining technique for extracting patterns from data; it is the knowledge discovery task that the software is intended for. In other words a task specifies the goal of a data mining process.
- A *Method* is a data mining methodology used to discover the knowledge; different methods serve different purposes. It can be thought as a structured manipulation of the input data to extract knowledge.
- An *Algorithm* is the way through which a data mining task is performed.
- A *Software* is an implementation (in a programming language) of a data mining algorithm.
- A *Suite* implements a set of data mining algorithms: every algorithm may perform different tasks and employ different methods to achieve the goal.
- *Data Source* is the input on which data mining algorithms work to extract new knowledge.
- *Human Interaction* specifies how much human interaction with the discovery process is required and/or supported.

Once we have defined the basic concepts, we must describe their internal structure by means of properties. Considering the data mining domain the types of object properties in the ontology could be:

- "*extrinsic*" properties usually refer to a peculiar property of a concept. An extrinsic property could be the computational complexity (e.g., belonging to P or NP, upper and lower bounds, etc.) of an algorithm.
- "*part*", if the object is structured (such as the various algorithms implemented by a Suite);
- "*relationship*" to other concepts (e.g., the property *performs task* represents a relationship between a data mining task and an algorithm).

The **third step** defines a concepts hierarchy through taxonomic relations. Taxonomies are used to organise ontological knowledge in the domain using two kinds of relationships through which simple/multiple inheritance could be applied:

- *specialisation/generalisation* ("is-a"): specialises/generalises general/specific concepts in more specific/general ones. An "is-a" relation states that a class A is a subclass of B if every instance of A is also an instance of B. For example Classification or Clustering are subclasses of Data Mining Task.
- *part of/has part*: defines a partition as subclass of a class. In our ontology the "has part" relationship is established between Suite and Algorithm, i.e. "Suite is composed of a set of data mining Algorithms".

A major approach to taxonomy organization is having a large number of small local taxonomies that may be linked together via relations or axioms.

In DAMON we have several small local taxonomies derived from the specialization of the basic classes. Fig. 1 shows the taxonomy obtained by creating subclasses of the data mining Task.

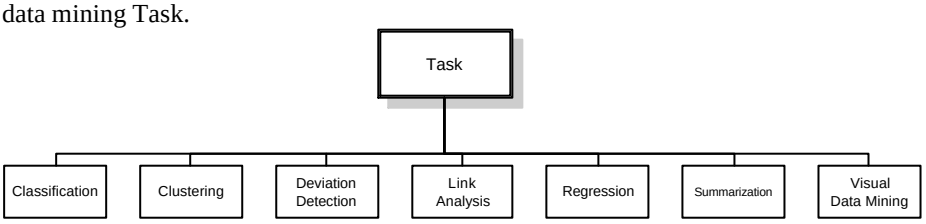

Fig. 1. Data Mining Task taxonomy

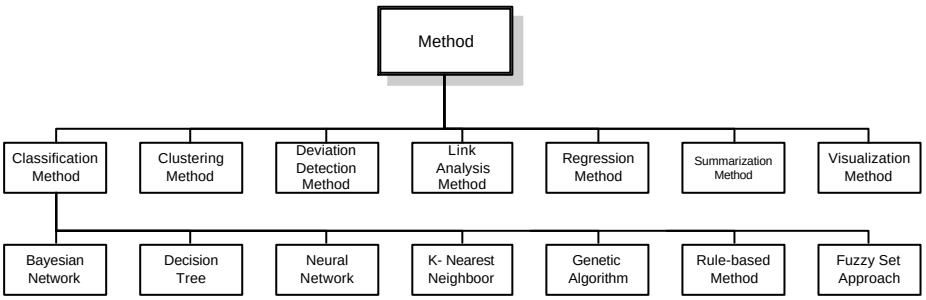

Fig. 2. Data Mining Method taxonomy

Fig.s 2 to 4 show the taxonomies of the other basic concepts (Method, Algorithm, and Software) of our ontological model; for each of these taxonomies we construct a subsequent specialization level for every task identified in the taxonomy of Fig. 1. Due

to specialization relationship that is encoded in the hierarchy, the set of objects in a subclass is a subset of its super class. To simplify the layout of taxonomic structures and to give a better understanding of them, only some specialization of the *Classification* task are shown.

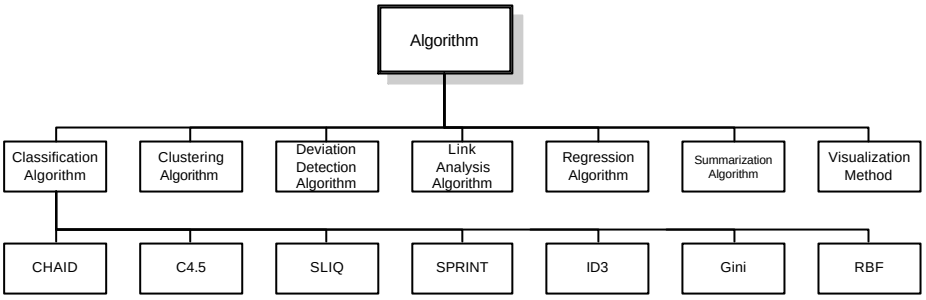

Fig. 3. Data Mining Algorithm taxonomy

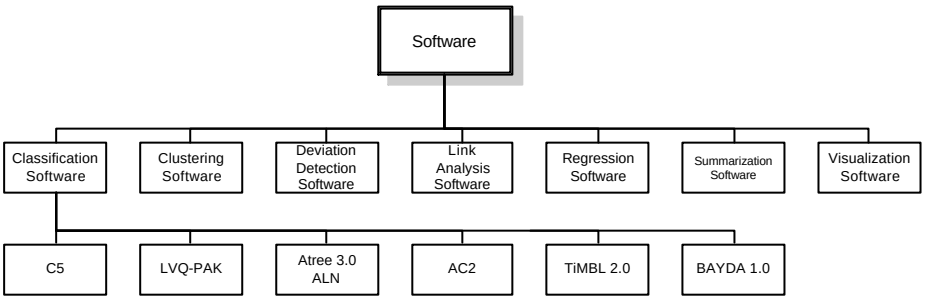

Fig. 4. Data Mining Software taxonomy

The **fourth step** in the development process is the construction of *axioms*. Axioms are formal assertions that model sentences that are always true. Axioms provide a way of representing more information about concepts, such as constraining on their internal structure, and their mutual relationships, verifying their correctness or deducting new information.

In the data mining world, we can use axioms to represent, for example, the following constraints:

- Mutual constraints on the values of several properties of a single concept; for example for the *Classification Algorithm* concept, the *UsesMethod* property must have *Classification Method* as filler and the *PerformsTask* property must have the *Classification Task* as filler.
- Facts about the relations among objects, such as "Every *Software* implements an *Algorithm*".
- Constraints on property or role values for related concepts. The *Software* and *Algorithm* concepts are related through the *ImplementsAlgorithm* property; this relationship constraints the *ImplementsAlgorithm* property to have as value an algorithm referring to the same task referred by the related software. For example

the *ImplementsAlgorithm* property of a *Classification Software* must have as filler a *Classification Algorithm.*

In Fig. 5 a small part of the ontology regarding the conceptualization of the C5 *Classification Software* is illustrated. It should be noted that other than the hierarchical concepts classification (is-a relations), the relationships (realized by means of properties) among concepts belonging to different taxonomies are also shown. In the example, C5 *is a Classification software* that implements the *C5 Algorithm.* C5 Algorithm is a *Classification Algorithm* performing (*PerformsTask* property) the *Classification* task and using (*UsesMethod* property) the *Decision Tree* method that is constrained to be a *Classification Method* specifying (*SpecifiesTa*sk property) the *Classification* task.

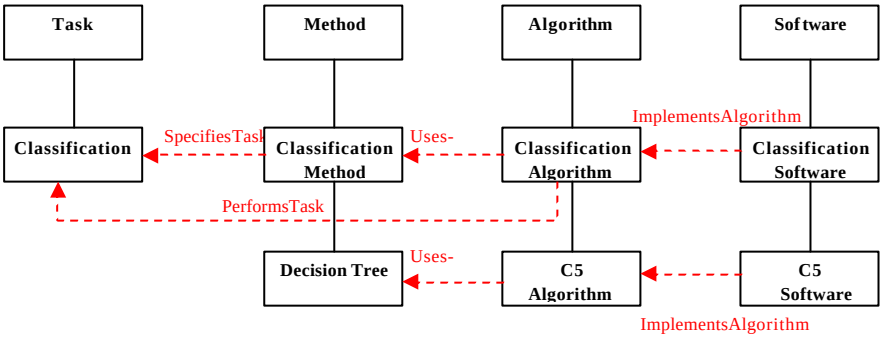

Fig. 5. A fragment of the DAMON ontology for the C5 software

3.4 DAMON Implementation

This Section describes the DAML+OIL implementation of DAMON. Due to space reasons, only the basic ontology concepts Task, Method, Algorithm, Software, Suite with respect to the data mining Classification are shown.

The following DAML+OIL code (Fig. 6) expresses that *Classification* is a Data Mining *Task*.

```
<daml:Class rdf:about="file:/C:/OilEd/ontologies/Data Mining.
daml#Classification">
  ...
  <rdfs:subClassOf>
   <daml:Class rdf:about="file:/C:/OilEd/.../#Task"/>
  </rdfs:subClassOf>
```

Fig. 6. Description of the Classification concept

The DAML+OIL statement of Fig. 7, describes the *Classification Method* concept. The `subClassOf` element asserts that its subject *Classification Method* is a subclass of the resource identified by `#Method`. *Classification Method* is further a subclass of a property (*HasMethod*) restriction that restricts the property's filler type to a disjoint union of classification methods. The code shown in Fig. 8 describes the *Classification Algorithm* concept: it is a subclass of the *Algorithm* class, and it uses a *Classification Method* (i.e. *Decision Tree*) and performs the *Classification* task.

```
<daml:Class rdf:about="file:/C:/.../#Classification Method">
 <rdfs:label>Classification Method</rdfs:label>
 <rdfs:subClassOf>
  <daml:Class rdf:about="file:/C:/.../#Method"/>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
 <daml:Restriction>
 <daml:onProperty rdf:resource="file:/C:/.../#HasMethod"/>
   <daml:disjointUnionOf parseType="daml:collection">
     <daml:Class rdf:about="file:/C:/.../#Decision Tree"/>
   <daml:Class rdf:about="file:/.../#GeneticAlgorithms"/>
  </daml:disjointUnionOf>
```

Fig. 7. Description of the Classification Method concept

```
  <daml:Class       rdf:about="file:/C:/.../#ClassificationAlgo-
rithm">
    <rdfs:subClassOf>
     <daml:Class rdf:about="file:/C:/...#Algorithm"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <daml:Restriction>
      <daml:onProperty rdf:resource="file:/C:/...#UsesMethod"/>
      <daml:toClass>
       <daml:Class
rdf:about="file:/C:/.../#ClassificationMethod"/>
      </daml:toClass>
     </daml:Restriction>
     </rdfs:subClassOf>
     <rdfs:subClassOf>
      <daml:Restriction>
       <daml:onProperty
rdf:resource="file:/C:/.../#PerformsTask"/>
```

Fig. 8. Description of a Classification algorithm

The DAML+OIL code of Fig. 9 describes the C5 algorithm, i.e. a specific *Classification Algorithm*, that uses the Decision Tree *Classification Method*.

```
<daml:Class rdf:about="file:/C:/OilEd/.../#C5 Algorithm">
 <rdfs:label>C5 Algorithm</rdfs:label>
  ...
 <rdfs:subClassOf>
  <daml:Class rdf:about="file:/.../#Classification Algo-
  rithm"/>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
 <daml:Restriction>
  <daml:onProperty
rdf:resource="file:/C:/OilEd/.../#UsesMethod"/>
  <daml:toClass>
    <daml:Class rdf:about="file:/C:/OilEd/.../#Decision
   Tree"/>           </daml:toClass>
```

Fig. 9. Description of the C5 classification algorithm

Fig.s 10 and 11 respectively describe a specific classification software (C5) and a specific data mining suite (IND). The C5 software implements the C5 algorithm, whereas the IND suite implements two algorithms (CHAID and Gini).

```
<daml:Classrdf:about="file:/C:/OilEd/ontologies/DataMining.dam
l#C5">
    ...
    <rdfs:subClassOf>
      <daml:Class      rdf:about="file:/C:/.../#Classification
Software"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <daml:Restriction>
     <daml:onProperty
rdf:resource="file:/.../#ImplementsAlgorithm"/>
      <daml:toClass>
```

Fig. 10. Description of the C5 classification software

```
  <daml:Class rdf:about="file:/C:/.../#IND">
    <rdfs:label>IND</rdfs:label>
    <rdfs:subClassOf>
     <daml:Class rdf:about="file:/C:/.../#Suite"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
     <daml:Restriction>
     <daml:onProperty
rdf:resource="/.../#ImplementsAlgorithm"/>
      <daml:disjointUnionOf parseType=”daml:collection”>
       <daml:Class rdf:about="file:/C:/.../#C4.5"/>
       <daml:Class rdf:about="file:/C:/.../#Gini"/>
```

Fig. 11. Description of the IND suite

Finally, Fig. 12 draws a OilEd screenshot showing a portion of the DAMON ontology.

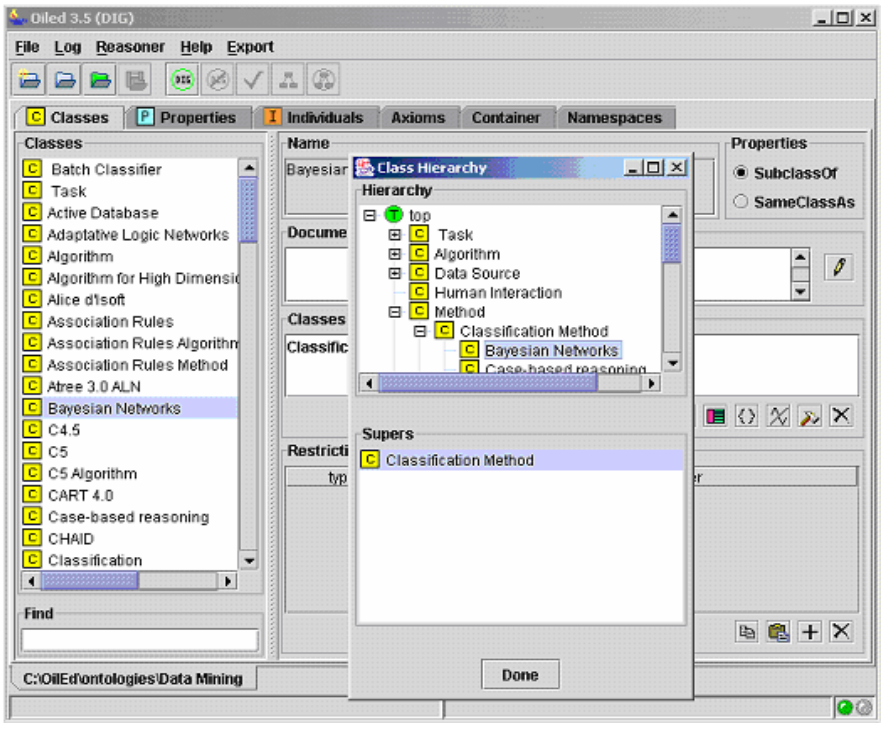

Fig. 12. DAMON ontology viewed by OilEd

4. DAMON-MAP: a tool for ontology browsing and querying

We have designed DAMON-MAP, a tool that allows the manipulation of DAMON ontology. The manipulation of our ontology can be realized both by a user that accesses DAMON-MAP through a graphical user interface, and by a Java-based component through the DAMON APIs. The API implementation is realized for accessing and querying the ontology: the API will provide a set of object-oriented abstractions of ontology elements such as Concept, Relation, Properties, and Instance objects providing query facilities.

DAMON-MAP (Fig. 13) is designed in a client-server three layers architecture consisting of the *Storage Layer*, the *Management Layer*, and the *Client Layer*.

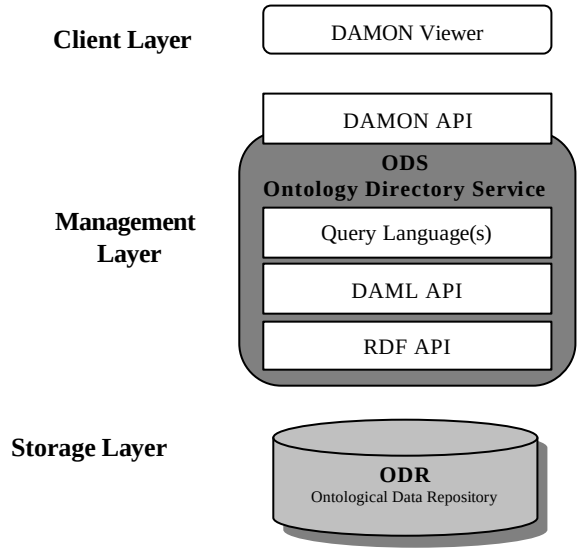

Fig. 13. DAMON-MAP architecture

The Storage Layer allows access to physical ontological data implemented as RDF Schema files stored in the *Ontological Data Repository (ODR)*. Currently this repository is implemented through the file system.

The Management Layer is the heart of the system. Its main goal is to provide a set of APIs (called DAMON API) that allows to access ontology elements. The APIs can be used to search and navigate the ontology structure. The Management Layer comprises four sub-layers:

- at the bottom the RDF API supports the basic operations of the Management Layer such as creation, manipulation and querying of RDF graphs. The API defines interfaces for parsing and accessing RDF models as sets of statements. Default implementations for those interfaces are included in the `org.w3c.rdf.implementation` package. [13];

- above the RDF API sits an inference layer realized through the DAML API implemented by the AT&T [25]. The DAML API V0.7 is a collection of Java interfaces and utility classes that implement an interface for managing DAML ontologies. The DAML API requires the RDF API.
- the querying layer builds upon RDF API and DAML+OIL API, is based on a DAML+OIL/RDF query languages (e.g. RQL, RDQL or DQL [17]). RDQL is an implementation of an SQL-like query language for RDF [4]. It treats RDF as data and provides query with triple patterns and constraints over a single RDF model. The target usage is for scripting and for experimentation in information modeling languages. RQL is a typed language and supports generalized path expressions featuring variables on both labels for nodes (i.e., classes) and edges (i.e., properties) [18].
- and finally at the top of the Management Layer sits the DAMON API that uses the functionalities of the preceding layers.

The Client Layer (DAMON Viewer) is a graphical interface through which the user can search and browse the ontology.

4.1 DAMON Operations

We can use the ontology to obtain information about Data Mining. In general, we have two ways to find the data we are looking for: we can *search*, and we can *browse*. The DAMON Viewer is a graphic interface to DAMON-MAP that provides a combined search and browse facility over the ontology. The DAMON Viewer is a navigation facility that presents an overview of the whole data set: it shows the classes, their relations and instances.

Ontology browsing. The DAMON Viewer gradually presents deeper levels of the ontology: the user starts at the top of the ontology and can navigate towards more specific topics by clicking the classes of interest (diving into the information). At any point, the map shows the current class, its parent and its subclasses.

Ontology-based domain specific search. A user can use the ontology-based search engine to query very detailed information about Data Mining resources annotated in DAML+OIL. The result set of the query is very accurate, because the semantic content of the terms searched is clearly indicated by concepts from the underlying ontology. Our ontology-based search engine will support several kinds of simple inference that can serve to broaden queries including equivalence, inversion, generalization, and specialization. Equivalence uses the DAML+OIL *samePropertyAs* and *sameClassAs* relations to restate queries that differ only in form. Generalization and specialization utilize the *subPropertyOf* and *subClassOf* relations to find matches or more general or more specific classes and relations. If the result set of a query is empty, the user can at least find objects that partially satisfy the query: some classes can be replaced by their superclasses or subclasses. Both narrowing and broadening the scope of the query are possible due to the ontological nature of the domain description.

Our tool can also help the user in the query formulation. Users encounter difficulties when having to provide terms that best describe their information need (vocabulary problem). In DAMON the classes that describe the domain of interest are explicitly shown, making the vocabulary choice much easier.

5 Ontology-based Application Design on the KNOWLEDGE GRID

As said before, the main goal of the DAMON ontology is to offer to a domain user a reference model for the different kind of data mining tasks, methodologies and software available to solve a given problem, helping a user in finding the most appropriate technological solution. The use of DAMON is exemplified in the design of distributed data mining applications on the KNOWLEDGE GRID, an environment for grid-based geographically distributed high-performance knowledge discovery applications [14].

5.1 The KNOWLEDGE GRID

The *KNOWLEDGE GRID* [14] is an environment to design, deploy and execute distributed data mining applications on the Grid. The *KNOWLEDGE GRID* architecture is defined on top of Globus [26], a widely used toolkit to deploy geographically distributed grids. The *KNOWLEDGE GRID* services are organized in two hierarchic levels: the *Core K-grid layer* and the *High level K-grid layer*.

The Core K-grid layer offers the basic services for the definition, composition and execution of a distributed knowledge discovery computation over the grid. Its main goals are the management of all metadata describing features of data sources, third party data mining tools, data management, and data visualization tools and algorithms. Moreover, this layer coordinates the application execution by attempting to fulfill the application requirements and the available grid resources.

The High-level K-grid layer includes services used to compose, validate, and execute a parallel and distributed knowledge discovery computation. Moreover, the layer offers services to store and analyze the discovered knowledge.

A first prototype of the KNOWLEDGE GRID (VEGA) allowing the design, development ad deployment of parallel and distributed KDD applications on the Grid has been implemented. VEGA is an environment running on the top of Globus that offers a graphical interface for the visual composition of data mining tools and data sources to be mined, to obtain an abstract execution plan (i.e. an abstract description of the application). The execution plan is then translated in source Globus RSL scripts, used to deploy and execute the application on the Grid. Further details can be found in [28].

5.2 Ontology-based Application Design

With the DAMON ontology we aim to realize a semantic modelling of user's tasks/needs and of the resources characterizing a data mining software, to offer high level services for dynamic software finding and applications composition.

DAMON will be used as an ontology-based assistant that suggests the KNOWLEDGE GRID application designer what to do and what to use on the basis of his/her needs as well as a tool that makes possible semantic search (concept-based) of data mining software. In other words, it can be used to enhance the application formulation and design, helping the user to select and configure the most suitable Data Mining solution for a specific KDD process.

The ontology could help the different users of the KNOWLEDGE GRID:

- Data mining novices. In this case the ontology-based assistant is a necessary component that guides the user to select and configure the most suitable Data Mining solution for a specific KDD process. Without DAMON, such kind of user should necessary consult a data mining expert to define his/her application. Actually the KNOWLEDGE GRID requires that user exactly know what kind of application has to be built.
- Data mining experts who know exactly how to configure a data mining application. Such users can query DAMON with more specific details on the different data mining resources.

In the following we show how the proposed ontology-based assistant will be used and integrated in the current KNOWLEDGE GRID architecture.

Currently, the design of a knowledge discovery application on the KNOWLEDGE GRID runs through the following steps:

1. Search and selection of the resources to be used in the knowledge discovery application;
2. Visual composition of the application through a graphic model that represents the involved resources and their relations (i.e. the data and control flows);
3. Generation of the execution plan corresponding to the graphic model of the application.

At the first step of the design process the ontology would play an important role. In fact, other than find which data mining software are available, where software can be found, and how software can be accessed, a user could benefit from DAMON by searching all the available software that satisfy some user requirements, such as performing a given task (e.g. classification), implementing a given algorithm (e.g. CHAID), using a specific methodology (e.g. decision trees), and requiring a specific input format. Some possible DAMON queries are:

- find data sources about a specific topic;
- find software implementing a desired data mining algorithm;
- find software performing a specified data mining task;
- find software/algorithm using particular methods.

Fig. 14 shows the KNOWLEDGE GRID architecture enriched by our DAMON ontology and tool. The basic metadata about installed data mining tools and data sources

are stored into the *Knowledge Metadata Repository* (KMR) and managed by the *Knowledge Directory Services* (KDS). In fact, metadata currently implemented in the KNOWLEDGE GRID are tightly bounded to the data mining software and data sources installed on a given physical node. For example, if two instances of the same data mining software, e.g. `AutoClass`, are respectively installed on nodes A and B, their metadata are replicated on the KMR of such nodes. Such metadata contain all the information that can be used by a client to access and use particular installed software (technical parameters, availability, location and configuration of data mining software and tools). Currently, the search of data mining software is conducted by browsing each KMRs and selecting the more appropriate version.

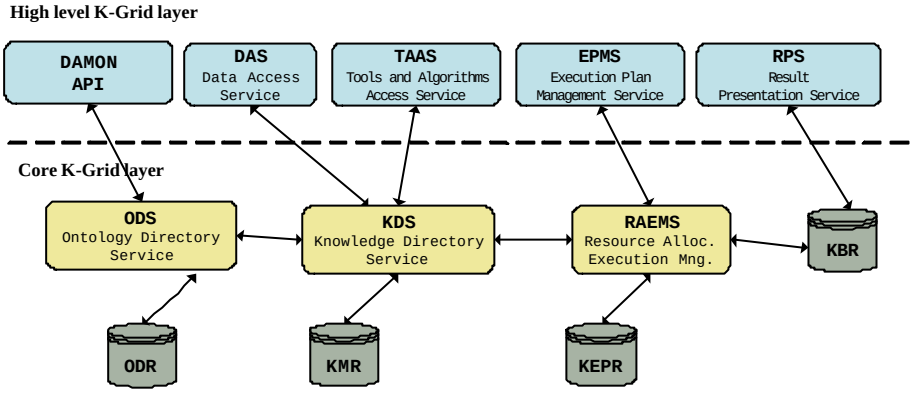

Fig. 14. Ontology services in the KNOWLEDGE GRID architecture

The *Ontology Directory Service* (ODS) is responsible for maintaining the ontological data and allows applications to query and manage them. The ontological data are represented by RDF Schema files and are stored in the *Ontological Data Repository* (ODR), whereas the metadata regarding the instance of each data mining resource (e.g. availability, location, and configuration) are stored into the.

The DAMON APIs are used to implement the DAMON Viewer that is responsible for the search and selection of data mining tools and software within the ontology. When a new data mining software is installed on a KNOWLEDGE GRID node, the owner has to "publish" it by using the KDS services, which store the metadata into the local KMR. If the software is already classified by the ontology, the KMR will simply send the new metadata URL to DAMON, otherwise, the owner is asked to update the ontology through DAMON.

The DAMON ontology can be thought as a knowledge base defined on the top of the current KNOWLEDGE GRID metadata. We have two metadata layers: at the top the domain ontology gives general information about resources, whereas specific information about installed software are maintained where resources resides. From an architec-

tural point of view the ontology is a central resource, whereas specific metadata are distributed ones.

The first step of the design process (see before) is split into two phases:

1. *Ontology-based resources selection*. Browsing and searching the ontology allows a user to locate the more appropriate tasks, methods, algorithms and finally data mining software to be used in a certain phase of the KDD process.
2. *KNOWLEDGE GRID metadata access*. The ontology gives the URLs of all instances of the selected resources available on the KNOWLEDGE GRID nodes, i.e. the URLs of the relevant metadata files stored in the KMRs.

As an example, a user logged on the KNOWLEDGE GRID node `g1.isi.cs.cnr.it` intends to perform a data mining application composed of two data mining steps, clustering and classification, on the data set `Unidb` stored on the KNOWLEDGE GRID node `g2.isi.cs.cnr.it`. The data set must be clustered using three different algorithms running in parallel on a copy of the data set. Clustering results must be analyzed by a classification algorithm that will be executed in parallel on three different nodes, generating three classification models of the same data set. By using DAMON, the user first searches the clustering algorithms by browsing or querying the ontology on the basis of some user requirements (computational complexity of the algorithm, attitude to solve the given problem or the method used to perform the data mining task), then he/she searches the clustering software implementing the algorithms and working on the data set `Unidb`, and finally locates the metadata URLs referring to the nodes `k1.deis.unical.it`, `k2.deis.unical.it` and `k3.deis.unical.it`, offering respectively the clustering software K-Means, Intelligent Miner, and AutoClass. Moreover, the user also finds the node `g2.isi.cs.cnr.it` that offers the C5.0 classifier. At this point the user can access specific information about the software by accessing the KMR on each identified node.

DAMON services are offered through the DAMON API, in particular they will be used to integrate the browse and search facilities of DAMON Viewer in the current Knowledge Grid authoring module.

6 Conclusions and Future Work

The use of ontologies to describe Grid resources will simplify and structure the systematic building of Grid applications through the composition and reuse of software components and the development of knowledge-based services and tools allowing a more effective resource management and scheduling.

The paper presented an ontology for the Data Mining domain whose main goal is to simplify the development of distributed knowledge discovery applications on the Grid, offering to a domain expert a reference model for the different kind of data mining tasks, methodologies and software available to solve a given problem, and helping him/her in finding the most appropriate solution.

A tool implementing the basic ontology operations, such as browsing and searching, and a practical application of the DAMON ontology have been presented, showing how the design of applications would be enhanced in the KNOWLEDGE GRID, a grid-based environment for distributed data mining applications.

Future work will regard the full implementation and integration of the DAMON-MAP tool into the KNOWLEDGE GRID, and its extension in different application domains. In fact, we think that the proposed approach could be generalized to become a core method in general purpose Grid-based Problem Solving Environments.

Acknowledgements

This work has been partially supported by Project "FIRB GRID.IT" funded by MIUR.

References

1. F. Lopez, Overview of Methodologies for building ontologies, Proc. of IJCAI-99, workshop KRR5, Sweden, 1999.
2. C. Wroe, R. Stevens, C. Goble, A. Roberts, M. Greenwood, A suite of DAML+OIL onthologies to describe bioinformatics web services and data, in International Journal of Co-operativeInformation System, March 2003 (in press) available from http://www.mygrid.org.uk.
3. S. Bechhofer, C. Goble, Towards annotation using DAML+OIL. K-CAP 2001workshop on Knowledge markup and Semantic Annotation, Victoria B. C, October 2001.
4. B. McBride, Jena: a semantic web toolkit, Internet Computing, Dec. 2002.
5. Sean Bechhofer, Ian. Horrocks, Carole Goble, Robert 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
6. T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993. Available on line at http://ksl-web.stanford.edu/KSL_Abstracts/KSL-92-71.html.
7. Foster I. and Kesselman C. (eds.), The Grid: Blueprint for a Future Computing Infrastructure, Morgan Kaufmann Publishers, 1999
8. de Roure D., Jennings, N. R. and Shadbolt, N. (2003) "The Evolution of the Grid". Concurrency and Computation: Practice and Experience. (2003)
9. de Roure, D. Jennings, N. R., Shadbolt, N. "The Semantic Grid: A future e-Science infrastructure". Concurrency and Computation: Practice and Experience, 2003.
10. I. Foster, C. Kesselman, J. Nick, S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.
11. D. Talia, "The Open Grid Services Architecture: Where the Grid Meets the Web", IEEE Internet Computing, Vol. 6, No. 6, pp. 67-71, 2002.

12. F. Berman. “From TeraGrid to Knowledge Grid”. CACM, Vol. 44, N. 11, pp. 27-28, 2001.
13. W. E. Johnston, “Computational and Data Grids in Large-Scale Science and Engineering”. Future Generation Computer Systems, Vol. 18, N. 8, pp. 1085-1100, 2002.
14. Cannataro M. and D. Talia, “KNOWLEDGE GRID An Architecture for Distributed Knowledge Discovery”, CACM, Vol. 46, No. 1, pp. 89-93, January 2003.
15. Cannataro M., Talia D. “Towards the Next-Generation Grid: A Pervasive Environment for Knowledge-Based Computing”, 4th IEEE International Conference on Information Technology: Coding and Computing (ITCC2003), April 28-30, 2002, Las Vegas, IEEE Computer Society Press, 2003
16. RDF, org.w3c.rdf
17. DQL, http://www.daml.org/dql/ DQL
18. RQL: A Declarative Query Language for RDF, G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, Michel Scholl, The Eleventh International World Wide Web Conference (WWW'02), Honolulu, Hawaii, USA, May 7-11, 2002.
19. Guarino N., Giaretta P., “Ontologies and knowledge bases, towards a terminological clarification, Toward Very Large Knowledge Bases, pp. 25-32, IOS Press, 1995.
20. Chandrasekaran, B.; Johnson, T. R.; Benjamins, V. R. “Ontologies: what are they? why do we need, them?”. IEEE Intelligent Systems and Their Applications. 14(1). Special Issue on Ontologies. Pages, 20-26. 1999.
21. Neches, R.; Fikes, R.E.; Finin, T.; Gruber, T.R.; Senator, T.; Swartout, W.R. “Enabling technology for knowledge sharing”. AI Magazine. 12(3)::36-56- 1991.
22. DAML, http://www.daml.org/
23. J. Han, M. Kamber, Data Mining, Concepts and Techniques, Morgan Kaufmann, 2000.
24. Oscar Corcho, Mariano Fernández-López, Asunción Gómez Pérez, OntoWeb: Technical Roadmap, www.ontoweb.org.
25. DAML API, http://grcinet/grci.com/
26. GLOBUS, www.globus.org.
27. Ontology (Special Issue on), CACM, Vol. 45, N. 2, 2002.
28. M. Cannataro, A. Congiusta, C. Mastroianni, A. Pugliese, D. Talia, Paolo Trunfio, Grid-Based Data Mining and Knowledge Discovery, in N. Zhong and J. Liu (Eds.), Handbook of Intelligent Information Technology, Volume 90, IOS Press, 2003.
29. Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, Ramasamy Uthurusamy: Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press 1996