Managing Heterogeneous Resources in Data Mining Applications on Grids Using XML-Based Metadata

Carlo Mastroianni[1], Domenico Talia[2], Paolo Trunfio[2]

[1]ICAR-CNR
Via P. Bucci, cubo 41-c, 87036 Rende (CS), Italy
mastroianni@icar.cnr.it

[2]DEIS
Università della Calabria, Via P. Bucci, cubo 41-c, 87036 Rende (CS), Italy
talia@deis.unical.it trunfio@deis.unical.it

Abstract

The Grid supports the sharing and coordinated use of resources in dynamic heterogeneous distributed environments. The effective use of a Grid requires the definition of an approach to manage the heterogeneity of the involved resources that can include computers, data, network facilities and software tools provided by different organizations. This issue get more importance when complex applications, such as data-intensive simulations and data mining applications, executed on a Grid. This paper is concerned with heterogeneous resource management in Grid-based data mining applications. It discusses how resources are represented and managed in the KNOWLEDGE GRID and how XML-based metadata are used to describe data mining tools, data sources, mining models and execution plans, and how those metadata are used for the design and execution of distributed data mining applications on Grids.

1. Introduction

The Grid infrastructure supports the sharing and coordinated use of resources in dynamic geographically distributed environments. The effective use of a Grid requires the definition of an approach to manage the heterogeneity of the involved resources that can include computers, data, network facilities and software tools provided by different organizations. Heterogeneity arises mainly from the large variety of resources within each category. For instance, software can run only on some particular host machines whereas data can be extracted from different data management systems such as relational databases, semi-structured databases, plain files, etc.

The management of such heterogeneous resources requires the use of metadata, whose purpose is to provide information about the features of resources and their effective use. A Grid user needs to know which resources are available, where resources can be found, how resources can be accessed and when resources are available. Metadata can provide answers about involved computing resources such as data repositories (e.g., databases, file systems, web sites), machines, networks, programs, documents, user agents, etc. Therefore, metadata can represent a key element to effective resource discovery and utilization on the Grid.

The role of metadata for resource management on Grids is more and more important as Grid applications are becoming more and more complex. Thus, Grids need to use mechanisms and models that define rich metadata schemas able to able represent the variety of involved resources.

This paper is concerned with heterogeneous resource management in Grid-based data mining applications. That is, it addresses the problems of locating and allocating computational, data and information resources, and other activities required to use data mining resources in a knowledge discovery process on Grids. In particular, the paper discusses an XML-based approach for managing heterogeneous resources in the KNOWLEDGE GRID environment [1].

The KNOWLEDGE GRID architecture uses the basic Grid services and defines a set of additional layers to implement the services of distributed knowledge discovery on world wide connected computers where each node can be a sequential or a parallel machine.

The KNOWLEDGE GRID architecture (see Figure 1) is designed on top of mechanisms provided by Grid environments such as Globus [2]. The KNOWLEDGE GRID uses the basic Grid services such as communication, authentication, information, and resource management to build more specific parallel and distributed knowledge discovery (PDKD) tools and services.

The KNOWLEDGE GRID services are organized into two layers: the *Core K-Grid* layer, which is built on top of generic Grid services, and the *High level K-Grid* layer,

which is implemented over the core layer.

The Core K-Grid layer comprises two basic services: the *Knowledge Directory Service* (*KDS*) and the *Resources Allocation and Execution Management Service* (*RAEMS*). The KDS manages the metadata describing the characteristics of relevant objects for PDKD applications, such as data sources, data mining software, results of computations, data and results manipulation tools, execution plans, etc. The information managed by the KDS is stored into three ad hoc repositories: the metadata describing features of data, software and tools, coded in XML documents, are stored in a *Knowledge Metadata Repository* (*KMR*), the information about the knowledge discovered after a PDKD computation is stored in a *Knowledge Base Repository* (*KBR*), whereas the *Knowledge Execution Plan Repository* (*KEPR*) stores the execution plans describing PDKD applications over the grid. The goal of RAEMS is to find a mapping between an execution plan and available resources on the grid, satisfying user, data and algorithms requirements and constraints.

The High level K-Grid layer comprises the services used to build and execute PDKD computations over the Grid. The *Data Access Service* (*DAS*) is used for the search, selection, extraction, transformation and delivery of data to be mined. The *Tools and Algorithms Access Service* (*TAAS*) is responsible for search, selection, and download of data mining tools and algorithms. The *Execution Plan Management Service* (*EPMS*) is used to generate a set of different possible execution plans, starting from the data and the programs selected by the user. Execution plans are stored in the KEPR to allow the implementation of iterative knowledge discovery processes, e.g., periodical analysis of the same data sources varying in time. The *Results Presentation Service* (*RPS*) specifies how to generate, present and visualize the PDKD results (rules, associations, models, classification, etc.), and offers methods to store in different formats these results in the KBR.

By using services, tools, and repositories provided by the two layers of the KNOWLEDGE GRID, a user can search and identify data sources, data mining tools, and computational resources. Then she/he can combine all these components to build a distributed/parallel data mining application that can be executed on a Grid.

In the next sections we discuss how resources are represented and managed in the KNOWLEDGE GRID; how XML-based metadata are used to define data mining tools, data sources, mining models and execution plans, and how those metadata are used in the design and execution of distributed data mining applications on Grids. Section 2 discusses the management of resources. Section 3 describes resource metadata representation. Section 4 discusses the execution plan representation. Section 5 presents an outline of related work and Section 6 concludes the paper.

2. Management of the Resources in the Knowledge Grid

A number of approaches to represent and manage metadata has been investigated in Grid environments such as Globus [2], DICE [3], and the NASA's Information Power Grid [4]. In particular, in the Globus Toolkit the *Monitoring and Discovery Service* (*MDS*) provides information about the status of the system components [5]. The MDS uses the *Lightweight Directory Access Protocol* (*LDAP*) [6] as a uniform interface to such information. MDS includes a configurable information provider called *Grid Resource Information Service* (*GRIS*) and a configurable aggregate directory service called *Grid Index Information Service* (*GIIS*). A GRIS can answer queries about the resources of a particular Grid node. Examples of information provided by this service include host identity (e.g., operating systems and versions), as well as more dynamic information such as CPU and memory availability. A GIIS combines the information provided by a set of GRIS services managed by an organization, giving a coherent system image that can be explored or searched by Grid applications.

The KNOWLEDGE GRID manages resources involved in a typical distributed data mining computation such as:

- Computational resources (computers, storage devices, etc.).
- Data to be mined, such as databases, plain files, semi-structured documents and other structured or unstructured data (data sources).
- Tools and algorithms used to extract, filter and manipulate data (data management tools).
- Tools and algorithms used to mine data, that is data mining tools available on the Grid nodes.
- Knowledge obtained as result of the mining process, i.e. learned models and discovered patterns.
- Tools and algorithms used to visualize, store and manipulate discovered models.

This large set of different resources that in some cases require a complex description, motivated the definition of a metadata model that extends the basic Globus model.

The basic objectives that guided us through the definition of the resource metadata are the following:

- Metadata should document in a simple and human-readable fashion the features of a data mining application.
- Metadata should allow the effective search of resources.
- Metadata should provide an efficient way to access resources.
- Metadata should be used by software tools that support a user in building a KNOWLEDGE GRID computation such as VEGA [7], a visual toolset for designing and executing data mining applications over

the KNOWLEDGE GRID.

The current KNOWLEDGE GRID implementation uses the Globus MDS, and therefore the LDAP protocol, to publish, discover, and manage information about the generic resources of the underlying grid (e.g., cpu performance, memory size, etc.). As mentioned before, the complexity of the information associated to more specific KNOWLEDGE GRID resources (data sources, mining algorithms, models) has led us to design a different model to represent and manage the corresponding metadata.

For managing data mining resources on Grids, we adopted the *eXtensible Markup Language* (*XML*), that provides a set of functionalities and capabilities that are making it a common emerging model for describing data structure and data set frameworks:

- XML provides a way to define infrastructure independent representations for information.
- XML allows a user to define complex data structures: for example the XML Schema formalism [8] provides a means for defining a strong control on simple and complex data types in XML documents.
- XML allows the use of powerful query languages: for instance the XML Query [9] provides SQL-like query facilities to extract data from real and virtual documents on the Web.
- It is easy to map XML documents into data structures of an object-oriented programming language: for example the Xerces library [10] performs the parsing of an XML document in a Java or C++ environment.

On the basis of these features, we decided to represent metadata by XML documents according to a set of XML schemas defined for the different classes of resources, as we discuss in the next section.

It is worth noting that by using XML for metadata definition we may benefit from a standard language that makes our model flexible and extensible. Furthermore, the resulting metadata model could be used to describe other advanced Grid applications.

In the KNOWLEDGE GRID, metadata are accessed and managed by means of a set of services. In particular, the KNOWLEDGE GRID architecture defines, as mentioned above, the KDS, that maintains the metadata documents and allows applications to query and manage them. The information managed by the KDS is stored into three ad hoc repositories:

1. the *Knowledge Metadata Repository* (*KMR*) that stores the metadata describing features of data, software, and tools;
2. the *Knowledge Base Repository* (*KBR*) that stores information about the discovered models, and
3. the *Knowledge Execution Plan Repository* (*KEPR*) which stores the execution plans describing distributed data mining applications over the Grid.

The KNOWLEDGE GRID DAS and TAAS services make use of the KDS for, respectively:

- search, selection (Data search services), extraction, transformation and delivery (Data extraction services) of data to be mined and
- search, selection and downloading of data mining tools and algorithms.

Figure 1 shows main metadata flows among KNOWLEDGE GRID services and repositories. The high level DAS and TAAS services use the core level KDS service to manage metadata about algorithm and data sources; in turn, the KDS interacts with the KMR repository to access and store such metadata. The EPMS service manages execution plan metadata, which are accessed through the core level RAEMS service, and stored in the KEPR database.

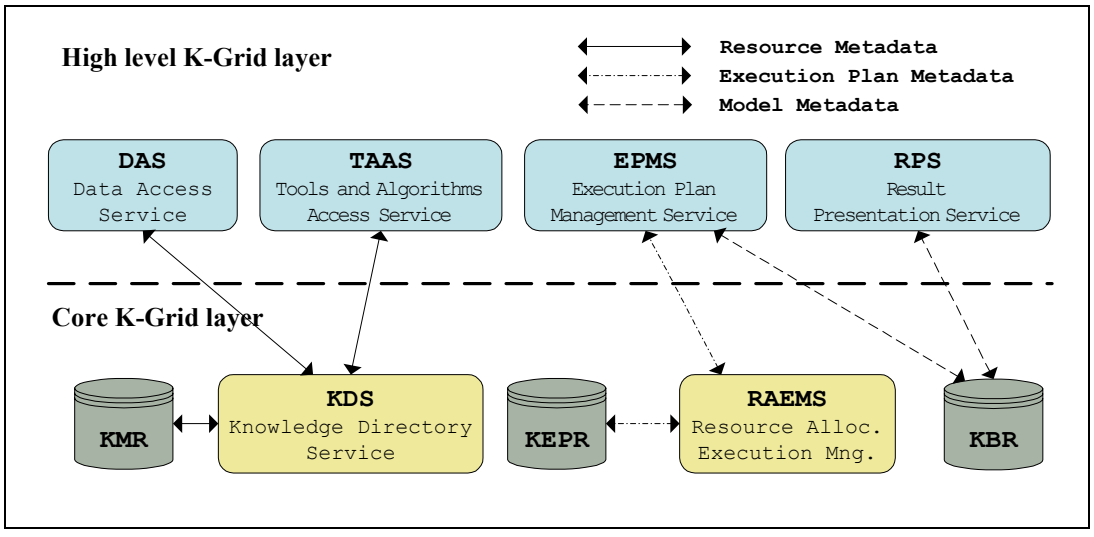

Figure 1. The KNOWLEDGE GRID architecture.

Furthermore, execution results and related model metadata are stored in the KBR repository, and processed by the RPS service.

Metadata management process is a key aspect in the developing of data mining applications over the KNOWLEDGE GRID. A typical life cycle of metadata consists of the following steps:

1. Resource metadata are published on the KMRs of the corresponding nodes.
2. The user specifies the features of the resources she/he needs to design a data mining application.
3. The DAS and TAAS services search the KMRs of the KNOWLEDGE GRID nodes for the requested resources;
4. Metadata describing the resources of interest are delivered by such services to the requesting user.
5. Metadata related to software, data and operations are combined into an execution plan (see Section 4) to design a complete data mining application.
6. After application execution, results are stored into the KBR; new and/or modified resources' metadata are published in the respective KMRs for future use.

3. Resource Metadata Representation

As stated before, our goal is to classify all the heterogeneous resources involved in a data mining application on the Grid, by defining the corresponding metadata.

The first step in designing the metadata is the categorization of the resources. In the current implementation of the KNOWLEDGE GRID, we focused on the definition of metadata related to data sources, data mining tools, and discovered knowledge. In the following, for each of those resource types, we present the metadata structure, and report a sample metadata document extracted from a real data mining application based on the AutoClass clustering tool [11].

3.1 Data Mining Software

The categorization of the data mining software has been made on the basis of the following classification parameters [12]:

- the kind of data sources on which the software works on;
- the kind of knowledge that is to be discovered by the software;
- the type of techniques that the software uses in the data mining process;
- the driving method, i.e. the mining process can be autonomous, driven by data or queries, or driven by the user (interactive).

Table 1 summarizes a number of possible values for each classification parameter. This table is mapped on a XML Schema which defines the format and the syntax of the XML file that will be used to describe the features of a generic data mining software. The second column of Table 1 reports the XML elements that correspond to the classification parameters.

As an example, Figure 2 reports the XML metadata related to the data mining software AutoClass.

The XML file is composed of two parts. The first part is the software `Description`, the second one is the software `Usage`. The `Description` section specifies one (or more) of the possible values, reported in Table 1, for each classification parameter. The `Usage` section contains all the information that can be used by a client to access and use the software. This section is composed of a set of subsections, among which `Syntax`, `Hostname`, `ManualPath`, and `DocumentationURL`. The `Syntax` subsection describes the format of the command that the client should use to invoke the software. This subsection is defined as a tree, where each node is an `Arg` element, and the root is the name of the software itself. The root children specify the arguments that should follow the software name in the software invocation, and these arguments can in turn have children, i.e. sub-arguments, and so on. Each `Arg` element has the following attributes: the `description` attribute is a textual description of the argument; the `type` attribute specifies if the argument is `optional`, `required`, or `alternative`. In the last case all the sibling arguments should have the same value for this attribute, meaning that only one of the siblings should be used in the software invocation. Finally, the `value` attribute (optional) specifies the fixed value of the argument. If the `value` attribute is omitted, the value is to be provided by the client. In the example shown in Figure 2, in the AutoClass execution command, the executable name should be followed by the `-search` argument to ask for a classification, or by the `-reports` argument, to obtain the model file. If the `-search` argument is chosen, it should be followed by four sub-arguments, all `required`.

Therefore, AutoClass can be invoked with the command:

```
/usr/autoclass/autoclass -search aFile.db2 \
    aFile.hd2 aFile.model aFile.s-params
```

3.2 Data Sources

Data sources are the input on which the data mining algorithms work to extract new knowledge [12]. They can be provided by relational databases, plain files, and other structured and semi-structured documents. In spite of the wide variety of the possible data source types, we aim to define a common structure of data source metadata in order to standardize the access and search operations on such resources.

Classification parameter	XML tag	Possible values
Kind of data sources to work on	<KindOfData>	relational database, transaction database, object-oriented database, deductive database, spatial database, temporal database, multimedia database, heterogeneous database, active database, legacy database, semi-structured data, flat file.
Kind of knowledge to be mined	<KindOfKnowledge>	association rules, clusters, characteristic rules, classification rules, sequence discovery, discriminant rules, evolution analysis, deviation analysis, outlier detection, regression.
Kind of techniques to be utilized	<KindOfTechnique>	statistics, decision trees, neural networks, genetic algorithms, Apriori, fuzzy logic, SVD, bayesian networks, nearest neighbors…
Driving method	<DrivingMethod>	autonomous knowledge miner, data-driven miner, query-driven miner, interactive data miner.

Table 1. Classification of data mining software.

```
<DataMiningSoftware name="AutoClass">
  <Description>
    <KindOfData>flat file</KindOfData>
    <KindOfKnowledge>clusters</KindOfKnowledge>
    <KindOfTecnique>statistics</KindOfTecnique>
    <DrivingMethod>autonomous knowledge miner</DrivingMethod>
  </Description>
  <Usage>
    ...
    <Syntax>
      <Arg description="executable" type="required" value="/usr/autoclass/autoclass">
        <Arg description="make a classification" type="alternative" value="-search">
          <Arg description="a .db2 file" type="required"/>
          <Arg description="a .hd2 file" type="required"/>
          <Arg description="a .model file" type="required"/>
          <Arg description="a .s-params file" type="required"/>
        </Arg>
        <Arg description="create a report" type="alternative" value="-reports">
          <Arg description="a .results-bin file" type="required"/>
          ...
        </Arg>
        ...
      </Arg>
    </Syntax>
    <Hostname>icarus.cs.icar.cnr.it</Hostname>
    <ManualPath>/usr/autoclass/read-me.text</ManualPath>
    <DocumentationURL>http://ic-www.arc.nasa.gov/ic/projects/...</DocumentationURL>
    ...
  </Usage>
</DataMiningSoftware>
```

Figure 2. An extract from an XML metadata sample for the AutoClass software.

The common structure of metadata is composed of two parts:

- an `Access` section that includes information for retrieving the data source;
- a `Structure` section that provides information about the data source logical and/or physical structure.

As an example, Figure 3 shows an XML metadata document for a flat file that can be used as an input by the AutoClass software.

The `Access` section includes file system information, e.g. the `Location` and the `Size` of the file, etc. The `Structure` section includes two subsections, `Format` and `Attributes`.

The `Format` subsection contains information about the physical structure of the flat file, e.g. the strings that are used to separate the records and the attributes within a record.

The `Attributes` subsection contains information about

the logical structure, i.e. it lists the table attributes and provides the relative specifications (such as the `name` of the `Attribute`, its `type`, etc.).

If the data source is a relational database, the high-level XML metadata format is the same (with the `Access` and `Structure` sections), but modifications can be operated on more specific details. For example, the `Format` subsection is no more needed, since the physical formatting is managed by the database system. Furthermore, new subsections should be defined; for instance, in the `Access` section, information should be provided for the connection to the database (e.g., the ODBC specifications).

3.3 Data Mining Models

The knowledge discovered through the data mining process is represented by "data mining models". Whereas till today no common models have been defined for the definition of the data mining resources discussed before, a standard model, called *Predictive Model Markup Language* (*PMML*) has been defined to describe data mining results. PMML is an XML language which provides a vendor-independent method for defining data mining models [13]. The PMML provides a *Document Type Definition* (*DTD*) to describe different kinds of models such as classification rules and association rules. We use it to define data mining models in the KNOWLEDGE GRID. As an example, in Figure 4 we show an extract from a PMML document that represents the clustering model produced by AutoClass from the dataset whose metadata have been reported in Figure 3. In this example, AutoClass performs a clustering on records concerning car imports in 1985.

The `MiningSchema` element of the model reported in Figure 4 points out that the clustering is based on three record attributes: *make*, *num-of-doors* and *body-style*. Moreover, two clusters (out of 12) are described, the former composed of 28 and the latter composed of 4 records.

Notice that each cluster record can be reconstructed by taking the values reported in the same position in each clustering field. For example, *<bmw two sedan>* is the first record belonging to *Cluster 1*. The portion of the PMML DTD that we used for this clustering model is available at [14].

4. Execution Plan Representation

A distributed data mining computation is a process composed of several steps which are executed sequentially or in parallel. In the KNOWLEDGE GRID framework, the management of complex data mining processes has been carried out by the definition of an *execution plan*. An execution plan is a graph that describes the interaction and data flow between data sources, data mining tools, visualization tools and output models.

```
<FlatFile>
  <Access>
    <Location>/usr/share/imports-85c.db2</Location>
    <Size>26756</Size>
    ...
  </Access>
  <Structure>
    <Format>
      <AttributeSeparatorString>,</AttributeSeparatorString>
      <RecordSeparatorString>#</RecordSeparatorString>
      <UnknownTokenString>?</UnknownTokenString>
      ...
    </Format>
    <Attributes>
      <Attribute name="symboling" type="discrete">
        <SubType>nominal</SubType>
        <Parameter>range 7</Parameter>
      </Attribute>
      <Attribute name="normalized-loses" type="real">
        <SubType>scalar</SubType>
        <Parameter>zero_point 0.0</Parameter>
        <Parameter>rel_error 0.01</Parameter>
      </Attribute>
      ...
    </Attributes>
  </Structure>
</FlatFile>
```

Figure 3. An extract from an XML metadata sample for a flat file.

```
<PMML version="2.0">
  ...
    <ClusteringModel modelName="Clustering on imports-85c"
                     modelClass="distributionBased" numberOfClusters="12">
       <MiningSchema>
          <MiningField name="make"/>
          <MiningField name="num-of-doors"/>
          <MiningField name="body-style"/>
       </MiningSchema>
       ...
       <Cluster name="Cluster 1">
         <Partition name="Partition 1">
           <PartitionFieldStats field="make">
             <Array n="28" type="string">bmw bmw jaguar nissan ...</Array>
           </PartitionFieldStats>
           <PartitionFieldStats field="num-of-doors">
             <Array n="28" type="string">two four four four ...</Array>
           </PartitionFieldStats>
           <PartitionFieldStats field="body-style">
             <Array n="28" type="string">sedan sedan sedan wagon ...</Array>
           </PartitionFieldStats>
         </Partition>
       </Cluster>
       <Cluster name="Cluster 2">
         <Partition name="Partition 2">
           <PartitionFieldStats field="make">
             <Array n="4" type="string">chevrolet chevrolet chevrolet dodge</Array>
           </PartitionFieldStats>
           ...
         </Partition>
       </Cluster>
       ...
     </ClusteringModel>
</PMML>
```

Figure 4. An extract from a PMML model file.

Starting from the XML representation of the data mining resources, an execution plan defines the high-level logical structure of a data mining process. The KNOWLEDGE GRID provides a visual environment, called VEGA [7], that allows a user to build an execution plan in a semi-automatic way.

An execution plan may contain *concrete resources* and *abstract resources*. A concrete resource is completely specified by its metadata that have been previously retrieved from remote and local KMRs. In an abstract resource metadata, some features are expressed as constraints and not as well known values.

For instance, whereas the metadata described in section 3 describes a concrete software Autoclass available on a given node, the metadata document shown below describes an (abstract) data mining software able to perform a clustering computation on flat files.

```
<DataMiningSoftware name="genericSoftware">
  <Description>
    <KindOfData>flat file</KindOfData>
    <KindOfKnowledge>clusters</KindOfKnowledge>
  </Description>
</DataMiningSoftware>
```

An abstract resource can be instantiated into an existing concrete resource whose metadata match the specified constraints.

An execution plan that contains at least one abstract resource is an *abstract execution plan*, whereas an execution plan containing only concrete resources is referred to as an *instantiated execution plan*. Such distinction is made to take into account the dynamic nature of a Grid environment, in which resources fail and become available, data gets deleted, software gets updated, etc. In general, a user builds an abstract execution plan, and the EPMS service attempts to transform it into an instantiated execution plan, by substituting abstract resources into concrete resources. Such action is performed by a scheduler that allows the generation of the optimal execution plan.

From an abstract execution plan, different instantiated execution plans could be generated, depending on the resources that are available on the KNOWLEDGE GRID in different times.

Figure 4 shows an extract from a sample (instantiated) execution plan. An execution plan gives a list of tasks and task links, which are specified using respectively the XML tags `Task` and `TaskLink`. The `label` attribute of the `Task` element identifies one basic task in the execution plan, and it is used in linking various basic tasks to form the overall task flow. Each `Task` element contains a task-specific sub-element, which indicates the parameters of

the particular represented task. For instance, the task identified by the `ws1_dt4` label contains a `DataTransfer` element, indicating that it is a data transfer task. The `DataTransfer` element specifies `Protocol`, `Source` and `Destination` of the data transfer. The `href` attributes of such elements specify respectively the location of metadata about protocol, source and destination objects.

A `TaskLink` element represents a relation between two tasks in an execution plan. For instance, the shown `TaskLink` indicates that the task flow proceeds from the task `ws1_dt4` to the task `ws2_c1` (that represents an `Execution` step), as specified by its `from` and `to` attributes.

An instantiated execution plan will be translated into the language of a specific Grid resource broker for its execution. The KNOWLEDGE GRID implementation uses the *Globus Resource Allocation Manager* (*GRAM*) [15] whose script language is the *Resource Specification Language* (*RSL*) [16]. RSL is a structured language by which a user can program the execution of a Grid application.

Figure 6 shows the RSL script corresponding to the sample XML execution plan of Figure 5. Such RSL script comprises two *resource descriptions*: the former refers to the `DataTransfer` task labeled as `ws1_dt4`, the latter corresponds to the `Execution` task labeled as `ws2_c1` in the instantiated execution plan. A typical resource description is composed of several attribute-value relationships in a conjunction.

For instance the first description in this RSL script specifies that a data transfer is to be performed using the `globus-url-copy` executable, located in the `minos.cs.icar.cnr.it` grid node, to copy the `imports-85c.db2` dataset from `minos.cs.icar.cnr.it` to `icarus.cs.icar.cnr.it`.

5. Related work

Several systems that support distributed data mining and data transformation processes have been proposed.

A few of those systems operate on the Grid, whereas most of the proposed systems work on clusters of computers or over the Internet. Here we shortly list their basic features and metadata management modalities.

Discovery Net [17] provides an architecture for building and managing KDD processes on the Grid. Like in the KNOWLEDGE GRID, metadata are crucial in the Discovery Net infrastructure: an XML language called *Discovery Process Markup Language* (*DPML*) is used to describe simple and compound applications, while data resources are characterized through the definition of "data types" stored in a "Meta-Information Server". However, the lack of distinction between logical and concrete resources can limit the efficient management of dynamics in the Grid environment.

Papyrus [18] is a distributed data mining system developed for clusters and superclusters of workstations as composed four software layers: data management, data mining, predictive modeling, and agent or Bast. Papyrus is based on mobile agents implemented using Java aglets. (*DSML*) for clusters and data information.

```
<ExecutionPlan>
 ...
 <Task ep:label="ws1_dt4">
  <DataTransfer>
   <Protocol ep:href="minos../GridFTP.xml"
             ep:title="GridFTP on minos.cs.icar.cnr.it"/>
   <Source ep:href="minos../imports-85c_db2.xml"
           ep:title="imports-85c.db2 on minos.cs.icar.cnr.it"/>
   <Destination ep:href="icarus../imports-85c_db2.xml"
                ep:title="imports-85c.db2 on icarus.cs.icar.cnr.it"/>
  </DataTransfer>
 </Task>
 ...
 <Task ep:label="ws2_c1">
  <Execution>
   <Program ep:href="icarus../autoclass3-3-3.xml"
            ep:title="autoclass on icarus.cs.icar.cnr.it"/>
   <Input ep:href="icarus../imports-85c_db2.xml"
          ep:title="imports-85c.db2 on icarus.cs.icar.cnr.it"/>
   ...
   <Output ep:href="icarus../classes.xml"
           ep:title="Classes on icarus.cs.icar.cnr.it"/>
  </Execution>
 </Task>
 ...
 <TaskLink ep:from="ws1_dt4" ep:to="ws2_c1"/>
 ...
```

Figure 5. An extract from an execution plan.

```
+
...
(&(resourceManagerContact=minos.cs.icar.cnr.it)
  (subjobStartType=strict-barrier)
  (label=ws1_dt4)
  (executable=$(GLOBUS_LOCATION)/bin/globus-url-copy)
  (arguments=-vb -notpt gsiftp://minos.cs.icar.cnr.it/.../imports-85c.db2
             gsiftp://icarus.cs.icar.cnr.it/.../imports-85c.db2
   )
)

...

(&(resourceManagerContact=icarus.cs.icar.cnr.it)
  (subjobStartType=strict-barrier)
  (label=ws2_c1)
  (executable=.../autoclass)
  (arguments=-search .../imports-85c.db2  .../imports-85c.hd2 .../imports-85c.model
             ...
   )
)
...
```

Figure 6. An extract from an RSL script.

This distributed data mining system uses PMML for predictive models and an XML language called *Data Space Markup Language* Another distributed data mining suite based on Java is PaDDMAS [19], a component-based tool set that integrates predeveloped or custom packages (that can be sequential or parallel) using a dataflow approach. Each system component is wrapped as a Java or CORBA object with its interface specified in XML. The XML definition may be used in PaDDMAS to automatically derive help on a particular component, to check the suitability of a component for analyzing a particular data set, the type of platforms that may support the component, etc.

Concerning the use of metadata for application description, the Chimera system [20] proposes a language, called *Virtual Data Language* (*VDL*), for defining data transformation processes. A VDL document is structured in a way similar to an execution plan defined in the KNOWLEDGE GRID.

Besides the systems we discussed above, other interesting distributed data mining systems have been developed. In such systems metadata management appears to be not a central issue, because they focus on the use of dedicated platforms or make use of homogeneous software components.

Among such systems, JAM [21] is an agent-based distributed data mining system that has been developed to mine data stored in different sites for building so called meta-models as a combination of several models learned at the different sites where data are stored. JAM uses Java applets to move data mining agents to remote sites.

A sort of metalearning, called collective data mining, is implemented also in the BODHI system [22]. BODHI is another agent-based distributed data mining system implemented in Java.

Metadata management models similar to the approaches discussed here are also deployed in other computer science areas such as problem solving environments (PSEs). Examples of significant PSEs that use XML-based metadata models for representation of heterogeneous resources are WebFlow and the Common Portal Application [23] .

6. Conclusions

This paper discussed the management of heterogeneous resources in Grid-based data mining applications. We motivated and presented the use of an XML-based approach to represent metadata in the KNOWLEDGE GRID environment. We introduced and discussed the metadata structure for the main classes of resources involved in a data mining process (data mining software, data sources, mining results, and execution plans). For each resource class we reported a sample metadata document extracted from a real data mining application. We are currently using the metadata model for the implementation of distributed data mining applications running on the KNOWLEDGE GRID.

As a result of our work we can conclude about the importance of the usage of metadata for the implementation of resource representation, search, and discovery services in heterogeneous computing environments such as Grids and meta-computing systems. In particular, metadata models are major players where complex applications, such as knowledge discovery processes or scientific simulations, must be developed in such environments.

Acknowledgements

This work has been partially funded by the project ”MIUR Fondo Speciale SP3: GRID COMPUTING: Tecnologie abilitanti e applicazioni per eScience”.

References

[1] M. Cannataro, D. Talia, P. Trunfio, *KNOWLEDGE GRID: High Performance Knowledge Discovery Services on the Grid*, Proceedings GRID 2001, LNCS, pp.38-50, Springer-Verlag, 2001.

[2] I. Foster, C. Kesselman, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, S. Tuecke, Intl. J. Supercomputer Applications, 15(3), 2001.

[3] Reagan W. Moore, *Persistent Archives for Data Collections SDSC*, UC San Diego SDSC TR-1999-2, October 1999.

[4] W. Johnston, *NASA's Information Power Grid: Production Grid Experience with Distributed Computing and Data Management*, In Second Global Grid Forum Workshop (GGF2), Washington, D.C., 2001.

[5] The Globus Project. *The Monitoring and Discovery Service*. http://www.globus.org/mds.

[6] RFC 2251 - Lightweight Directory Access Protocol (v3).

[7] M. Cannataro, A. Congiusta, D. Talia and P. Trunfio, *A Data Mining Toolset for Distributed High-Performance Platforms*, Proceedings 3rd Int. Conference Data Mining 2002, Bologna, WIT Press, pp. 41-50, September 2002.

[8] XML Schema. http://www.w3.org/XML/Schema.

[9] XML Query. http://www.w3.org/XML/Query.

[10] Xerces library. http://xml.apache.org.

[11] P. Cheeseman and J. Stutz, *Bayesian Classification (AutoClass): Theory and Results*, Advances in Knowledge Discovery and Data Mining, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy, (Eds.) AAAI Press/MIT Press, pp. 61-83, 1996.

[12] M.S. Chen, J. Han, and P.S. Yu, *Data Mining: An Overview from a Database Perspective*, IEEE Transactions on Knowledge and Data Engineering, 8(6): 866-883, 1996.

[13] R. L. Grossman, M. F. Hornick, G. Meyer, *Data Mining Standard Initiatives*, Communications of the ACM, Vol. 45, N. 8, August 2002.

[14] PMML 2.0 - DTD for Clustering Models http://www.dmg.org/pmmlspecs_v2/ClusteringModel.htm.

[15] The Globus Project. *The Globus Resource Allocation Manager*. http://www.globus.org/gram.

[16] The Globus Project. *The Globus Resource Specification Language*. http://www.globus.org/gram/rsl_spec1.html.

[17] V. Curcin, M. Ghanem, Y. Guo, M. Kohler, A. Rowe, J. Syed, P. Wendel. *Discovery Net: Towards a Grid of Knowledge Discovery.* ACM KDD 2002.

[18] R. Grossman, S. Bailey, S. Kasif, D. Mon, A. Ramu and B. Malhi, *The preliminary design of papyrus: a system for high performance, distributed data mining over clusters, meta-clusters and super-clusters*, International KDD'98 Conference, 1998, pp. 37-43.

[19] O.F. Rana, D.W. Walker, M. Li, S. Lynden and M. Ward, *PaDDMAS: parallel and distributed data mining application suite*, Proc. International Parallel and Distributed Processing Symposium (IPDPS/SPDP), IEEE Computer Society Press, 2000, pp. 387-392.

[20] I. Foster, J. Vöckler, M. Wilde, Y. Zhao, *Chimera: a Virtual Data System for Representing, Querying, and Automating Data Derivation*, SSDBM 2002, pp. 37-46.

[21] S.J. Stolfo, A.L. Prodromidis, S. Tselepis, W. Lee, D.W. Fan, P.K. Chan, *JAM: Java agents for meta-learning over distributed databases*, International KDD'97 Conference, 1997, pp. 74-81.

[22] H. Kargupta, B. Park, D. Hershberger and E. Johnson, *Collective data mining: a new perspective toward distributed data mining*, In H. Kargupta and P. Chan (eds.) Advances in Distributed and Parallel Knowledge Discovery, AAAI Press 1999.

[23] E. Houstis, A. Catlin, N. Dhanjani, J. Rice, J. Dongarra, H. Casanova, D. Arnold, G. Fox, *Problem-Solving environments*, in The Parallel Computing Sourcebook, M. Kaufmann Publishers, 2002.

Biographies

Carlo Mastroianni obtained his PhD in Computer Engineering in 1999 from the University of Calabria, Italy. He had a fellowship at the Politecnico of Turin (Italy) and worked as a Software Engineer at the Computer Department of the Italian Government, in Rome. He is currently a researcher at the ICAR-CNR - Institute for High Performance Computing and Networks of the Italian National Research Council. His research interests include grid computing, multicast and multimedia networks, mobile communication systems.

Domenico Talia is a professor of computer science at the Faculty of Engineering at the University of Calabria, Italy. He received his Laurea degree in Physics from the University of Calabria, Italy. From 1997 to 2001, he was a senior researcher at the ISI-CNR - Institute of Systems Analysis and Information Technology of the Italian National Research Council. His main research interests include parallel computing, parallel data mining, grid computing, parallel programming languages, computational science, and cellular automata.
Talia is a member of the editorial boards of the IEEE Computer Society Press, the Parallel and Distributed Practices journal, the Future Generation Computer Systems journal, and the INFORMATION journal, in addition he a member of the advisory board of Euro-Par conference series and a member of the advisory committee of the IEEE Task Force on Cluster Computing (TFCC). He served as a distinguished speaker in the IEEE Computer Society Chapter Tutorials Program and in the IEEE Computer Society Distinguished Visitors Program. He was guest editor of special issues of IEEE Transactions on Software Engineering, Parallel Computing and Future Generation Computer Systems and he is serving as a

program committee member of several conferences. He published three books and more than 120 papers in international journals and conference proceedings. He is member of the ACM and the IEEE Computer Society.

Paolo Trunfio since 2001 is a PhD student in Computer Engineering at the University of Calabria, Italy. He collaborates with the ICAR-CNR - Institute for High Performance Computing and Networks of the Italian National Research Council, in the area of grid computing. His current research interest include parallel and distributed computing, and peer-to-peer systems.