UNICZ

UNICAL

Architecture, Metadata, and Ontologies in the *Knowledge Grid*

Mario Cannataro
University “Magna Græcia” of Catanzaro, Italy
cannataro@unicz.it

joint work with: D. Talia, C. Comito, A. Congiusta, C. Mastroianni, A. Pugliese, P. Trunfio, P. Veltri

Outline

1. THE KNOWLEDGE GRID
    - Services
    - Architecture
    - Metadata: efficient access to Knowledge Grid resources
    - Ontologies: semantic modeling of Data Mining domain
    - Design and execution of Distributed Data Mining applications
2. Conclusions and future work
3. References

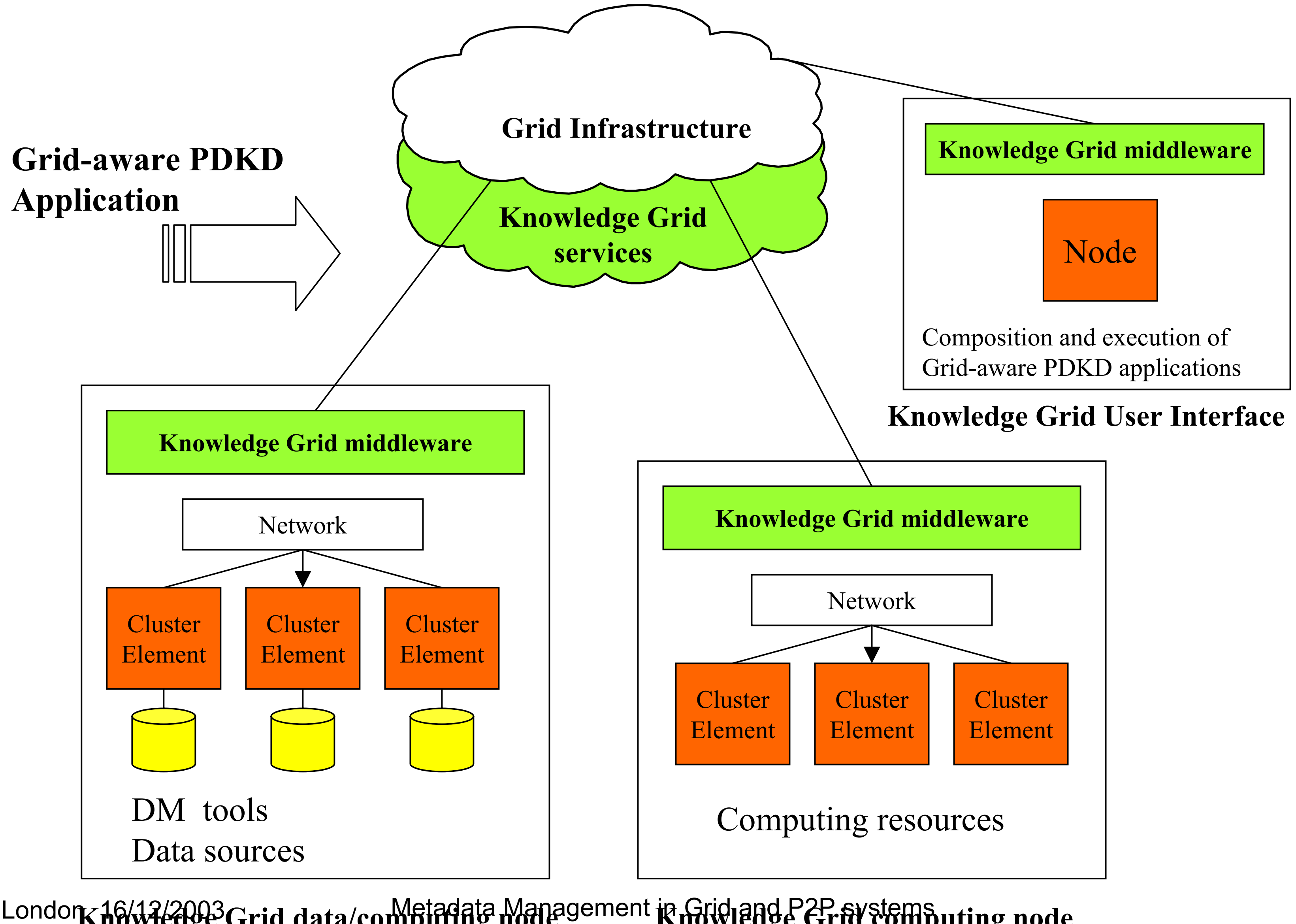
The Knowledge Grid (1)

- A software environment that integrates Data Mining techniques and Grid resources to build Grid-aware Data Mining Applications
  - **Metadata** for describing data mining tools, data sources, grid resources
  - **Ontologies** for Semantic Modeling of the application domain
- The GLUE that allows to integrate existing data mining tools in a Grid-aware PDKD application
  - Data mining tools are interfaced with lower-level Grid mechanisms and services
  - Grid infrastructure, such as Globus Toolkit, provide basic services (scheduling, security, etc.)
- Knowledge Grid applications are
  - designed through
    - selection of available tools and data sets
    - visual composition of such components to form distributed workflows
    - hiding the details of Grid programming and Grid resource location
  - and executed leveraging Grid resource management and scheduling

The Knowledge Grid (2)

- The architecture is a "bag of services" useful to develop distributed KDD applications:
  - Search, Location and Remote access to DM resources (DM tools, ETL tools, data sources)
  - Graphical Definition, Design and Development of PDKD applications through the "semi-automatic composition" of Data Mining components, driven by ontology and metadata
  - Execution and Monitoring of PDKD applications on the Grid by using grid services (at least resource allocation and scheduling)
  - Management of the extracted knowledge
- A Knowledge Grid computation, i.e. a PDKD application, uses:
  - a set of Knowledge Grid nodes (K-GRID nodes), i.e. Grid nodes implementing the Knowledge Grid services and offering:
    - DM tools, or Data sources or simply Computing power
  - a Grid infrastructure offering basic Grid services

Knowledge Grid infrastructure



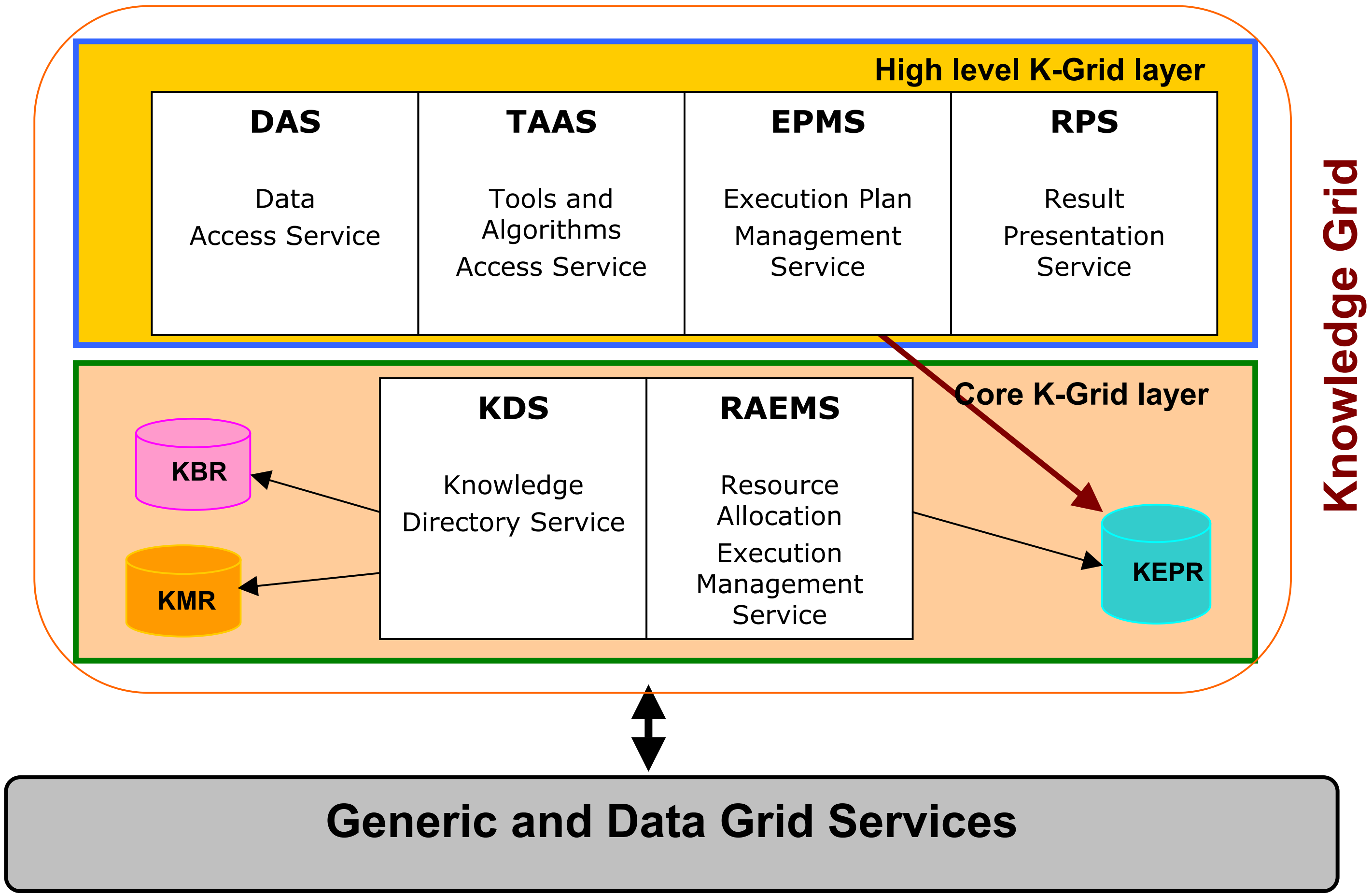
Grid Infrastructure
Knowledge Grid services
Grid-aware PDKD Application
Knowledge Grid middleware
Node
Composition and execution of Grid-aware PDKD applications
Knowledge Grid User Interface
Knowledge Grid middleware
Network
Cluster Element
Cluster Element
Cluster Element
DM tools
Data sources
Knowledge Grid data/computing node
Knowledge Grid middleware
Network
Cluster Element
Cluster Element
Cluster Element
Computing resources
Knowledge Grid computing node

Knowledge Grid services

The *Knowledge Grid* services are organized in two hierarchic levels:

- *Core K-Grid layer* offers the basic services for the definition, composition and execution of a PDKD computation over the Grid
  - manages all metadata describing features of data sources, third party data mining tools, data management, and data visualization tools and algorithms.
  - coordinates the application execution by attempting to fulfill the application requirements and the available grid resources.
- *High level K-Grid layer* includes services used to:
  - compose, validate, and execute a parallel and distributed knowledge discovery computation.
  - store and analyze the discovered knowledge.

Knowledge Grid architecture (1st version)


High level K-Grid layer
DAS
Data Access Service
TAAS
Tools and Algorithms Access Service
EPMS
Execution Plan Management Service
RPS
Result Presentation Service
Core K-Grid layer
KBR
KMR
KDS
Knowledge Directory Service
RAEMS
Resource Allocation Execution Management Service
KEPR
Knowledge Grid
Generic and Data Grid Services

Metadata model for resources description

- The metadata model can play a main role for an efficient management of heterogeneity. Metadata should:
  - document in a simple and human-readable fashion the features of a data mining application.
  - allow the efficient search of resources.
  - provide an effective way to access resources.
  - be programmatically accessed by software tools that support the visual building of Knowledge Grid computations (e.g. VEGA, a Visual Environment for Grid Applications).
- The current Knowledge Grid implementation uses the Globus Monitoring and Discovery Service (MDS), based on the LDAP protocol, to publish, query and access Grid resources.
- We use the MDS model only for generic grid resources, but extend it with an XML metadata model to manage specific Knowledge Grid resources.

Metadata model and K-GRID resources

- **K-GRID resources involved are:**
  - Computational resources
  - Data sources (structured or unstructured)
  - Mining tools and algorithms
  - Knowledge extracted (models, i.e. DM results)
  - Visualization tools and algorithms
  - Execution Plans
- **Each K-GRID resource is described by (XML, LDAP) metadata**
  - The **KMR** (Knowledge Metadata Repository) is implemented by LDAP entries in the Globus MDS service, referencing locally stored XML documents
    - LDAP object classes such as **`k-gridDataSources`** and **`k-gridSoftware`** are used by a K-Grid node to publish the availability of data sources to be mined and public-accessible data mining tools
  - The **KBR** (Knowledge Base Repository) stores discovered models
  - The **KEPR** (Knowledge Execution Plan Repository) stores Execution Plans

Resource Metadata: Data Mining Software

- The metadata structure is composed of two parts:
  - A **`Description`** part, that is used to classify software on the basis of following parameters as:
    - the kind of data sources the software works on;
    - the kind of knowledge to be discovered;
    - the techniques adopted;
    - the driving method (autonomous, interactive, driven by data).
  - An **`Usage`** part, that contains the information needed by clients to access and use the software:
    - generic access information (e.g. URL)
    - software invocation syntax.

Example: Metadata of the Software AutoClass

```
<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>
    <Arg description="create a report" type="alternative" value="-reports">
     <Arg description="a .results-bin file" type="required"/>
     ...
    </Arg>
   </Arg>
  </Syntax>
 </Usage>
</DataMiningSoftware>
```

Resource Metadata: Data Sources

- Data sources are the input on which mining algorithms work to extract new knowledge. Metadata is composed of:
  - A `Description` section, that includes information for retrieving the data source:
  - A `Structure` section, that contains information on logical and physical data structure.

Example: Metadata of a Flat File

```
<FlatFile>
 <Access>
  <Location> /usr/share/imports-85c.db2 </Location>
   ...
 </Access>
 <Structure>
  <Format>
   <AttributeSeparatorString>,</AttributeSeparatorString>
   <RecordSeparatorString>#</RecordSeparatorString>
  </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>
```

Discovered Knowledge Models

- Knowledge models (results) are represented by means of the PMML language (Predictive Model Markup Language).
- PMML is a vendor-independent XML language that provides DTDs for describing several kinds of models (classifications rules, association rules, clusters etc.)

Execution Plans

- A distributed data mining computation is a process composed by several sequential and/or parallel tasks
- An `executon plan` is a graph that describes the interactions among tasks
- An execution plan is:
  - `Abstract`, if contains a t least one abstract resource
  - `Instantiated`, if all resources are concrete
- Instantiation is performed by the scheduler of the RAEMS service.
- From an `abstract` execution plan, several `instantiated` execution plans could be generated, depending on the resources available on th Knowledge Grid at different times

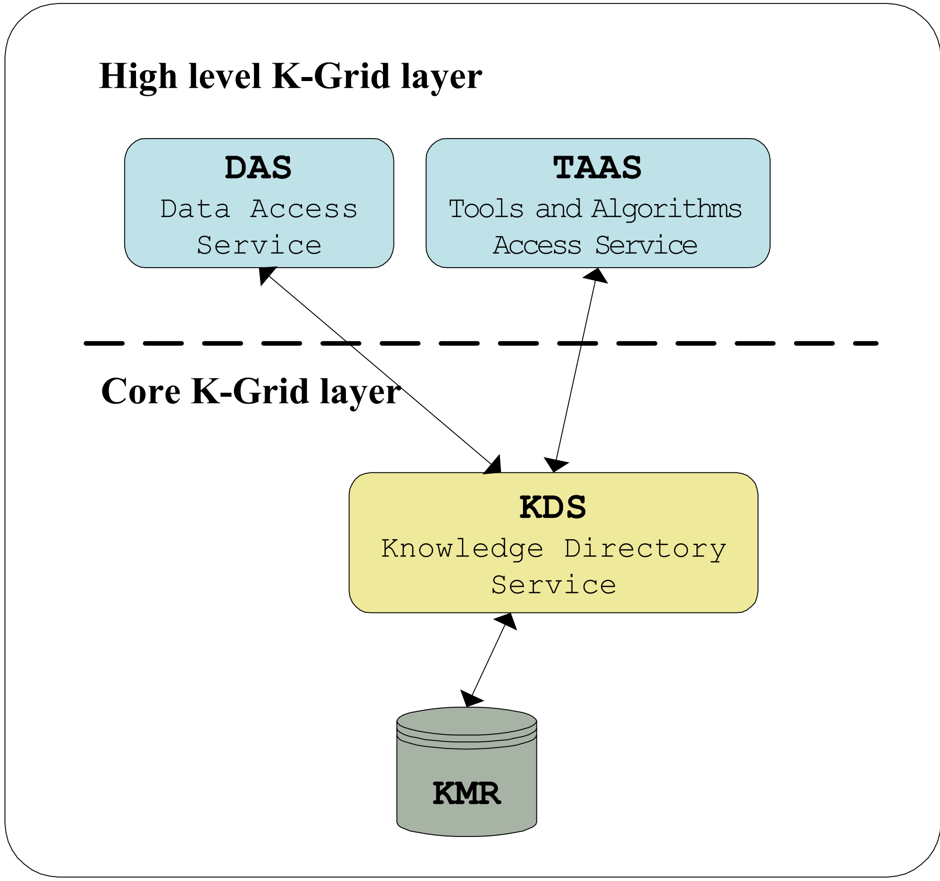
Example: Metadata of an Execution Plan

```
<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"/>
                                    ...
```

Metadata Management and Flows (1/2)

- Resource metadata are published in the KMRs of the Knowledge Grid nodes
- The user specifies the features of the resources he needs
- DAS and TAAS services search the KMRs for the requested resources, using the core level KDS service
- Metadata describing the discovered resources are delivered to the user

Resource metadata flows

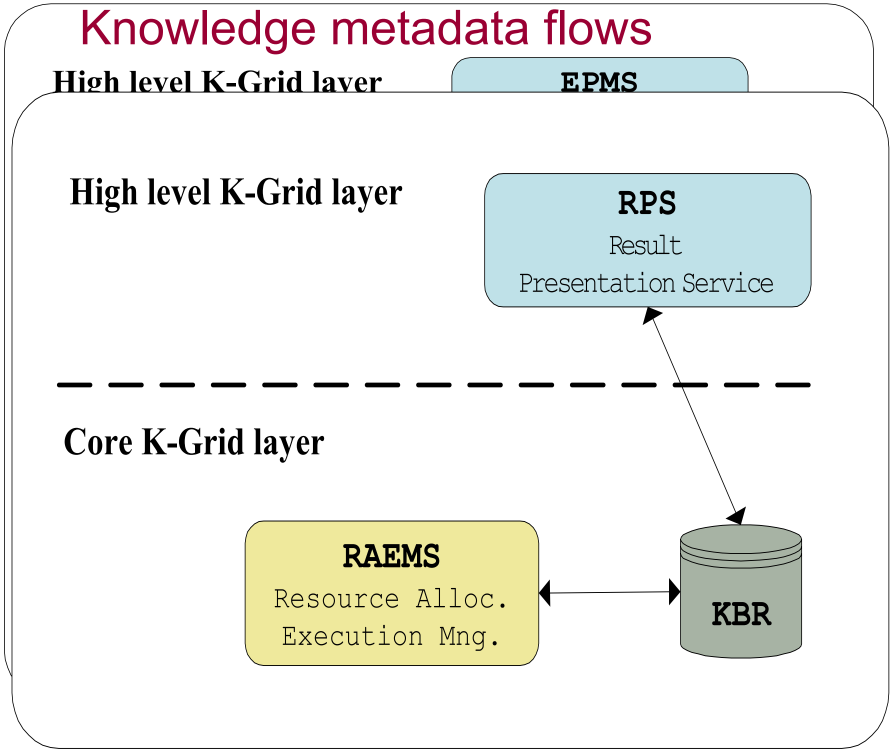


Metadata Management and Flows (2/2)

5. Metadata related to software, data and operations are combined into an execution plan
6. Execution plan metadata are managed by the EPMS service, accessed by the core level RAEMS service, and stored in the KEPR

Execution plan metadata flows

Knowledge metadata flows



7. After application execution (driven by the RAEMS), model metadata are stored into the KBR and processed by the RPS service
8. New and modified resources' metadata are published in the KMRs

Ontology: semantic modeling of Data Mining domain

- A Domain Ontology is the shared understanding of a domain of interest, described through a set of classes (concepts), relations, functions, axioms and instances
- Problem: making easier the development of distributed KDD applications
  - many available pre-existing software tools
  - many different implementations and methodologies
- in heterogeneous, multi-owned environments (the Grid)

☞ DAMON = DAta Mining ONtology

DAMON design: ontology scope

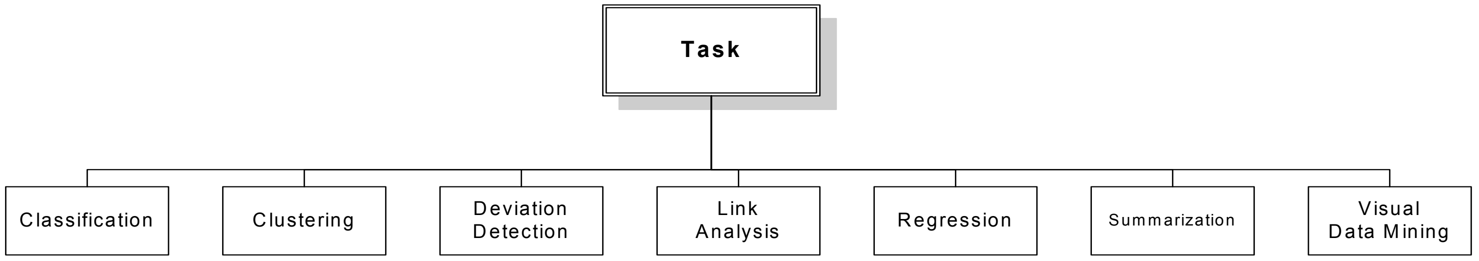
- for what we are going to use DAMON
  - Classify Data Mining Software on the basis of some parameters useful to select the more suitable ones to solve a KDD problem:
    - The Data Mining **task** performed by the software
    - The **methodologies** that the software uses in the data mining process
    - The kind of **data sources** the software works on
    - The degree of required **interaction** with the user
- for what type of questions the ontology should answer
  - to assist the user suggesting him/her the software to use on the basis of the user's requirements/needs
  - to allow the semantic search (concept-based) of data mining software and others data mining resources

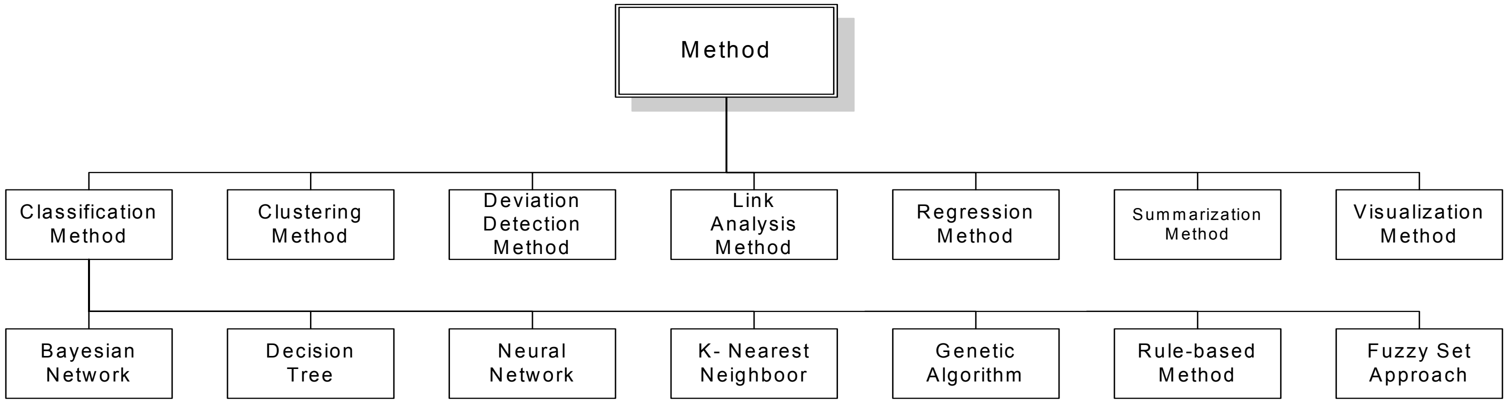
DAMON design: basic concepts and properties

- ***Task***: the knowledge discovery goal that the software is intended for
- ***Method***: a data mining methodology used to discover the Knowledge
- ***Algorithm***: the way through which a data mining task is performed
- ***Software***: an implementation (in a programming language) of a data mining algorithm
- ***Suite***: implements a set of data mining algorithms
- ***Data Source***: 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 andor supported

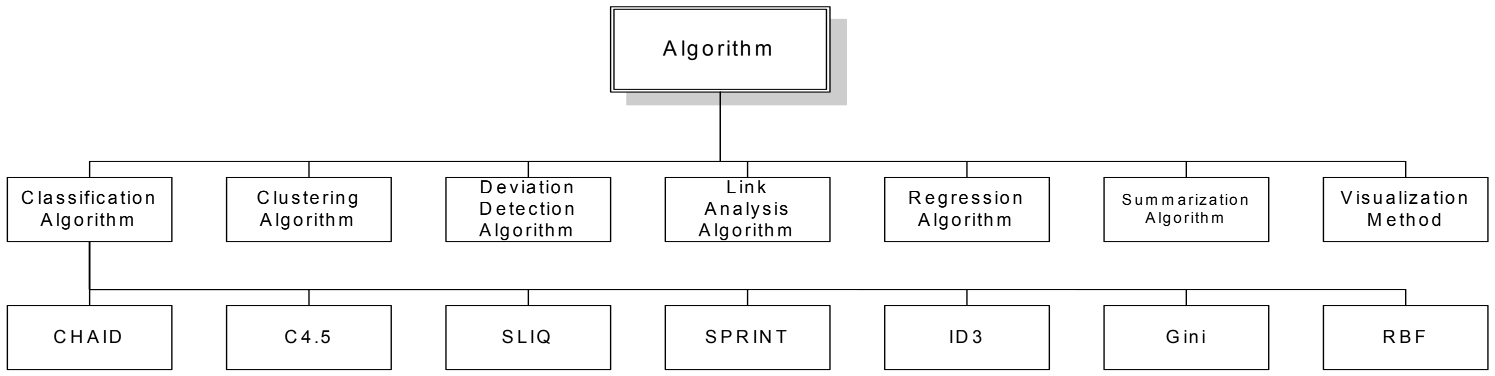
DAMON design: concepts hierarchy (1)

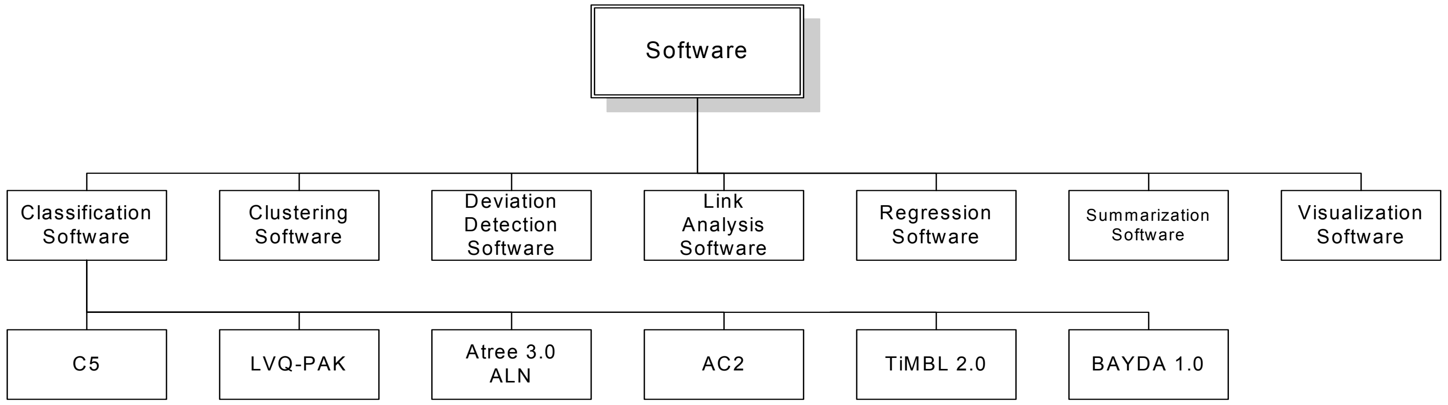
DAMON has several small taxonomies derived from the specialization of the basic concepts of the ontology.





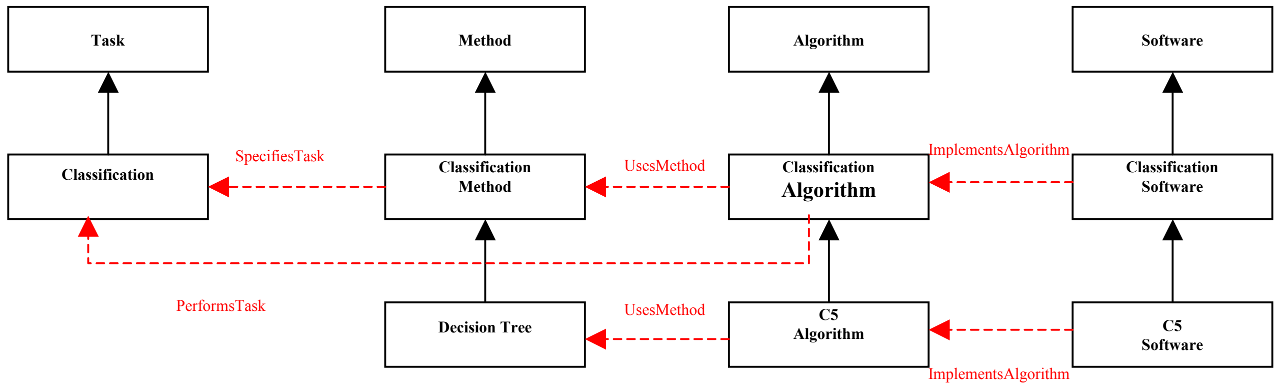
DAMON design: concepts hierarchy (2)





DAMON design: axioms

- Axioms allow to represent more information about concepts:
  - constraining on their internal structure, and their mutual relationships,
  - verifying their correctness or deducting new information.



- Example: conceptualization of the C5 Classification Software
  - C5 is a Classification Software implementig the C5 Algorithm;
  - C5 Algorithm is a Classification Algorithm
    - performing the Classification Task
    - and using the Decision Tree method that is constrained to be a Classification Method specifying the Classification task

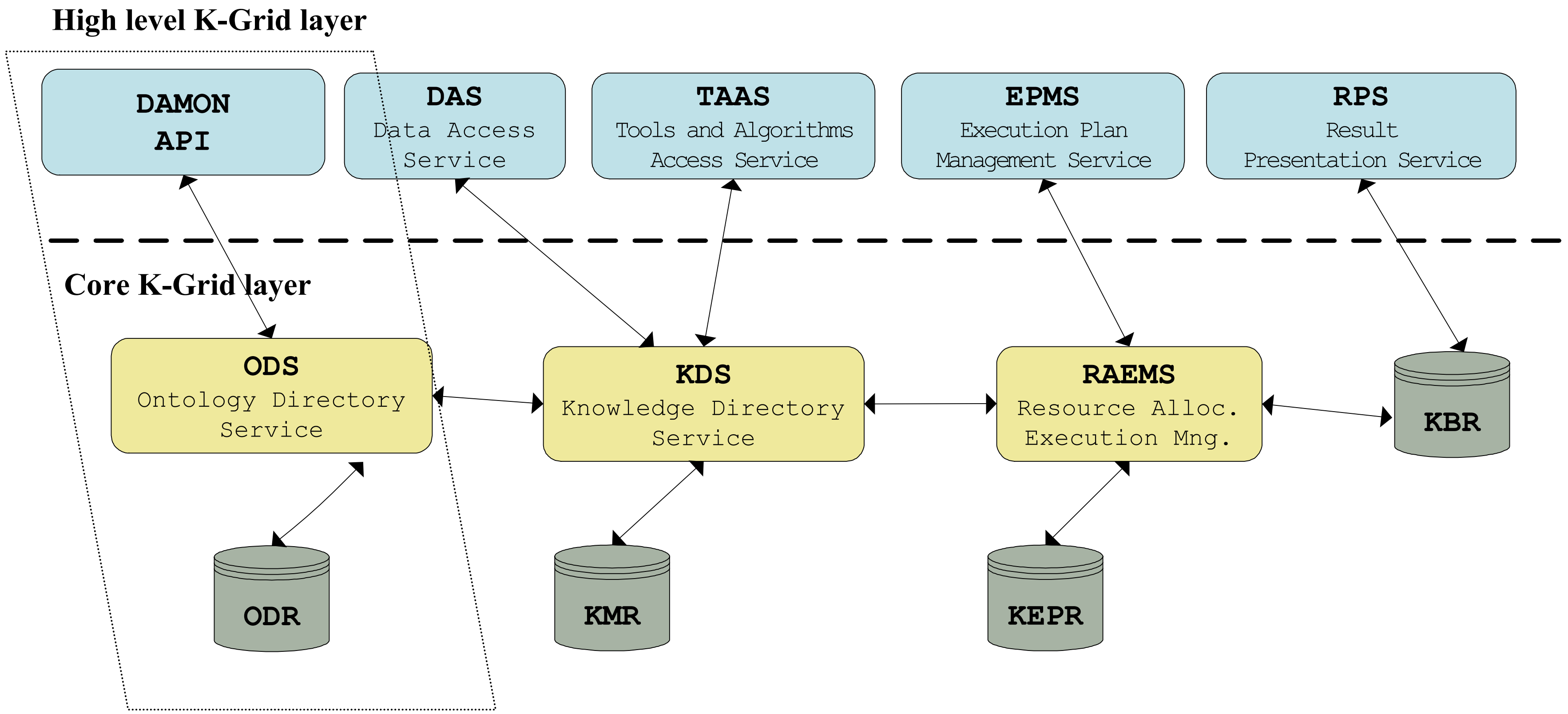
Ontology-based Grid Programming

- DAMON is used as an ontology-based assistant that:
  - suggests the Parallel and Distributed Knowledge Discovery (PDKD) application designer what to do and what to use on the basis of his/her needs
  - makes possible semantic (concept-based) search of data mining software
  - enhancing the application formulation and design,
  - helping the user to select and configure the most suitable Data Mining solution for a specific KDD process
- DAMON can be thought as a knowledge base on specific software/data sources metadata
  - domain ontology gives general information about resources,
  - metadata gives specific information about installed software (maintained where resources reside).
- Ontology is logically centralized, metadata are distributed

DAMON operations

- Browsing. i.e. showing:
  - which data mining software are available,
  - where software can be found,
  - and how software can be accessed.
- Searching all the available software that satisfy some user requirements:
  - 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 software implementing a desired data mining algorithm;
  - find software performing a specified data mining task;
  - find software/algorithm using particular methods;
  - find data sources about a specific topic.

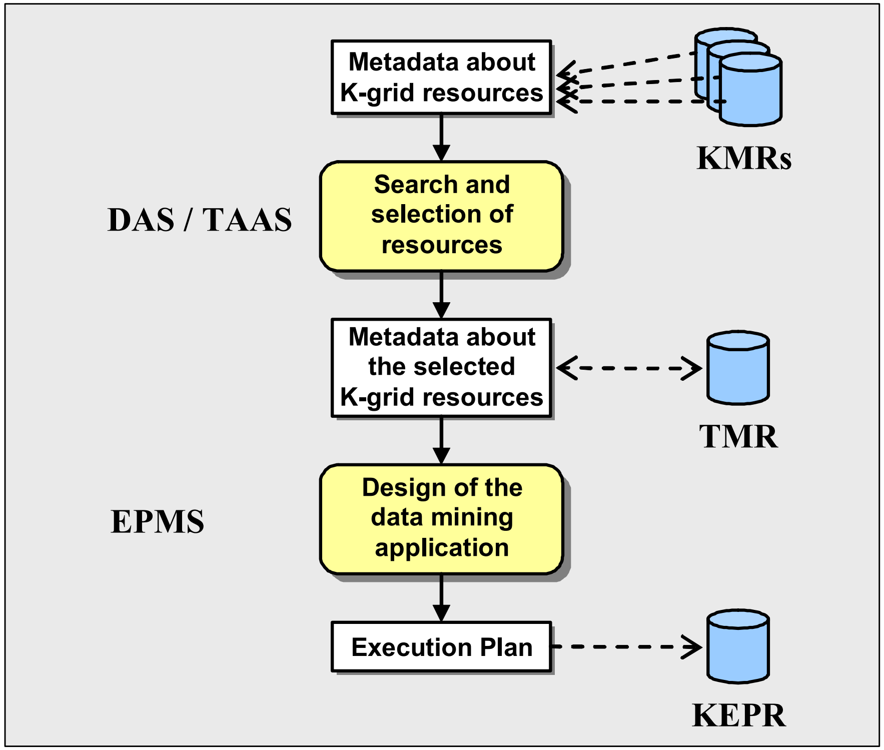
Knowledge Grid Architecture (2nd version): ontology-based services


High level K-Grid layer
DAMON API
DAS Data Access Service
TAAS Tools and Algorithms Access Service
EPMS Execution Plan Management Service
RPS Result Presentation Service
Core K-Grid layer
ODS Ontology Directory Service
KDS Knowledge Directory Service
RAEMS Resource Alloc. Execution Mng.
KBR
ODR
KMR
KEPR

DAMON=Ontology data + index to specific XML metadata about installed software

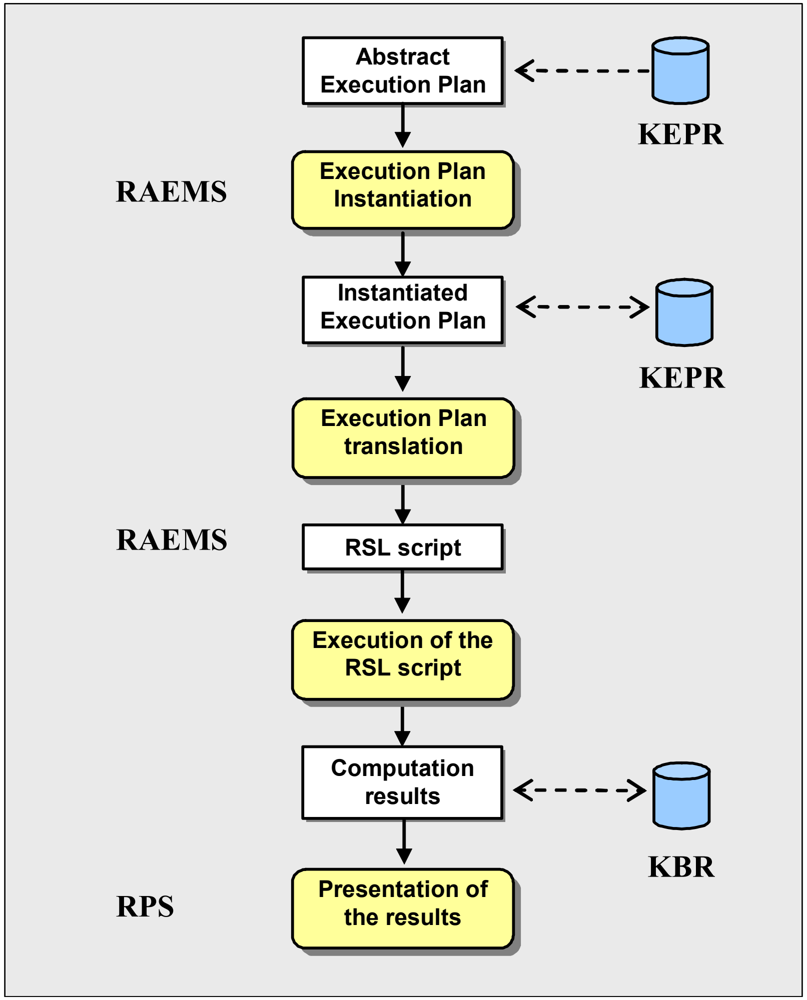
Designing a Distributed Data Mining Application

1. Search and selection of the resources to be used in the knowledge discovery application (DAS and TAAS);
   - ontology-based resource selection
   - XML metadata access
2. Visual composition of the application through a *graphic model* that represents the involved resources and their relationship (EPMS);
3. Generation of the *execution plan* corresponding to the graphic model of the application (EPMS).


Metadata about K-grid resources
KMRs
DAS / TAAS
Search and selection of resources
Metadata about the selected K-grid resources
TMR
EPMS
Design of the data mining application
Execution Plan
KEPR

Executing a Distributed Data Mining Application

1. Instantiation of the Execution Plan;
2. Translation of the instantiated Execution Plan into a specific Grid *broker script*;
3. Application execution, i.e., submission of the broker script to the allocation manager of the underlying Grid middleware, and
4. Results presentation.



Abstract Execution Plan
KEPR
RAEMS
Execution Plan Instantiation
Instantiated Execution Plan
KEPR
Execution Plan translation
RAEMS
RSL script
Execution of the RSL script
Computation results
KBR
RPS
Presentation of the results

Conclusions and future work

- Metadata and Ontologies are major players when complex applications are developed in Grid environment.
- We proposed a two layer metadata schema that separates concepts and metadata about resources :
  - DAMON ontology simplifies the formulation and building of applications through the composition and reuse of software components
  - XML-based metadata specify details about installed software and data sources, allowing efficient access to resources
- We showed how metadata and ontologies are managed to build and execute distributed data mining applications on the Knowledge Grid.
- The proposed ontology-based application design approach will be used as a core method in general purpose Grid-based Problem Solving Environments

Thank you!

References

- M. Cannataro, **Knowledge Discovery and Ontology-based services on the Grid**, Ninth Global Grid Forum, Semantic Grid Research Group Workshop, October 5-8, 2003, Chicago
- M. Cannataro and C. Comito, "**A Data Mining Ontology for Grid Programming**", 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing, in conjunction with WWW2003, Budapest, 20-24 May 2003.
- C. Mastroianni, D. Talia, Paolo Trunfio: **Managing Heterogeneous Resources in Data Mining Applications on Grids Using XML-Based Metadata**. IPDPS 2003: 99. (2003).
- M. Cannataro, D. Talia, "**KNOWLEDGE GRID: An Architecture for Distributed Knowledge Discovery**", *Comm. of the ACM*, January 2003.
- M. Cannataro and Domenico Talia, **Towards the Next-Generation Grid: Semantics and Knowledge Grids**, submitted to IEEE Intelligent systems Special Issue on Semantic Grid.
- S. Malaika, A. Eisenberg, J. Melton. **Standards for Databases on the Grid**. ACM SIGMOD Record, Vol. 32, No. 3, September 2003.

Credits

- The ICAR-UNICAL-UNICZ Grid Team
  - Domenico Talia
  - Mario Cannataro
  - Carmela Comito
  - Antonio Congiusta
  - Carlo Mastroianni
  - Andrea Pugliese
  - Paolo Trunfio
  - Pierangelo Veltri
- Web site: www.icar.cnr.it/kgrid

Contact

Mario Cannataro

University "Magna Græcia" of Catanzaro, ITALY

Email: cannataro@unicz.it,

Web: www.icar.cnr.it/cannataro