Grid-enabling data mining applications with DataMiningGrid: An architectural perspective

Vlado Stankovski[a], Martin Swain[b,*], Valentin Kravtsov[c], Thomas Niessen[d], Dennis Wegener[d], Jörg Kindermann[d], Werner Dubitzky[b]

[a]*University of Ljubljana, Jamova cesta 002, SI-1000 Ljubljana, Slovenia*

[b]*University of Ulster, Cromore Road, Coleraine BT52 1SA, Northern Ireland*

[c]*Technion - Israel Institute of Technology, Technion City, 32000 Haifa, Israel*

[d]*Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, D-53754 Sankt Augustin, Germany*

Abstract

The DataMiningGrid system has been designed to meet the requirements of modern and distributed data mining scenarios. Based on the Globus Toolkit and other open technology and standards, the DataMiningGrid system provides tools and services facilitating the grid-enabling of data mining applications without any intervention on the application side. Critical features of the system include flexibility, extensibility, scalability, efficiency, conceptual simplicity and ease of use. The system has been developed and evaluated on the basis of a diverse set of use cases from different sectors in science and technology. The DataMiningGrid software is freely available under the Apache License 2.0.

Key words: Data mining, distributed systems, service-oriented architecture, grid

1 Introduction

Due to the increased computerization of many industrial, scientific, and public sectors, the amount of available digital electronic data is growing at an unprecedented rate. The effective and efficient management and use of these data, and in particular their transformation into information and knowledge, is considered a key requirement for success in such knowledge-driven sectors. *Data mining* [7,17,25] (also known as knowledge discovery in databases) is the de-facto technology addressing this information need. Data mining is an inductive, iterative process that extracts information or knowledge patterns from volumes of data [36].

The rise of distributed computing environments has profound implications in terms of how distributed data are analyzed and interpreted [48,32]. Future data mining applications will need to operate on massive data sets and the programs for processing, analyzing, evaluating, and visualizing the data will increasingly reside at geographically distributed sites on heterogeneous computing platforms. Distributed data mining and in particular grid-enabled data mining has become an active area of research and development in recent years [6,38].

Grid computing can be viewed as a virtual computing architecture that provides the ability to perform higher throughput computing by taking advantage

* Corresponding author:
Email address: `mt.swain@ulster.ac.uk` (Martin Swain).

of many computers connected by a network (usually local and wide area networks) [20,19].

Grid technology evolves rapidly and this often poses challenges, such as interoperability problems, when building applications on open source technology as the available functionality frequently changes. In the past years, grid standardisation efforts have concentrated on achieving the goal to define an open framework for modeling and accessing stateful resources by using Web services. The actual OASIS standard named Web Services Resource Framework (WSRF) v. 1.2 was approved only recently in April 2006. This paper describes a large-scale effort aimed at developing a system that brings WSRF-compliant grid computing technology to users and developers of data mining applications. The resulting system is the output of the DataMiningGrid project, which was largely funded by the European Commission [16].

A challenge undertaken by the DataMiningGrid was to develop an environment suitable for executing data analysis and knowledge discovery tasks in a wide range of different application sectors, including the automotive, biological and medical, environmental and ICT sectors. By analysing the diverse requirements of these applications the DataMiningGrid has developed generic technology for grid-based data mining. While existing grid technology already provides a diverse set of generic tools, its emphasis on generality means that the available functionality may lack the sophistication needed to specifically support advanced data mining use-cases. Therefore the DataMiningGrid developed enhancements to open source grid middleware in order to provide the specialised data mining functionality required by our use-cases. This includes functionality for tasks such as data manipulation, resource brokering, application searching according to different data mining tasks and methodologies, and supporting different types of functionality for parameter sweeps. The result is a grid with all the generic functionality of its component middleware, but with additional features that ease the development and execution of complex data mining tasks. Some key aims in the development of the DataMiningGrid system included the following:

(1) Grid transparency: Domain-oriented end users should be able to carry out the data mining tasks without needing to understand detailed aspects of the underlying grid technology.
(2) Application development support: Developers of data mining solutions should be able to grid-enable existing data mining applications, techniques and resources (e.g., database management systems) with little or no intervention in existing application code.
(3) Service-orientated architecture and interoperability: The system should adhere to existing and emerging grid and grid-related standards such as WSRF, and be based on widely used open source technology.

To enable a detailed assessment of the benefits and shortcomings of the DataMiningGrid technology in light of related developments, this presentation is deliberately intended to be comprehensive and as much as possible self-contained. The remainder of the paper is organized as follows. Section 2 presents the results obtained from an extensive requirements analysis phase. Sections 3, 4 and 5 present a detailed description of the system and its components. Section 6 describes an evaluation of the system based on two selected use cases, and Section 7 discusses the results in the context of related work. Finally, Section 8 provides a short summary and some concluding remarks.

2 Requirements of grid-based data mining

The main function of a grid-based data mining system is to facilitate the sharing of data, data mining application programs, processing units and storage devices in order to improve existing, and enable novel, data mining applications. Such a system should take into account the unique constraints and requirements of data mining applications with respect to the data management and data mining software tools, and the users of these tools. These high-level goals lead to a natural breakdown of the requirements for a grid-based data mining system – we distinguish *user*, *application* and *system* requirements. The user requirements are dictated by the need of end users to define and execute data mining tasks, and by developers and administrators who need to evolve and maintain the system. Application program requirements are driven by technical factors such as resource type and location, software and hardware architectures, system interfaces, standards, and so on. The DataMiningGrid project's main aim is to identify the requirements for a grid-based data mining system and to design, implement and evaluate a system that meets these requirements.

To determine and specify the detailed requirements of the DataMiningGrid system, the project analyzed a representative set of use cases. The use cases are based on real-world data mining applications in the following areas: medicine, biology, bioinformatics, customer relationship management and car repair log diagnostics in the automotive industry, monitoring of network-based computer systems and civil engineering (ecological modeling). Some selected use cases are listed below. Section 6 provides additional information on the implementation and evaluation of two of these use cases.

Use Case 1: Evolutionary algorithms for reverse-engineering gene regulatory networks: This is a compute-intensive data mining application which derives the interaction topology as well as interaction logic (i.e., the functional description how one gene expression influences another) from gene expression data [39].

Use Case 2: Analysis of molecular dynamics protein unfolding simulation data: Computer simulations based on molecular dynamics generate large volumes of data ($>$ 100 MB per simulation) [9]. To facilitate analysis of such large data sets, this use case investigates the shipping of data mining applications to an execution machine near (i.e., with a very fast data transfer connection) the data source.

Use Case 3: Data mining of distributed medical databases: This application involves medical databases residing in several geographic regions in Slovenia. Privacy and security considerations are essential for this application. In addition, it is important that the application scales well as more and larger databases become part of this analysis.

Use Cases 4-6: Text mining: These use cases include fast distributed text classification (Section 6.3) and ontology learning for quality and customer relationship management in the automotive industry. Another text mining use case is concerned with information retrieval in digital libraries. A critical aspect of this is the distribution of the source documents.

Use Case 7: Ecological modeling: This application is concerned with the task of building mathematical models of ecosystems, in particular population dynamics in aquatic ecosystems [42]. Data mining is used to discover equations that reflect the dynamics of the ecosystem (Section 6.2).

Use Case 8: Grid monitoring scenarios: Traditional monitoring services in distributed applications are based on gathering local logs. Experts browse these logs manually or by using software. The aims of this data mining application are to provide more powerful monitoring functionality, to allow ongoing analysis of the system's condition, and to provide critical diagnostic information [30].

The main requirements arising from the analysis of the use cases are described in the following subsections.

2.1 User requirements

The typical end user is mainly driven by concepts, issues, tasks and requirements arising from his or her application domain. It is not uncommon that such end users have only limited knowledge of the technical details of the underlying data mining and grid technology. A biologist, for example, may want to classify scientific texts in terms of a given set of cancer classes in order to identify relationships between cancer mechanisms and genes. Of key importance to this type of user are the data reflecting domain concepts and particular type of data mining analysis that needs to be performed. Besides efficiency and effectiveness, the main concerns of such a user include ease-of-use, responsiveness, interactivity, flexibility and conceptual simplicity of a grid-enabled data mining application.

At the other end of the spectrum are end users who possess considerable knowledge of data mining and grid technology or both. Such users may want to make specific choices in terms of defining and configuring a data mining process (selecting algorithms, setting parameters, defining workflows, and specifying preferences for grid resources used to execute a particular data mining application). An analyst, for example, may want to define a detailed workflow designed to pre-process and subsequently cluster data in a specific way with particular algorithms. Such a user is concerned with openness and flexibility in configuring and executing data mining applications. Clearly, efficiency and effectiveness is of great importance to this type of user.

Other categories of users are application and system developers, and system administrators. While rarely performing actual analyses, such users are concerned with requirements such as extensibility, maintainability, user management, system integration, and so on.

From the above considerations, and the use case scenarios analyzed in the DataMiningGrid project, the following key user requirements of the DataMiningGrid system have been identified:

(1) *Effectiveness, efficiency, novel use.* (1) A grid-enabled version of an existing data mining application should offer one or more of the following benefits to the end user: (a) be more effective, (b) be more efficient (higher throughput, which relates to speed-up), or (c) provide a novel feature or a novel way of using the application in comparison to the non-grid version. (2) The DataMiningGrid system should be scalable, i.e., it should allow seamless adding of grid resources to accommodate increasing numbers of users and user demands without performance loss.

(2) *Scope of application domain and task.* (1) The system should be capable of accommodating a widely differing set of existing end user application domains, and data mining tasks and scenarios (e.g., text mining, rela-

tional mining, pre-processing, clustering, network induction, and so on). (2) Furthermore, it should be flexible enough to permit entirely novel data mining applications that are impractical or impossible outside a grid environment.

(3) *Ease and flexibility of use.* (1) Application-oriented end users should be able to use the system without needing to know technological details, in particular those relating to the underlying grid computing technology. (2) Technology-aware end users should be able to define, configure and parameterize details of the data mining application, the data mining process, and various grid aspects. (3) Users should be able to search for available data mining applications based on various criteria.

(4) *Monitoring.* Users should be able to monitor the progress of their applications and be able to respond with the appropriate action (e.g., abort a running application based on exceptions or intermediate results).

(5) *Extensibility.* (1) Application developers should be able to grid-enable existing data mining applications with little or no modification to the original data mining application program. (2) System developers should be able to extend the features of the core system without major modifications to the main system components.

(6) *Maintenance and integration.* (1) Application developers, system developers, and administrators should be able to easily integrate new applications and core system components with other technology (networks, Web services, grid components, user interfaces, etc). (2) Maintenance (e.g., upgrades, user management) of the core system and the already grid-enabled applications should be simple, seamless and non-interruptive.

2.2 Application and system requirements

To meet the user requirements presented in Section 2.1, the DataMiningGrid system needs to fulfill a number of additional technical requirements relating to data mining application software (data, programs) and the underlying grid components. We call these requirements data mining application and (data mining grid) systems requirements. The following is a list of the most important requirements in this category:

(1) *Resource sharing and interoperation.* The system is required to facilitate the seamless interoperation and sharing of critical data mining resources, in particular, data mining application programs, data resources, storage devices and processing units.

(2) *Applications.* (1) The DataMiningGrid system should be able to run a wide range of data mining application programs. (2) To facilitate ease of use and flexibility in defining data mining tasks and processes, the system needs to provide a flexible workflow component with a graphical user interface. (3) In order to execute data mining applications within the DataMiningGrid system, the system needs to understand the requirements, constraints, and user-defined settings associated with an application. For this purpose, each application needs to provide an application description (see Section 4.1).

(3) *Resource brokering and job scheduling.* Like any grid system, the DataMiningGrid system needs to (1) match available resources to job requests (resource broker), (2) schedule the execution of the jobs on matched resources (scheduler), and (3) manage and monitor the execution of jobs (job execution and monitoring) [26]. Unique requirements for resource

brokering and job scheduling of a data mining grid system include data-oriented scheduling, parameter sweep support, and consideration of: (a) the type of data mining task (e.g., classification, clustering), (b) the data mining technique (artificial neural network, decision tree), and (c) the data mining method (e.g., C4.5, C5.0).

(4) *Standardization.* To facilitate interoperation and evolution of the DataMiningGrid system, the design and implementation should be based on existing and emerging grid and grid-related standards and open technology.

3 DataMiningGrid system architecture

The DataMiningGrid architecture is depicted in the diagram of Figure 1. Generally, components in higher layers make use of components organized in lower layers. The bottom layer labeled Resources shows grid software (data, data mining applications) and hardware resources (storage devices, processing elements and computer networks). The main grid middleware layer (large box labeled Globus Toolkit 4) provides the core grid middleware functionality to the DataMiningGrid system. High-level DataMiningGrid Services are organized in the layer above the middleware layer (box labeled DMG High-Level Services). Finally, the top layer depicts the client side components of DataMiningGrid applications (large box labeled DMG Client Applications). Below we will describe the different layers and their relationships in detail, in particular those components developed by the DataMiningGrid project (highlighted in red in Fig. 1).

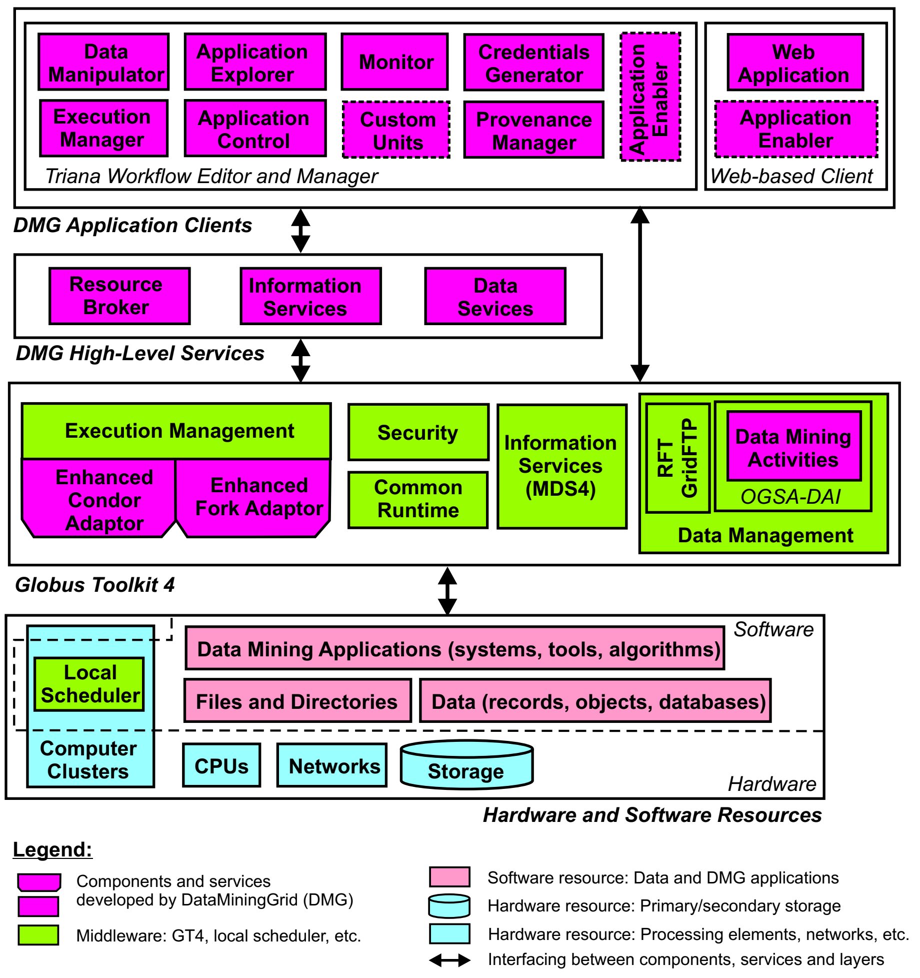


Fig. 1. **DataMiningGrid system architecture**

3.1 Resources

The box labeled Hardware and Software Resources at the bottom of the diagram in Fig. 1 illustrates the grid resources layer in the DataMiningGrid system architecture. The main purpose of the DataMiningGrid system is to facilitate sharing and interoperation of such resources in the context of data mining.

Typical basic hardware resources include processing units (CPUs) and primary and secondary storage devices (e.g., RAM, magnetic disks). These are crucial for processing and storing large quantities of data.

Clusters (or cluster computers), nowadays common in many organizations, are a special kind of resource consisting of collections of loosely coupled computers. In the diagram, the label Local Scheduler represents a typical distributed batch or scheduler system (a grid middleware) used to facilitate convenient access to computers within a cluster while preserving the owner rights of the machines in the cluster. Common batch systems or local schedulers of this kind include Condor [29] Platform's Load Sharing Facility (LSF) [47] Sun Grid Engine [21], etc. In the current DataMiningGrid test bed we make extensive use of Condor.

Ultimately data is the main 'substrate' of all data mining applications. Typically, electronic data is provided in files, spreadsheet applications, database management systems and other IT systems. Grid mechanisms facilitating the management of data resources are of critical importance to the DataMining-Grid system.

Finally, data mining applications are a fundamental type of software resource in the DataMiningGrid system. We define a *data mining application* as an executable software program, or a collection of such programs, that performs one or more data mining tasks. Data mining applications include executable data mining algorithms, data mining tools, components, libraries, and so on. Enabling this type of resource for operation in grid computing environments facilitates the development of flexible, scalable and distributed data mining applications.

3.2 Globus Toolkit 4

The middleware layer of the DataMiningGrid architecture is concerned with core grid functionality, such as virtual organization management, resource management (discovery, registration, allocation, scheduling), job and execution management, data management, monitoring, security, and so on. In the DataMiningGrid system, these core capabilities, functions and services are provided mainly by the Globus Toolkit 4 software [37]. The Globus Toolkit 4 meets the requirements of OGSA and it implements middleware services adequate for grid applications and the WSRF. Some of the elements of Globus Toolkit 4 that are relevant to the DataMiningGrid system are described below.

3.2.1 Monitoring and Discovery System 4

Globus Toolkit 4's Information Services: Monitoring and Discovery System 4 (MDS4) provides information about the available grid resources and their status. It is used to monitor (e.g., to track usage) and discover (e.g., to assign computing jobs and other tasks) the services and resources in a grid system. To facilitate monitoring and discovery MDS4 has the ability to collect and store information from multiple, distributed information sources. The DataMining-Grid high-level services (in particular the Resource Broker and Information Services) are using MDS4 to implement their functionality.

3.2.2 Data management

The data management components from Globus Toolkit 4, used by the Data-MiningGrid architecture, are GridFTP, the Reliable File Transfer (RFT), and the data services provided by OGSA-DAI [18,5]. In addition, we use the Java CoG Kit [44] for file manipulation on application client machines that do not have a Globus Toolkit 4 installation. GridFTP is a basic platform on which a variety of higher-level functions can be built. The RFT facilitates reliable management of multiple GridFTP transfers. The Globus Toolkit 4 data access and integration tools (OGSA-DAI component) provide grid-enabled access to files, relational databases, and XML databases. None of Globus Toolkit 4's data replication services are used in the data DataMiningGrid, as the data sets to be data mined are either exposed to the grid as OGSA-DAI data resources, or they are uploaded to the grid via the Triana workflow editor.

In the DataMiningGrid system, GridFTP servers are used to receive any data that is introduced to the grid via Triana. Data transfers between grid nodes are orchestrated by the DataMiningGrid Resource Broker service (Section 3.3.1) and are performed using the GridFTP and RFT components (see also Fig. 3). OGSA-DAI is used to provide access to various distributed data resources, in particular to relational databases. The data services it provides can be used to query, update, transform and manipulate data from these resources in various ways. OGSA-DAI provides these functions as activities. Activities are the operations that data service resources can perform on behalf of a client. They expose capabilities of the underlying data resource and are generally used to carry out data resource manipulations or data delivery operations, such as executing SQL or XPath statements, performing XSL-T transformations, and data delivery using GridFTP. The DataMiningGrid activities are extensions to OGSA-DAI. They can be used by DataMiningGrid services to provide basic data preparation and transformation operations relevant to data mining, such as data aggregation, data discretization, computation of derived data, and cross-validation.

Using the APIs provided by OGSA-DAI, with the DataMiningGrid extensions, specialized clients can be constructed to perform data access and transformation, and these clients can be integrated into the Triana workflow editor. Such clients are able to access data from distributed databases and files, integrate these data into a single data set, filter and transform them (e.g., into training and test data sets), and then convert them into the format required by a particular data mining algorithm, such as the attributed-relationship file format (ARFF) of Weka [46].

3.2.3 Addressing and security

The information contained in a service's endpoint reference reveals how it can be contacted by messages. The endpoint reference contains an address property (URL) and reference properties and parameters. A binding directive (binding in short) constitutes how this information is copied to the message and protocol fields that are to be sent to this service. The WS-Addressing specification defines the SOAP binding as the default binding mechanism for WSRF-compliant Web services.

The Grid Security Infrastructure (GSI) was developed by the Globus Alliance for the Globus Toolkit. It enforces security by using a Public-Key-Infrastructure (PKI) implemented in X.509 compliant certificates for authorization [22]. For communication with a grid system built on the Globus Toolkit, so-called 'proxy certificates' are used. These are only valid for fixed periods of time, and are created for a user using the Globus client-side security API. When the client contacts the service, it has to authenticate the user by first passing the proxy. This is implemented inside the Globus API, which is used by the DataMiningGrid clients. GSI supports two levels of security, message-level and transport-level security [45], which differ in concept, performance, and standards conformity. By default, the DataMiningGrid uses transport-level security, the Globus Toolkit 4 default security setting, as this gives the best performance.

3.2.4 Execution management

Globus Toolkit 4's execution management tools handle the initiation, monitoring, management, scheduling, and coordination of remote computations. Globus Toolkit 4 provides the Grid Resource Allocation and Management (GRAM) interface as a basic mechanism for these purposes. A Web services version of the GRAM (the WS-GRAM) supports a suite of Web services with which clients can interact to submit, monitor, and cancel jobs on local or remote computing resources. The DataMiningGrid Services are interacting with services provided by the WS-GRAM to achieve their functionality (Section 3.3).

Enhanced Condor Adapter. The original Globus Toolkit 4 Condor Adapter is designed to submit relatively simple batch programs to Condor pools. It lacks two critical functions that are needed in the DataMiningGrid system for submitting a wide range of programs to Condor pools. First, the current Condor implementation and the Globus Toolkit 4 Condor Adaptor restrict data movement to simple copying of files. Second, despite the sophisticated Java submission environment provided by Condor, it is not possible with the Globus Toolkit 4 Condor Adaptor to submit arbitrary Java jobs to Condor pools. In the DataMiningGrid project we developed two mechanisms to address these shortcomings in the Globus Toolkit 4 Condor Adaptor: to cope with the data movement restrictions, we enabled it to compress recursive directory structures into a single archive file, move the file to the execution machine and extract the archive's content before the actual job execution; to facilitate flexible Java job submission, we made it capable of handling JVM parameters, class path information, and so on.

Enhanced Fork Adapter. Similar to the Globus Toolkit 4 Condor Adaptor, the Globus Toolkit 4 Fork Adaptor does not support flexible execution of Java jobs. The Globus Toolkit 4 Fork Adaptor was modified to handle JVM parameters, class path information, and so on.

3.3 High-level services

The three high-level services of the DataMiningGrid system are represented by the DataMiningGrid Resource Broker, Information Services and Data Services.

3.3.1 Resource Broker

A (grid) *job* could be anything that needs a (grid) resource, e.g., a request for bandwidth or disk space, an application or a set of application programs [31]. In the DataMiningGrid system, jobs consist mainly of one or more DataMiningGrid-enabled data mining applications which need data and processor resources in order to be executed. Based on a grid scheduler component, a grid *resource broker* must make resource selection decisions involving resources spanning over multiple administrative domains (hosts, routers and networks managed by a single administrative authority). In such a distributed environment, a resource broker has no control over the local resources and the information about the resources is often limited or obsolete.

Specifically, a grid resource broker addresses the following tasks [31]:

(1) *Resource discovery.* The resource broker determines the list of available, grid-enabled resources.
(2) *Resource selection.* Based on the description of one or more jobs and their requirements, the resource broker selects those resources that best match the requirements in the job description. To do this, the resource broker matches the job requirements against the resource descriptions provided by an information service (Section 3.3.2 describes the DataMiningGrid Information Services).
(3) *Job preparation and submission.* Before the resource broker submits a job or a collection of jobs to the selected resources, it has to make sure that the resources have all they need to run. In particular, this may involve certain setup activities and data staging or provision.
(4) *Job monitoring and clean-up tasks.* The resource broker needs to inform the user about the progress of the job(s), make sure that the results are available to the user, and initiate possible clean-up operations after the jobs have been completed.

In particular with applications involving multiple resources, users, jobs and sites, resource brokering becomes very complex. Grid resource broker technology is an area of active research [31].

Based on the requirement of the DataMiningGrid system, the specific requirements of the DataMiningGrid Resource Broker [27], are as follows: The Broker needs to (a) facilitate the discovery of data mining applications, (b) accommodate complex workflows (pre-defined or user-defined), (c) identify and select suitable grid resources to execute such workflows, (d) execute such workflows, (e) provide execution monitoring and result retrieval, (f), operate in such a way that the underlying grid details are transparent to the user, (g) should adhere to grid standards (mainly WSRF), and (h) should be able to evolve without requiring users having to install each new version. Critical challenges in this scenario include intelligently matching user and application requests and requirements with available computation and data resources, and performing the automated staging of data.

After careful investigation of several resource brokers (Community Scheduling Framework, GridWay, GridLab Resource Management System, Nimrod/G and GridBus Resource Broker) we found that none satisfied the requirements. However, we found that with some additional development, the GridBus Resource Broker could be made to meet the requirements.

GridBus has the advantage of being able to submit jobs to the Globus Toolkit 4 execution subsystem, it is clearly structured and well designed, and, because it does not require any specific information or security subsystem, it can be easily integrated with other systems.

Currently the GridBus is not service-based, and so it cannot be used for a global grid application as it needs to be installed on every client machine. To address this, a wrapper was developed which exposes its main functions through a pre-defined WSRF-compliant service interface. GridBus also lacked automated resource compilation and MDS4 querying mechanisms. These mechanisms are implemented via the DataMiningGrid Information Services (Section 3.3.2). An illustration of how the different services interact when a DataMiningGrid application is launched, including the job submission steps, is provided in Section 4.4 below.

3.3.2 Information Services

Any comprehensive grid system needs information services to discover, characterize and monitor resources, services and computation [15]. Typically, a grid information service feeds into other grid components and services, including services for discovery, replication, scheduling, troubleshooting, application adaptation, and so on. The DataMiningGrid Information Services include the InformationIntegrator Service, which is responsible for creating the application registry from DataMiningGrid Application Description Schema, and grid information services provided by Globus Toolkit 4. In particular, they provide information on (a) all resources available in a DataMiningGrid grid system, (b) the properties of these resources in terms of storage and CPU capacity, operating system, and so on, and (c) the current load of these resources. The DataMiningGrid Resource Broker requires this information to plan, allocate, carry out, and monitor job execution. The Information Services component automatically queries the MDS4 and compiles a list of the resource entries maintained there. This approach has the following advantages.

First, no user interaction is required for discovering available resources as this is done automatically via MDS4 queries. As a result, the overall system becomes more user-friendly.

Second, the original design of the resource broker facilitates the inclusion of different kinds of schedulers (e.g., based on a cost, round-robin or any other policy). However, in some scenarios requirements may arise with constraints for which the actual scheduler cannot cater, for example, certain confidential data may only be processed on machines belonging to certain organizations. For this purpose it is reasonable to exclude all resources not belonging to the required organizations from the applied scheduler. As the scheduler must select resources from a list automatically compiled by the Information Services, the resource broker can perform a preliminary step of filtering this list before the actual scheduler is called.

3.3.3 Data Services

The DataMiningGrid's data services are OGSA-DAI data services that utilize DataMiningGrid activities for performing data operations on data sets exposed to the grid as data resources. For the use-cases investigated so far, these data sets usually reside in distributed relational databases (Section 3.2.2). Typically the result of these operations will be one or more URIs to pre-processed data sets residing in a file, or a set of files, which the DataMiningGrid Resource Broker then schedules for processing with a data mining algorithm.

3.4 Application clients

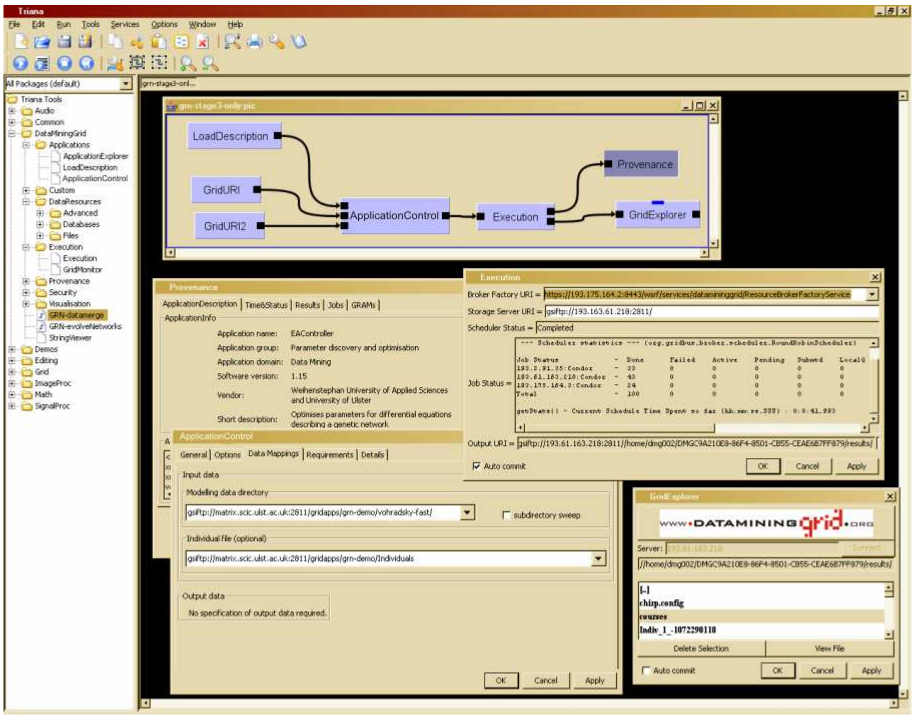
In its current version, the DataMiningGrid architecture comprises two different DataMiningGrid application clients: A workflow editor and manager, and a Web client. Both clients are able to handle multi-component data mining applications. The workflow editor and manager client are designed to facilitate detailed user control of the individual data mining application components, including workflow composition, parameter settings, input and output setting, etc. The DataMiningGrid workflow editor and manager are based on Triana [40]. The Web client is intended for end users (e.g., biologists, medics, customer relationship manager, etc.) with a domain-oriented view of the data mining application. This client is useful for end users with limited knowledge of the underlying data mining and grid technologies.

Both types of clients have an Application Enabler component which is normally operated by a data mining application *developer* and not the *user*. The Application Enabler client components are discussed in Section 4, Grid-enabling data mining applications.

3.4.1 Triana workflow editor and manager client

The workflow editor and manager used to design complex DataMiningGrid application workflows is based on Triana [40]. Triana is an open source problem-solving environment developed at Cardiff University, UK, as part of the Grid-Lab and GridOneD projects. The Triana tool is structured into two major parts: (1) a graphical workflow editor to help users compose workflows from a set of building-blocks that are dragged into a work-space window and connected using a mouse, and (2) a workflow manager or engine to execute workflows.

The diagram in Fig. 2 illustrates a DataMiningGrid application and its presentation via the Triana workflow user interface. Here, the output of the Load Description unit is used to instantiate the Application Control unit with the application description for a data mining application. In this case the application is an evolutionary algorithm and for simplicity, the input data is already located at a URI on the grid. The Application Control unit is used to define



Provenance

ApplicationDescription | Time&Status | Results | Jobs | GRAMs

ApplicationInfo

Application name: EAController
Application group: Parameter discovery and optimisation
Application domain: Data Mining
Software version: 1.15
Vendor: Weihenstephan University of Applied Sciences and University of Ulster
Short description: Optimises parameters for differential equations describing a genetic network

Execution

Broker Factory URI = https://193.175.164.2:8443/wsrf/services/datamininggrid/ResourceBrokerFactoryService
Storage Server URI = gsiftp://193.163.61.218:2811/
Scheduler Status = Completed

Job Status =
```
--- Scheduler statistics --- (org.gridbus.broker.scheduler.RoundRobinScheduler)

Job Status                 -  Done    Failed   Active   Pending   Submd   LocalQ
193.2.91.35:Condor         -  33      0        0        0         0       0
193.61.163.218:Condor      -  43      0        0        0         0       0
193.175.164.2:Condor       -  24      0        0        0         0       0
Total                      -  100     0        0        0         0       0

getStats() - Current Schedule Time Spent so far (hh:mm:ss.SSS) : 0:8:41.993
```
Output URI = gsiftp://193.61.163.218:2811//home/dmg002/DMGC9A210E8-86F4-8501-CB55-CEAE6B7FF879/results/

[✓] Auto commit

ApplicationControl

General | Options | Data Mappings | Requirements | Details

Input data

Modelling data directory: gsiftp://matrix.scic.ulst.ac.uk:2811/gridapps/grn-demo/vohradsky-fast/
[ ] subdirectory sweep

Individual file (optional): gsiftp://matrix.scic.ulst.ac.uk:2811/gridapps/grn-demo/Individuals

Output data

No specification of output data required.

GridExplorer

Server: 193.61.163.218

//home/dmg002/DMGC9A210E8-86F4-8501-CB55-CEAE6B7FF879/results/

[..]
chirp.config
courses
Indiv_1_-1072290118

[ ] Auto commit

Fig. 2. **A DataMiningGrid application composed with the Triana workflow editor user interface**

the options or parameters for the evolutionary algorithm, to map data inputs from the GridURI units to the algorithm's inputs, and finally to define certain execution requirements, such as the operating system and minimum amount of memory required. In this case, the evolutionary algorithm defines 100 jobs, each job representing an independent evolutionary population. The 100 jobs are distributed for execution on different grid nodes, based in Slovenia, Germany and Northern Ireland. The Execution unit shows the execution status of these jobs. The final two units in the workflow, the Provenance Manager and Grid Explorer units, are used (a) to display a record of all information concerning the execution of this evolutionary algorithm application, and (b) to view and download the resulting output data files.

Triana is capable of discovering and binding to Web services and WSRF-compatible services. To develop a user-friendly DataMiningGrid interface, it was necessary to implement the WSRF binding for most DataMiningGrid units. Additionally, all DataMiningGrid units contain bindings for WS-Security and WS-SecureConversation [45]. This permits the passing of the end user proxy credentials to the contacted WSRF service and facilitates secure conversations between the units and the corresponding WSRF services.

Credentials Generator. Security plays an important role in grid computing. As each user requires a proxy certificate, which is a copy of the user's original X.509 [22] certificate with limited lifetime, the workflow editor has to provide an easy-to-use means to create and manage such proxies. The Credentials Generator unit will generate a new proxy certificate. If users possess several certificates (e.g., from different authorities) they can choose which one to use. Furthermore, they can select a suitable private key and specify the lifetime of the proxy (12h, 24h, 1 week, 1 month). Interaction with this unit is only required once per workflow.

Application Explorer. The Application Explorer is a client to the MDS4 service (Section 3.2.1). It is used to search the grid-wide application registry for data mining applications. Search criteria include application ID, application name, group, data mining technique or method, functional area (data mining task such as classification, clustering, etc.), CRISP-DM phase [36], and so on. The parameters are translated into an XPath query which is passed to MDS4. The query results are displayed in a table from which the application developer may select a suitable data mining application. Once the data mining application is selected, MDS4 is contacted again to obtain the full description of this grid-enabled data mining application (Steps 3-8 in Fig. 3). The obtained application description is passed to the next unit in the workflow, which is usually the Application Control unit.

Data Manipulator. The DataMiningGrid Data Manipulator component consists of a collection of units that is divided into three groups. These groups are units that manipulate file systems, relational databases, and a fairly large number of units for expert users that provide direct access to the functionality provided by OGSA-DAI.

The file manipulation units have two main functions. First, they transfer data between the client applications and the grid infrastructure, most importantly this includes uploading data sets from the user's own machine to the grid for data mining, and downloading the data mining results to the user's machine. Second, the units are used to browse file systems and view files on remote machines hosting a GridFTP server. These units are implemented using the Java CoG Kit.

The relational database units are specialized OGSA-DAI clients, providing many functions that are hidden from the user. For example, a user simply enters an SQL query - on execution of the workflow the data is accessed from distributed databases, filtered and transformed, and then delivered to a set of files which are formatted for cross-validation by a particular algorithm.

The output from a Data Manipulator unit will always be a data set (files or directories of files), and one or more URIs pointing to the data. The URIs are subsequently mapped onto application parameters using the Application Control unit.

Application Control. The Application Control unit is a central point of control when developing a workflow for a DataMiningGrid application. In this unit, the end user specifies all application parameters and assigns input data as well as application parameters. The unit takes as input (1) a description of a previously selected grid-enabled data mining application, and (2) any number of data URIs, which may be used as input to the data mining application. The graphical user interface is created dynamically, based on the description of the selected data mining application.

The unit displays general information about the data mining application, a description of the necessary input data (e.g., a file in coded ARFF), and the default settings of the application parameters. Application parameters (also referred to as options) can be of different types, such as strings and integers, as defined by the DataMiningGrid Application Description Schema (Section 4.1). For each application parameter, an input field is presented to the end user. The input field depends on the data type of the application parameter. For example, the value of a decision tree learning parameter A is a real number, with the default value of $A_{def} = 0.43$. In order to execute the application in a distributed way, the end user can also carry out a sweep of values of parameter A for a specified interval, i_A, and step-size, s_A. For example, the interval $i_A = [0.40, 0.45]$ and step-size, $s_A = 0.01$ will result in exactly six job execution requests. Similarly, if the data mining application uses a parameter B, which specifies the class in a decision tree training data file (e.g., iodine) and it belongs to a string data type, the end user can also specify a whole list of strings prompting the system to generate and distribute and execute one job per class in the list.

The input data is denoted by a URI, which may point to either a remote directory containing sub-directories and files (e.g., text documents for ontology-learning) or a single file, which may be a data file (e.g., a data file coded in ARFF) or a file containing some additional parameter settings (e.g., differential equations for an equation discovery learning algorithm). In case of a directory URI, the Application Control unit always allows the end user to specify that a sweep is to be performed over all files in the directory, meaning that there will be as many jobs produced as there are files contained in the directory. For example, the directory may contain a set of task files making it possible that the different tasks can be executed in a distributed fashion, one job per task. The file sweep can only be performed if the application input is of type *file*. If it is of type *directory*, a subdirectory sweep is performed.

The end user can also customize the execution requirements for a particular application. For example, the user may decide to specify the minimum memory (or any other resource capacity) needed for execution. Furthermore, the end user is allowed to specify an exact WS-GRAM service on which all jobs are to be executed. This is especially important when the application has to be moved to a location where the data resides.

The output of the Application Control unit is a fully specified instance of the Application Description Schema (Section 4.1), which represents a multi-job description. This description is passed on to the Execution Manager unit for further processing (starting from Step 9 in Fig. 3; see also Section 3.4.1).

Execution Manager. The Execution Manager provides a Triana user interface client to the WSRF-compliant DataMiningGrid Resource Broker service. This client was built so as to hide the grid complexity from the user. The main Execution Manager provides the user with mechanisms facilitating the (a) submission of jobs, (b) monitoring of jobs, and (c) propagation of the result URIs. The execution of the unit commences when a fully specified data mining application description is received from the Application Control unit. This description is then passed as input to the Resource Broker service for execution of the jobs. After the job execution has been initiated, end users can monitor the execution status of the jobs until all jobs are completed (either successfully or not). Upon completion of all jobs, the unit propagates to the next units in the workflow a URI of the results location and provenance information.

Provenance Manager. The Provenance Manager unit displays all the relevant provenance and metadata and permits these data to be saved for later use on a (local) computer specified by the user. It represents the final operation in a simple DataMiningGrid Triana workflow. The information presented by the Provenance Manager is part of the DataMiningGrid Application Description Schema. This schema includes the following information:

(1) Complete description of the data mining application.
(2) Timestamp recording when the application was submitted to the DataMiningGrid Resource Broker and when it was completed.
(3) Status of the application, in particular whether it completed successfully or failed.
(4) URI specifying the location of the results obtained from a successful execution of the application.
(5) Information about the execution of each individual job of the data mining application. For each job, the following information is available: (a) job ID; (b) job status (success/failure); (c) submission to WS-GRAM time and completion time; (d) failure description (exception code indicating the nature of the failure); and (e) application parameters with which a job started.

Saving the provenance information enables users to search and discover algorithms, training data, and prediction models including scores indicating how these fared. It also permits users to reconstruct the process that has led to a particular result.

Monitor. The Monitor unit provides general grid monitoring capabilities. It displays, in intuitive graphical form, the current status of the grid and its components. For each WS-GRAM in the system the unit presents the WS-GRAM's status: the number of busy CPUs, number of available CPUs and the total number of CPUs. This unit is intended as stand-alone unit and may not be connected with other units in the workflow. The Monitor unit's information is periodically updated to reflect changes in the WS-GRAMS available in the grid environment.

Custom units. Custom units can be used to develop specific data operations, result visualization and other custom functions. These functions can then be added to the data mining process in the usual manner.

Triana Application Enabler. Before a data mining application can be used within the DataMiningGrid, a description of the application needs to be provided. Since this task is concerned with grid-enabling data mining applications, it is described in more detail in Section 4.2.

3.4.2 Web-based client

Similar to the Triana client, the Web client provides a developer and an end user component for grid-enabling and running data mining applications on the DataMiningGrid.

Web Application user client. The Web Application client of the DataMiningGrid system is intended for users who do not want to concern themselves with developing detailed data mining workflows. Instead, the Web Application client permits the running of complex DataMiningGrid applications while hiding many of the underlying complexities of the both the data mining application and the grid from the user. Thus, the user can concentrate on controlling important parameters of the application without needing to know the low-level 'wiring' of the application and the grid.

We have investigated several ways of building Web-based clients for executing DataMiningGrid applications in a grid environment. Generally speaking, Web-based clients are less flexible when compared to a workflow editor like Triana, but can offer a greater degree of user-friendliness for those users who do not require detailed technological control. A user has to specify application information via the Web Application client, and this includes information on the following (see Section 4.3): (1) general information about the application; (2) general execution details; (3) input data requirements; (4) where to store output data; and (5) specific execution requirements.

While it is relatively straightforward to realize a very simple Web-based DataMiningGrid application, the developer effort needed to provide a Web-based *user interface* for complex, multi-step data mining applications can be considerable. In this respect the Triana solution provides more modularity and greater flexibility.

Web Application enabler client. Before a Web client can be used within a grid environment, a description of the associated data mining application needs to be provided. The Web Application Enabler is a DataMiningGrid tool that allows developers to carry out this task through a Web browser. Since this task is concerned with grid-enabling of data mining applications, it is described in more detail in Section 4.3.

4 Grid-enabling data mining applications

In the DataMiningGrid, grid-enabling existing data mining programs is achieved through the use of metadata and the associated Information and Resource Broker Services. This is a generic approach and may be extended to make use of Semantic Grid technologies.

Grid-enabling data mining applications for the DataMiningGrid requires two elements. First, the data mining application and its properties and requirements need to be described in a uniform way and be made known to the system. This description of data mining applications could be viewed as a data mining application resource *abstraction*. Second, the data mining application's software resources need to be registered and stored in the grid. This section and its subsections describe the basic tasks, components, processes and tools involved.

4.1 Application Description Schema

The DataMiningGrid Application Description Schema (ADS) is a central feature of the DataMiningGrid system. It is the basis for registering an application in the grid and providing flexible application search mechanisms, dynamic configuration of GUIs, resource brokering, and so on. The ADS, whose instances are realized as XML documents, describes various aspects about data mining applications that are enabled to run on the DataMiningGrid system.

The ADS is divided into two parts: the first part describes aspects that are *common* to all applications enabled to run on the system; the second part describes information that specifically relates to *data mining* applications. Below we summarize the structure and content of the ADS.

The common part of the ADS captures application-relevant information that falls into three major categories: *general*, *execution* and *application*.

The *general* part specifies different 'general' aspects of the application which will be useful for different purposes (searching for applications, provenance, administration, and so on). The information contains elements like a global unique identifier and a time stamp (both created by the system), a textual description of application, vendor and version information, etc.

The *execution* part contains information relevant to the execution of the application program, such as the underlying programming environment (e.g., Bash Shell or Windows operating system), programming language (e.g., C, Python, Java), executable file(s) and required libraries, commands and arguments used at start-up, directory information, and so on.

The application part is by far the most comprehensive one. It provides configuration information of the application such as

- *Options* or parameters used when the data mining algorithm implemented by the executable is executed. All options or parameters are typed and can be optional (a default may or may not exist and may be overwritten by the user) and hidden (an option is transparent to the user).
- *Parameter lists and loops* define parameter lists and information for executing iterations over loop counters, files or directories. The Application

Control unit uses this part of the ADS (Section 3.4.1). The list element facilitates a 'sweep' over a list of numeric or symbolic values. Such a list may either be provided explicitly by the user, or generated automatically by the system if a repeated execution with a list of different values is required. The loop element is used for 'sweeps' or iterations over a user-defined interval with a fixed step size.

- *Data input and output* slots are used to describe the application's input and output data, i.e., data types (file or directory), transfer protocols permissible for particular data, physical location of data, other descriptors (e.g., whether data input is optional or mandatory), etc.
- The *requirements* slot of the ADS captures the application's system requirements. This information is processed by the DataMiningGrid Resource Broker and used to match applications to resources (Section 3.3.1). Typical entries include requirements for memory, disk space, the type of WS-GRAM job manager (Fork or Condor), optional user-defined IP addresses of execution machines, operating systems and processor architectures (e.g., Motorola PowerPC, Intel i386 and higher).
- *Environment variables* that need to be set at the execution machine before execution of the application.

The data mining part of the ADS describes application information that is specific to the data mining aspect of the application. The content structure of the ADS is mainly based on CRISP-DM [36] and to a lesser extent on the Data Mining Ontology for Grid Programming [10]. This information is used to facilitate fast discovery of data mining applications and data-mining-aware resource brokering. Among other things, this information includes the:

(1) Description (free text) of the data mining application and the domain-

specific problem it is used to address.

(2) CRISP-DM phase the application is used for.

(3) Data mining task to which the application applies.

(4) Data mining method or methodology the application employs.

(5) Data mining algorithm used by the application.

(6) Particular software implementation of the algorithm.

(7) Software suite the application is part of (e.g., Weka, Clementine).

4.2 Application development with the Triana user client

Triana-based application development with the DataMiningGrid system has two aspects: application *deployment* and *use*. *Deployment* is concerned with the grid-enabling of data mining applications for the DataMiningGrid system, and *use* refers to employing applications that have already been grid-enabled for the DataMiningGrid system to solve an actual problem. This approach of deploying and using data mining applications in a grid offers the following advantages. Firstly, all applications are handled by the system in a uniform manner. This ensures that the system can be easily extended and accommodate new applications. Secondly, end users need concern themselves only with information required to find, parameterize and launch applications. Domain-oriented users who do not want to delve too deep into the grid technicalities will find this feature especially appealing.

Deploying a data mining application requires that an *initial* instance of the ADS is created for this particular application. This ADS instance defines all aspects of the application including default options for certain parameters that the user may override later. A DataMiningGrid developer usually performs this task. The developer specifies all required information needed to instantiate the ADS for the particular application. The Application Enabler generates the corresponding initial ADS instance, registers the application details in the MDS4 registry, and uploads and stores all application files (e.g., executables, libraries) in the MDS4 database. A developer (as opposed to the end user) is normally concerned with operating the Application Enabler, and this is reflected in Fig. 1 by the dashed outline of the boxes representing the Application Enablers in both the Triana and Web client case.

To use an already grid-enabled DataMiningGrid application, the user typically identifies the application he or she needs and then specifies the input data, output data location, application options and parameters, and other information required to launch and execute the application in the DataMiningGrid environment. They usually perform these tasks by using the Application Explorer, Application Control, Execution Manager, and Data Manipulator (Section 3.4.1) Triana units.

DataMiningGrid Generic Job Template (GJT) has been developed to support the use of any data mining application enabled for the DataMiningGrid system. This is similar to the work flow shown at the top of Fig. 2. The GJT consists of different Triana units for (a) selecting a data mining application, (b) selecting input data, (c) specifying application options, parameters, and mapping data inputs to the application, and (d) remote execution of the job on the grid.

To locate and select a data mining application registered in a grid, the user employs either the Load Description or the Application Explorer units. These units load the application's ADS description either from the MDS4 or from a local file.

The user has many ways to specify what data the application is to mine, e.g., by uploading a local data file, by executing an application-specific OGSA-DAI client, by selecting a file on the grid, etc. The Grid Explorer and Grid URI units and the Data Manipulator units provide a means to carry out this task, and all ultimately communicate at least one URI to the Application Control unit.

The Application Control unit is used to specify parameters of the application. The output of this unit is a fully specified ADS instance containing all information needed to execute the application. If a parameter sweep or a file sweep is set, then the ADS instance describes the execution of multiple jobs rather than a single execution.

The Execution Manager unit receives the ADS instance and initiates the execution of the application by communicating the ADS instance to the Resource Broker. Finally, the execution results are passed on to a storage server and details about the execution process are passed on to the Provenance Manager.

The Provenance Manager and Grid Explorer units are used to view the results of the application execution and details about the execution process. The user may also chose to store the metadata contained in the Provenance Manager for further use.

4.3 Application development with the Web user client

The basic purpose of the Web-based Application Enabler is the same as the Application Enabler Triana unit. The Web Application Enabler is a Web-based tool used by the developer to define a initial DataMiningGrid ADS instance, register the application in the grid-wide application registry (MDS4), and store the application resources in the system. This facilitates easy discovery and use by both users and developers. The Web version of the Application Enabler consists of several form-based Web pages. These guide the developer through the entire process of creating a DataMiningGrid ADS instance, and uploading and registering the application in the grid. The steps include filling in forms for General Information, Execution Information, Input Data, Output Data, Requirements and Upload, and once these are completed the user may initiate the application's execution with the Start page.

4.4 Application run-time

Once an application is grid-enabled to operate in the DataMiningGrid environment, the system architecture (Fig. 1) comes to 'life' by:

(1) Collecting the necessary information of the data to be processed, the data application to be used, and the available resources.
(2) Matching the application requirements to suitable resources.
(3) Executing the distributed data mining application.

A more detailed account of this process is provided in the diagram of Fig. 3 and the accompanying description.

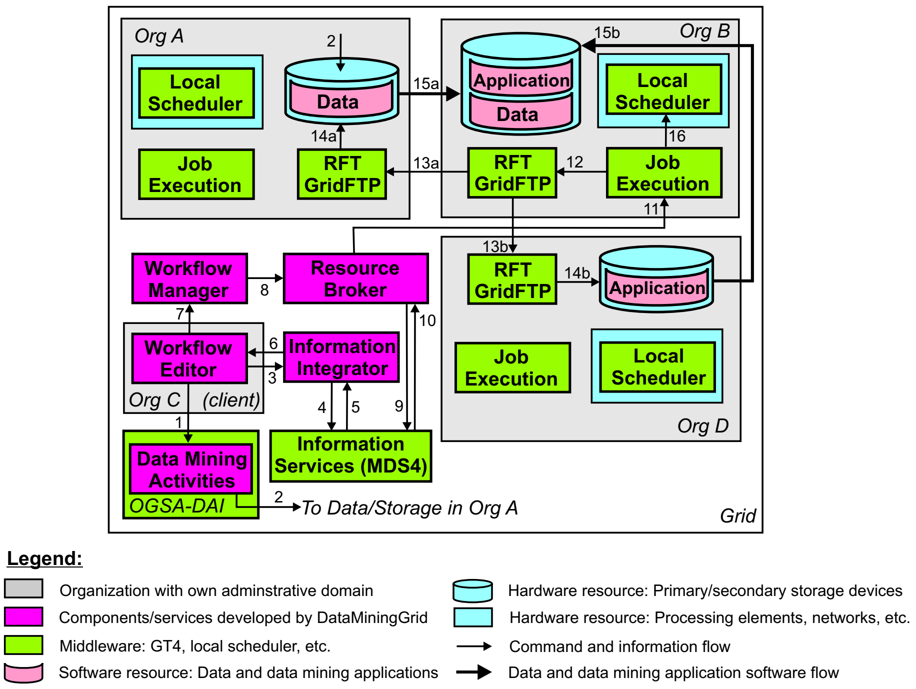

Org A
2
Local Scheduler
Data
14a
Job Execution
RFT GridFTP
15a
13a
15b
Org B
Application
Data
Local Scheduler
16
RFT GridFTP
12
Job Execution
11
13b
Workflow Manager
8
Resource Broker
7
10
RFT GridFTP
14b
Application
Workflow Editor
6
3
Information Integrator
Org C
(client)
Job Execution
Local Scheduler
1
4
5
9
Org D
Data Mining Activities
OGSA-DAI
Information Services (MDS4)
2
To Data/Storage in Org A
Grid
Legend:
Organization with own adminstrative domain
Components/services developed by DataMiningGrid
Middleware: GT4, local scheduler, etc.
Software resource: Data and data mining applications
Hardware resource: Primary/secondary storage devices
Hardware resource: Processing elements, networks, etc.
Command and information flow
Data and data mining application software flow

Fig. 3. **Process view of the DataMiningGrid system**

Fig. 3 depicts a scenario involving four different organizations (labeled *Org A* to *Org D*) from a virtual organization which are part of a grid. In the scenario, data from *Org A* and an application from *Org D* are staged by the system on resources at *Org B* and then executed there. This somewhat simplistic scenario is chosen for reasons of diagrammatic clarity. Clearly, the system supports more complex scenarios involving multiple processes running simultaneously at many different sites involving multiple data and application program resources. The numbers in the diagram denote the different steps involved in this process:

Steps 1-2: Selection, query, extraction and pre-processing of data from *Org A*. This involves the workflow component Data Manipulator (Section 3.4.1).

Steps 3-8: Querying information on available data mining applications. This process extracts the information about the data mining applications as described in the corresponding DataMiningGrid Application Description Schema instance (Section 4.1) and produces the corresponding job descriptions.

Steps 9-10: The Resource Broker queries MDS4's Information Services directly to retrieve up-to-date information on the grid, including available machines, workload statistics, and so on.

Step 11: Passing job descriptions to Globus Toolkit 4's Job Execution Services running on the selected execution machine(s) at *Org B*. In the depicted scenario this is performed only at a single organization. In a different scenario, Globus Toolkit 4 Job Execution Services in several organizations may be called simultaneously. After Step 11, processing splits into two parallel streams denoted by a and b respectively.

Steps 12-15: The Job Execution machine calls RFT/GridFTP components to initiate the transfer of data (prepared by Step 2) from *Org A*, and the files related to the application from *Org D*, to the storage component in *Org B*. The steps labeled 13a, 14a and 15a relate to the transfer of data and the steps 13b, 14b and 15b describe the transfer of application files such as executables, libraries and other files. The optimization of the stage-in process is based on the assumption that data movement is an expensive operation. To minimize data movement, the stage-in process is performed once per GRAM (as opposed once per job) and each job is provided with the complete URL of the local data storage location. Usually, the GRAM is responsible for at least one cluster of machines. The per-GRAM approach has the advantage that the stage-in operation is performed far less than would be necessary in a per-job approach. After successful stage-in of all the executables, data and libraries, the executable is launched, and the execution is monitored until completion.

Step 16: Finally, this step describes the execution request, in which the job description is passed from Globus Toolkit 4's Job Execution Services to the Local Scheduler in organization *Org B*.

5 Summary of main features

This section summarized the main features of the DataMiningGrid system. Some of these features partially overlap or are interdependent to some degree.

Scope of application. The DataMiningGrid system is able to accommodate data mining applications from a wide range of platforms, technologies, application domains and sectors.

Service-oriented architecture. The DataMiningGrid system is designed around SOA principles. The system implements many independent services, each capable of carrying out a set of predefined tasks. The integration of all these services into one system with a variety of general and designated clients, results in a highly modular, reusable, interoperable, scalable and maintainable system.

User interfaces. The DataMiningGrid system provides two types of flexible user interfaces based on a sophisticated workflow technology (Triana [40]) and a Web client. The workflow client facilitates the composition and execution of complex data mining processes by users with comprehensive data mining expertise. The Web client is designed for domain-oriented end users who do not require detailed knowledge of the underlying data mining and grid technology.

Dynamic configuration of graphical user interfaces. The metadata included DataMiningGrid Data Mining Application Description Schema instances allows data miners to operate data mining applications through a dynamically configured graphical user interface, providing slots for specifying public options, algorithm parameters, data inputs, and so on. This frees the developer from the need to develop such interfaces for each and every application.

Searchable applications. The DataMiningGrid's repository of searchable applications, which contains meta-data about each application available at that particular time, provides data miners with a mechanism to search for data mining applications available on the grid. The search can be performed based on different criteria such as general attributes (application name, vendor, version, etc.) or data mining specific attributes (CRISP-DM [36] phase, functional area or data mining task, data mining technique or method). Additionally, users may also specify custom queries using XQuery.

Shipping of applications. The DataMiningGrid system facilitates the 'shipping' of data mining applications to targeted machines on the grid (as determined by either the user manually, or by the DataMiningGrid Resource Broker automatically). This feature has two main advantages: (a) there is no need for pre-installation of applications (this supports a dynamic grid infrastructure), and (b) shipping of algorithms to data (this is useful if data cannot be transferred because of large data volumes, privacy issues, etc.). See also *ease of deployment* below.

Data Mining Application Description Schema. The DataMiningGrid ADS is a central feature of the DataMiningGrid system. Each data mining application is characterized by an instance of the ADS. Particular ADS instances are used for registering applications in a grid, searching applications, creating Triana-based user interfaces dynamically, resource brokering, etc.

Data Services. The DataMiningGrid Data Services are represented by a set of clients based on the OGSA-DAI software [5]. They facilitate integrating distributed databases, 'on-the-fly' data formatting to support data mining operations, performing data assays or summaries of accessed data sets, and the extension of OGSA-DAI services to provide additional data preparation and transformation operations. In addition, remote file browsing and transfer utilities provide easy user access to file-based applications and are implemented using the Java CoG Kit [44].

Ease of deployment. An important feature of the DataMiningGrid system is that it allows users to mine their data without the need to install grid middleware or any data mining software at the location where their data resides. Many users will have data sets that they wish to mine which are not already exposed as data resources on the DataMiningGrid. To use the DataMiningGrid system, a user must have the Triana workflow editor installed and a valid DataMiningGrid certificate. The DataMiningGrid software consists of Java libraries and these, along with important Java libraries from grid middleware (CoG Kit, GridBus, OGSA-DAI and Globus Toolkit, are simply bundled together and copied to a library folder of the Triana workflow editor. These libraries contain the client software that is used to manipulate data resources, select DataMiningGrid applications and orchestrate all the DataMiningGrid services needed to execute these applications.

Resource Broker. The DataMiningGrid Resource Broker is essential to the DataMiningGrid system. It combines the following features: (a) fully automated resource aggregation, (b) data-oriented scheduling, (c) zero-footprint on execution machines, (d) extensive monitoring of resources, (e) brokering based on application requirements typical to data mining (e.g., parameter sweep, data mining task, etc.), (f) support for user-friendly applications, (g) interoperability, and (h) adherence to relevant existing and emerging grid and grid-related standards (e.g., interoperability, security). Extensive performance experiments indicate that the Resource Broker exhibits good speed-up[1] and scale-up[2] behavior [27,23,28].

Standardization. Even in the fast-changing field of grid computing adherence to standards (and de-facto standards) is crucial for building flexible, interoperable and future-proof systems. A critical standard the system supports is WSRF, which has been widely adopted by the grid community. Globus Toolkit 4 is the key middleware upon which the DataMiningGrid system is built. Globus Toolkit 4 implements the WSRF and follows the OGSA standard.

Re-use of (open-source) technology. A critical consideration in developing the DataMiningGrid technology was to ensure that it will be taken up by the industrial and R&D communities and can evolve into the future. To support this goal the DataMiningGrid has been developed based on existing open technology such as Globus Toolkit 4, OGSA-DAI, Triana and GridBus.

[1] Speed-up is typically defined as the ratio of serial execution time of the fastest known serial algorithm (requiring a certain number of basic operations to solve the problem) to the parallel execution time of the chosen algorithm.
[2] Scale-up may be defined as the ability of a greater number of processors to accommodate a proportionally greater workload in a more-or-less constant amount of time.

6 Evaluation

The section presents experiments that were carried out to evaluate the performance of the integrated system components.

6.1 Test bed

To evaluate the DataMiningGrid system, a comprehensive test bed spanning the three European countries UK, Germany and Slovenia was set up. The test bed includes several servers with Globus Toolkit 4 installations and three sites with Condor compute clusters of different composition. The basic test bed hardware configuration was as follows:

- The Grid1 GRAM in Germany: Pentium 1.4 GHz, 1 GB RAM (running middleware).
- The Matrix GRAM installation in the UK, with 64 Itanium II 900 MHz, 2 GB RAM; and 10-25 Pentium, 2 to 3 GHz, 1 GB RAM machines in the condor pool.
- The Grid2 GRAM installation in Germany, with 5 Pentium 1.4 GHz, 1 GB RAM machines in the condor pool.
- The Kanin GRAM installation in Slovenia, with 40 Pentium 1.4 GHz, 512 MB RAM machines in the condor pool.

In addition to performing comprehensive data mining studies based on the main use case scenarios, numerous tests were run using a wide range of application programs. Experiments relating to two main use case scenarios are presented below [43].

6.2 Ecosystem modeling

The increasing pollution of aquatic systems has triggered intensive efforts in the monitoring and management of such systems. These efforts include ecological modeling approaches supported by information technology. Computational ecosystem models allow decision makers to run simulations under different conditions corresponding to alternative environmental policies. In this area decision-support tools from machine learning, data mining and other fields are increasingly employed to model and simulate ecosystems. Because of the underlying complexity (non-linear dynamics phenomena, multiple experiments under different conditions) the necessary calculations are conceptually complex and compute-intensive. In this use case scenario, the DataMining-Grid system was employed to (1) induce models of aquatic ecosystems from modeling knowledge and measurement data, and (2) simulate these models under changing (different) environmental conditions. The main purpose of the developed models was to predict phytoplankton dynamics of two European lakes: Lake Glumsoe (Denmark) and Lake of Bled (Slovenia) [42].

This study is based on Lagramge [41], which is a system for equation discovery. Equation discovery is the area of machine learning concerned with the automated discovery of quantitative laws, expressed in the form of equations, from collections of measured data. Equation discovery methods could be viewed as an extension to system or model identification methods, as they aim to identify both an adequate structure of the equations and appropriate values of the constant parameters. In order to perform complex scenarios with Lagramge, parallelization of the discovery process is very attractive as it can reduce the process time significantly. We implemented the entire procedure of model building using the DataMiningGrid system.

To execute the experiments on the DataMiningGrid system, ecological modeling experts developed two workflows, which were then combined into a (two-step) complex workflow. The first step corresponds to the model *induction* task. It is concerned with discovering an equation structure and model parameters that fit the training data well. The second step corresponds to the *simulation* task. This step calculates the variables representing the ecological system over time under different conditions with the model(s) derived in step one.

The first step (model induction) of the workflow is the one that is computationally demanding. The induced model is expressed in the form of an ordinary differential equation. This part of the workflow interacts with the measurement data, which are files stored on the grid as data resources, and with the model storage directory, in order to store the output results, i.e., the induced models.

The second (model simulation) part of the workflow interacts with (a) the model storage directory (a data resource in the DataMiningGrid system) in order to obtain the previously induced model(s), and (b) with the measurement data (data resource) to obtain the data set defining the initial conditions for simulation as well as with (c) another storage directory in order to store the simulation results, which are then analyzed and interpreted by the domain experts.

In order to test the system, seven test cases were prepared and executed in the test bed. Each test case contains different ecological modeling scenarios, resulting in different number of jobs. For these experiments only 18 out of ca. 120 machines in the test bed were available due to operating system requirements of Lagramge 2.0, which runs only under Linux. The speed-up was evaluated for a workload of 60 jobs, by varying the number of machines to process the jobs. Changing the number of machines from 1 to 10 corresponded to speed-up of 6.70. Furthermore, increasing the number of machines by a factor of 2.50 and the workload by a factor of 2.00 produced a scale-up of 1.80. In addition to speed-up in the model construction process, the ecological modeling experts found the user-friendly access to the data mining tools to be a great benefit.

6.3 Text classification

In customer relationship management and quality management, Daimler-Chrysler maintains distributed repositories with about 15 million text documents, and in the order of 104 new documents are added daily. In this application (a) these documents need to be classified according to subject – this facilitates their redirection to suitable subject matter experts, (b) new emerging subjects need to be discovered, and (c) keywords need to be extracted from documents or groups of documents. It is infeasible to handle and aggregate this information manually or on a centralized server. This application requires the automation of these tasks, handling of their distribution, and features methods for distributed text classification. Because these data are confidential they cannot be described here. Instead, we have applied the same techniques on newswire data available from the German news agency dpa (Deutsche Presse-Agentur).

We performed measurements of runtimes for cross-validation with standard SVM kernels (called SVM-CV) and cross-validation with structured graph kernels (called Hypergraph-CV) [12]. The measurements were run identically on three different GRAM installations with 5 (Grid2 Condor pool), 24 (Kanin Condor pool) and 64 (Matrix condor pool) machines respectively. There was no run which included two or more of the geographically separated GRAMs at the same time. The measurements contain 1, 2, 4, 8, 16, 32, 64 and 128 jobs on each Condor pool with the same input data for SVM-CV and 1, 2, 4, 8, 16 jobs for Hypergraph-CV (this uses unpublished algorithms from Fraunhofer). The Hypergraph-CV measurements were also executed on an input data volume that is 50% of the size of the original data. Each set of jobs was run in two modes:

- *Fork mode* denotes the sequential execution of each job on the GRAM installation machine in question to obtain reference values.
- *Condor mode* denotes the distributed execution of the jobs on the computing pool, which is controlled by the Condor batch job system.

Cross-validation was performed on 1 000 documents of the dpa collection of October 2004. The pre-processing yields approximately 10 000 variables (distinct content words), the exact number depends on the pre-processing parameters. The source XML file of October 2004 is 39 MB, the pre-processed data file is 1.70 MB. The XML data resides at Fraunhofer IAIS, Germany, and was transferred to the remote GRAM machines. Pre-processing was performed locally at the Condor pools.

To determine the performance of a classifier on the given data set, cross-validation consisted of pre-processing documents, training the classifier, and classifying in N identical runs. We executed the runtime measurements on cross-validation for both classifier variants: SVM-CV and Hypergraph-CV.

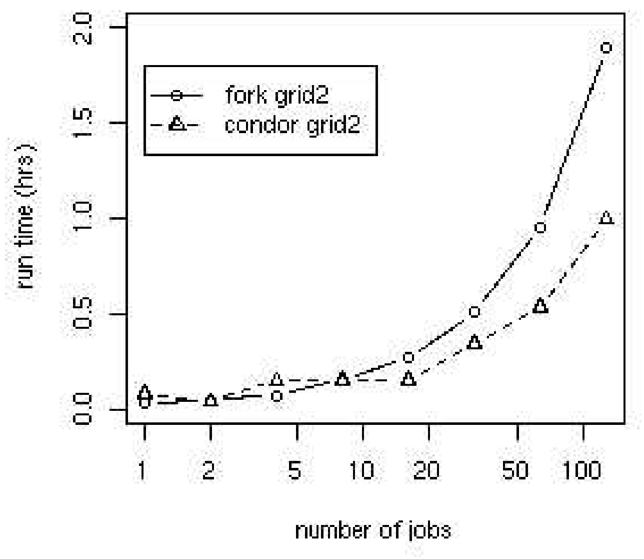


Fig. 4. **Cross-validation: Grid2 GRAM installation**

Speed-up depends on the location of input data, in particular because input files can become rather large. For a central file server with Gigabit Ethernet, we measured a linear speed-up only for up to 5 machines. Unlimited linear speed-up is possible and was measured only if the input files are mirrored on locally installed disks. Fig. 4 to Fig. 6 show the test results of the SVM-CV application.

- *Comparison of Fork / Condor execution:* Both Fork and Condor execution have a linear increase in runtime. Execution on a Condor pool is faster than Fork execution on a single machine so the increase is more moderate.
- *Comparison of Fork execution:* Fork execution is nearly the same at each GRAM installation machine, the Matrix machine is marginally faster, because it has faster hardware.
- *Comparison of Condor execution:* The more machines the pool has the faster is the Condor execution.

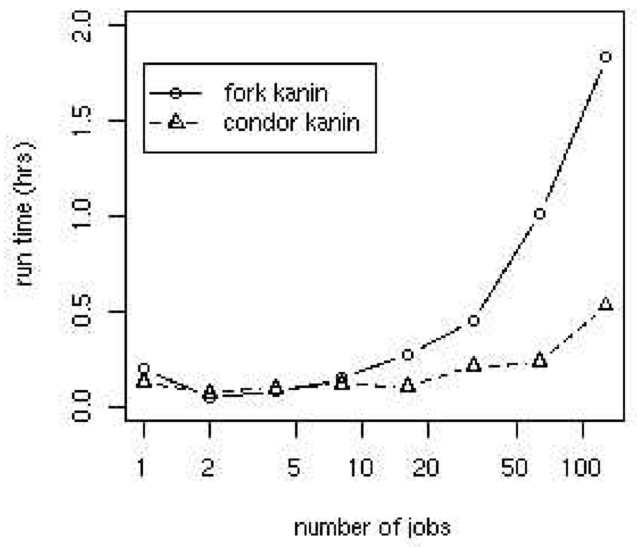


Fig. 5. **Cross-validation: Kanin GRAM installation**

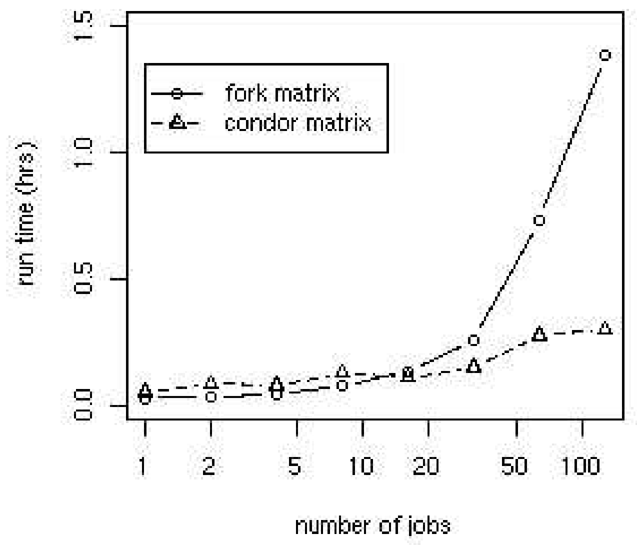


Fig. 6. **Cross-validation: Matrix GRAM installation**

7 Related work

The purpose of this section is to briefly review some of the main efforts in the area of distributed, and in particular grid-enabled, data mining technologies and compare these, as much as this is possible, to the DataMiningGrid system. The criteria used to compare and discuss these systems are listed below, and a summary of the comparison is provided in Table 1. Although relevant and interesting, due to space limitations, this review does not consider work on distributed data mining in peer-to-peer networks, privacy-preserving data mining, distributed data stream mining, data mining in mobile and embedded devices, distributed data mining in sensor networks, or parallel data mining (unless it is related to distributed or grid computing environments).

The following list describes a number of criteria we deemed relevant in discussing and comparing the DataMiningGrid system to related systems:

- *Domain restrictions.* The system is restricted to applications in certain domains.
- *Data mining task/technique restrictions.* The system is restricted to certain data mining tasks (e.g., classification, clustering) or methods or algorithms implementing the tasks (e.g., C5.0, c-means).
- *Data mining technology restrictions.* The system is restricted to certain data mining platforms, tools or implementations (e.g., Weka).
- *Modification of data mining application source.* The system requires modification at source-code level of the data mining application program to be grid-enabled.
- *Data mining standards supported.* Major data mining standards that are directly supported by the system (e.g., CRISP-DM, PMML).
- *Data mining-aware resource brokering.* The system provides a grid resource broker that takes into account requirements and constraints relevant to data mining.
- *Data mining application metadata* The system allows the characterization of data mining application programs or components with computer-readable

metadata. In particular, this metadata should describe the constraints, properties and requirements of the application in terms of data mining dimensions (task, method, implementation) and computing requirements (memory, hardware architecture, software and operating system requirements). The grid resource broker should be able to access and use this information for effective resource brokering.

- *Shipping of data mining application.* The system facilitates the automated shipping of data mining application programs to different locations within a grid.
- *User interface.* The user interface the system provides for different types of users (end user, developer, administrator).
- *Workflow support.* The system provides a means to define and execute complex data mining workflows.
- *Application search.* The system provides a way for the user to identify suitable data mining applications within a grid (based on metadata used to characterize applications).
- *Service-oriented architecture.* The system is based on principles of service-oriented architecture.
- *Distributed data management.* The system facilitates management of distributed data sources, i.e., access to distributed data sources and shipping/staging of data across grid nodes.
- *Grid standards supported.* The grid standards supported by the system.
- *Middleware.* Critical grid middleware the system is based on (e.g., Globus Toolkit, Unicore, etc.).
- *Inherent data distribution.* The system supports direct mining of data which are inherently distributed, i.e., data which cannot be merged or staged on a single grid site or node. Typically, such scenarios exist where data are

Table 1
Comparing major grid-enabling technology with DataMiningGrid.

Feature / System	DataMiningGrid	GridMiner	K-Grid	DMGA/WekaG	SODDM	FAEHIM
Domain restrictions	No	No	No	No	No	No
DM task/technique restriction	No	No	No	Part	HDM	No
DM technology restriction	No	No	No	Part	–	No
Modification of DM application	No	–	No	No	Yes	No
DM application metadata	Yes	Yes	Yes	Yes	No	Yes*
DM standards supported	CRISP	PMML	–	Weka	No	Weka
DM-aware resource brokering	Yes	–	Yes	Yes*	No	Yes*
Shipping of DM application	Yes	–	Yes	No	No	No
User interface	G/W	G	G	G	OBPD	G
Workflow support	Yes	Yes	Yes	Yes*	Yes	Yes
Application search	Yes	–	Yes	Yes*	No	Yes
Service-oriented architecture	Yes	Yes	Yes	Yes	Yes	Yes
Distributed data management	Yes	Yes	Yes	Yes	No	No
Grid standards supported	WSRF	OGSA	WSRF	WSRF	WSBP	WS
Middleware	GT4	GT3	GT4	GT4	No	No
Inherent data distribution	Part	–	Yes	No	HO	No
Fine-grained parallelism	Part	Yes	No	Yes	Yes	Part
Open source	AOSL2	–	No	Weka	No	GPL
Seamless application handling	Yes	–	–	–	–	Yes

partitioned horizontally (a shared set of variables across distributed sets of observations) or vertically (shared set of observations over distributed sets of variables).

- *Fine-grained parallelism.* The system allows portions of a single data mining algorithm instance to be run in parallel on distributed grid nodes.
- *Open source.* Whether the system software is available as open source license (and if yes, what kind of license).
- *Seamless application handling.* Data mining applications can be flexibly added, modified, removed without any changes to user components.

Legend for Table 1: **Yes/No**: Does/does not possess feature/property; **Part**: Possesses feature/property partially; **CRISP**: CRISP-DM: CRoss Industry Standard Process for Data Mining; **WSRF**: Web Services Resource Framework; **OGSA/I**: Open Grid Services Architecture / Infrastructure; **GT2/3/4**: Globus Toolkit 2/3/4; **G/W**: Graphical / Web browser user interface; **WSBP**: Business Process Execution Language for Web Services; **HDM**: Geared towards high-dimensional data; **HO**: Horizontal data partition; **OBPD**: Oracle BPEL Process Designer; **AOSL2/GPL**: Apache Open Source License v2 / Gnu Public License; **CL**: Command line (interface); **WS**: Web services-related standards; **DM**: data mining; *Not implemented; –: Information could not be obtained

GridMiner [8] has been designed to support data mining and online-analytical processing (OLAP) in distributed computing environments. GridMiner implements a number of common data mining algorithms, some as parallel versions, it also supports some text mining tasks. Each data mining service is implemented as standalone grid service specified by OGSA. The system architecture is service-oriented and a Dynamic Service Composition Engine facilitates the integration and execution of services as a workflow. GridMiner provides the integration of distributed databases based on OGSA-DAI. A Knowledge Base component of GridMiner handles all elements of the data mining process, and serves as a service and user registry. The task of bridging the Web and grid environment gap is performed by a Dynamic Service Control Engine component.

From a high-level architecture perspective GridMiner seems to have a lot in common with the DataMiningGrid. Perhaps one of the main advantages of the DataMiningGrid over GridMiner is that DataMiningGrid complies with the more recent trend towards WSRF.

Knowledge Grid (K-Grid) [11], [14] is a high-level, service-oriented framework providing grid-based data mining tools and services. The K-Grid system facilitates a wide range of data mining tasks and related tasks such as data management and knowledge representation. The system architecture is organized into a High-level K-Grid Services and a Core-level K-Grid Services layer, which are built on top of a Basic Grid Services layer. The Core K-Grid Services handle (a) publication and search of data sources and search for mining results, (b) publication and search for data extraction, mining and visualization tools, (c) definition and management of abstract execution plans used to describe complex data mining processes, and (d) presentation of data mining results. The High-level K-Grid Services are responsible for (a) managing metadata describing K-Grid resources, and (b) mapping of the requests described in the abstraction execution plan to available resources, and managing and executing the abstract execution plan. The main components of K-Grid are implemented using the VEGA (Visual Environment for Grid Applications), which is itself based on Globus Toolkit. Recent developments of K-Grid [14] seek to re-implement an earlier pre-WSRF [11] as a WSRF-compliant version.

K-Grid is not too dissimilar to DataMiningGrid, except that DataMiningGrid was conceived from the outset with OGSA and WSRF in mind. Recent K-Grid developments are geared towards WSRF compliance [14]. Unlike DataMining-Grid, K-Grid is not available as open source.

Data Mining Grid Architecture (DMGA) [34] is a flexible data mining grid architecture, based on the main phases of a data mining process: pre-processing, data mining itself and post-processing. This architecture is composed of generic, data grid and specific data mining grid services. WekaG [33] is an implementation of this architecture based on Weka [46], one of the most well known and used data mining tools and Globus Toolkit 4. The main advantages of the DMGA/WekaG combination include: (1) DMGA/WekaG can be adapted to the requirements of complex data mining processes. (2) WekaG is very easy to use for data mining users; the user interfaces of Weka and WekaG are the same. (3) New data mining services can be added in a flexible way. (4) It is possible to use a combination of different data mining services that make use of trading and negotiation protocols for selecting the most suitable service. (5) DMGA provides the composition of services, that is, the ability to create workflows. DMGA offers both horizontal composition, in which different functional services are composed, and vertical composition, in which several instances of the same service access different data sets. (6) Finally, WekaG is able to support parallel implementations of data mining algorithms. Weka is available as open source and WekaG is likely to be open source in the future.

While DMGA is flexible with regard to underlying data mining applications, its WekaG implementation (which is based on Weka) is restricted to applications implemented in WekaG. DataMiningGrid is not restricted to specific data mining applications.

Anteater [24] is a service-oriented architecture for data mining that relies on Web services to achieve extensibility and interoperability. Anteater is designed to handle large volumes of data and high computational demands, and to handle a diverse user population. A particular feature of the Anteater system is its capability to distribute fine-grained parallel data mining applications and to provide high degrees of scalability. Anteater's architecture revolves around a set of *servers*, including the: data server, mining server, application server and visualization server. Anteater uses visual metaphors to present the system's functionality to domain-oriented end users by keeping technical details transparent to them. To exploit parallelism while maintaining performance, Anteater provides a runtime system called Anthill. Anthill requires data mining applications to be decomposed into *filter-stream* subcomponents, which abstract a data mining application into a series of *filters* (computation) and *streams* (communication). Anteater has been evaluated on a number of algorithms and real-world data mining applications.

Anteater requires data mining applications to be converted into a filter-stream structure. While this provides scalability, this overhead cuts down on the number of applications that will actually be ported to this platform. It is not clear to what extent Anteater provides resource brokering. This highlights one of the key features of the DataMiningGrid system – it is very easy to grid-enable existing data mining applications and to seamlessly distribute its execution within a grid computing environments based on the resource allocation and management provided by the DataMiningGrid Resource Broker.

Service-oriented distributed data mining (SODDM): Work by Cheung and colleagues [13] is based on a platform which focuses on services-oriented distributed data mining (we refer to this platform as SODDM). This approach concentrates on real-time, on-demand, self-adaptive, distributed and privacy-preserving mining of inherently distributed data. Privacy preservation is achieved via a learning-from-abstractions approach, which is based on statistical properties of private data rather than individual data items. The overall mining process is performed via a global analysis of local data abstractions. SODDM is based on Web services and the underlying data mining processes are specified in the Business Process Execution Language for Web Services (BPEL4WS also known as WSBPEL. In addition to supporting long-running (stateful) interactions, BPEL4WS provides a model and a grammar for specifying interactions between a business process and its partners through Web services interfaces. The learning-from-abstraction and self-adaptive approach to distributed data mining was demonstrated using different data mining applications, including clustering and manifold unfolding for visualization. To achieve self-adaptation (trade-off between overall data mining quality and source data granularity) SODDM implements a negotiation procedure between the SODDM global service broker and the local data sources (which are endowed with autonomous negotiation properties). Cheung et al. consider service-orientation a critical step towards future distributed data mining since it will help to find better ways to control and manage aspects such as accuracy, privacy, efficiency and resource utilization.

In contrast to DataMiningGrid, which has no restrictions with regard to data mining domains, applications, techniques or technology, SODDM is geared towards data mining applications focusing on high-dimensional data. Also, SODDM seems to have some limitations in terms of resource brokering and other grid-related aspect (see Table 1).

Federated Analysis Environment for Heterogeneous Intelligent Mining (FAEHIM) [4] implements a toolkit for supporting data mining based on a grid services approach. The toolkit consists of data mining grid services and a workflow engine to enable users to compose those services into a solution for a particular problem. Three types of grid services are provided to implement data mining functions: (1) classification, (2) clustering and (3) association rules. The algorithms used to implement the data mining functions are taken from Weka [46]. In addition, visualization grid services based on GnuPlot and Mathematica are also provided to visualize the output of the data mining grid services. Data sets may be read by the grid services from the local file space or streamed from a remote location.

A limitation of FAEHIM seems to be its reliance on Weka. It is not clear if FAEHIM supports any form of intelligent resource brokering. Like DataMiningGrid, FAEHIM uses Triana [40] to define data mining processes.

Platform Independent Text Mining Engine Tool (Pimiento) [3] is an object-oriented application framework (OOAF), i.e., an application 'template', for text mining. Implemented in Java Standard Edition, the Pimiento OOAF gives application developers the primary benefits of an OOAF, such as modularity, reusability, extensibility, and inversion of control. It implements several text mining functions, including text categorization, language identification, clustering and similarity analysis. The overall aims of the Pimiento OOAF platform are to let application developers incorporate text mining functionalities without having to worry about the complexity implicit in a text mining engine, the scalable management of text documents, or access control. Addressing these aims, the Pimiento development was driven by the following key requirements: (1) Interoperability: Based on an open architecture and open application interfaces, the Pimiento system should enable seamless interaction and integration between the application and the text mining platform. (2) Modularity: Applications should continue to function (using the text mining platform via services or components) after changes to the functionality of the platform. (3) Simplicity: Software developers should be able to add text mining functionalities to their applications, without requiring in-depth knowledge on text mining. (4) Sustainability: Existing applications should not be negatively affected when new algorithms, languages and features are added to the platform. (5) Scalability and performance: In particular, the platform should be able to perform and scale well as the amount of analyzed documents increases. (6) Security.

Pimiento is an interesting mining tool with focus on text mining. While this tool supports distributed text mining tasks, it does not seem to facilitate sophisticated resource brokering.

Discovery Net [35] focuses on scientific discovery from high-throughput data generated in life science, geo-hazard and environmental domains. Discovery Net provides a service-oriented computing model for knowledge discovery, allowing users to access and use data analysis software and data sources made available by third parties. The system is based on the Globus Toolkit and provides components (a) to declare the properties of analytical software components and scientific data stores, (b) to retrieve and compose Knowledge Discovery Services, (c) to integrate structured and semi-structured data from different data sources using XML schemas, and (d) to deploy and publish existing knowledge discovery procedures as new services. The system also provides a Discovery Process Markup Language (DPML) which is an XML-based representation of the Discovery Net workflows. Processes created with DPML workflows can be deployed and shared as new services on the grid. Another feature of Discovery Net allows dynamic access and integration of various data sets in the workflows. Interfaces to SQL databases, OGSA-DAI sources, Oracle databases and custom designed wrappers are built to enable data integration.

Discovery Net does not seem to support WSRF. It is not clear to what extent the system supports resource brokering, specifically with sensitivity to data mining applications.

Algorithm Development and Mining (ADaM) [1] consists a wide variety of data mining and image processing components designed for analyzing scientific and remote sensing data. The components can be configured to create customized data mining processes. In its latest version, individual ADaM data mining operations (called toolkits) are available as executables, libraries (e.g., C/C++), modules (Python), and can be accessed via multiple external interfaces facilitating the implementation of data mining and image processing components as Web and grid services.

ADaM focuses on image processing tasks. While the ADaM system supports a wide range of interfaces and a composition mechanism to realize customized data mining processes, its seem to lack a proper workflow editing and management facility and grid resource brokering.

^{my}*Grid* [2] aims to exploit the growing interest in grid technology, with an emphasis on the information grid and bioinformatics. In addition to basic issues of storage, computation and resource management, ^{my}*Grid* is designed to provide higher level functionalities over an existing grid infrastructure that supports scientists in making use of complex distributed resources. In particular, this includes supporting the (a) scientific process of experimental investigation, evidence accumulation, and result assimilation, (b) scientist's use of the community's information, and (c) facilitation of scientific collaboration, allowing dynamic groupings to tackle emergent research problems. ^{my}*Grid* consists of a collection of loosely-coupled middleware components and services designed to support data intensive in silico experiments in biology. Workflows and query specifications link together third party and local resources using Web services technology. Experiment Knowledge Discovery is the part of the in silico life cycle that relates to data mining. The ^{my}*Grid* system includes a sophisticated workflow tool called Taverna Workbench, which allows users to construct complex analysis workflows from components located on both remote and local machines, run these workflows on their own data and visualize the results. The ^{my}*Grid* software can be freely downloaded and has been used for building discovery workflows for investigations into various life science studies. The ^{my}*Grid* project is part of the Open Middleware Infrastructure Institute UK Consortium (OMII-UK). The integration of the ^{my}*Grid* software stack with the production level OMII-UK software promises to increase the commitment of the existing user community, and to encourage a significantly wider deployment of the workflow and semantic grid capabilities that ^{my}*Grid* offers.

In contrast to DataMiningGrid, ^{my}*Grid* focuses on distributed information and knowledge management in the context of the life sciences. Part of the system is devoted to data mining activities. DataMiningGrid is much leaner and concentrates on distributing data mining tasks to enhance performance; to facilitate the sharing of data, data mining applications, and hardware; and to promote resource exploitation. For users and developers interested solely in grid-enabled data mining, DataMiningGrid seems to be more suited than ^{my}*Grid*.

Overall we believe the collection of features of the DataMiningGrid technology make it a unique and competitive contender for grid-enabling existing and developing new data mining applications for grid computing environments.

8 Conclusion

As the demand for automated analysis of large and distributed data grows, new data mining challenges in distributed computing environments emerge.

The aim of DataMiningGrid project is to address some important requirements arising from modern data mining scenarios. The technology developed by the project facilitates grid-enabling of existing, and the development of novel, data mining applications. Major features of the DataMiningGrid technology include high performance, scalability, flexibility, ease of use, conceptual simplicity, compliance with emerging grid (e.g., WSRF) and data mining standards (e.g., CRISP-DM), and use of mainstream grid and open technology. The DataMiningGrid technology has been evaluated on the basis of a wide range of representative modern data mining applications. In comparison with related technologies, the DataMiningGrid system offers an attractive alternative. The software is freely available under the Apache Open Source License version 2 via the project Web site [16] or SourceForge. Future developments of the DataMiningGrid technology include to explore the technology in the context of new applications and to further develop the system accordingly. This may be approached by the DataMiningGrid partners individually or in concert or indeed by anyone who wishes to do so.

Acknowledgements

This work was supported by the European Commission FP6 grant No. 004475. The DataMiningGrid Consortium consists of five organizations: University of Ulster, Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, DaimlerChrysler AG, Technion – Israel Institute of Technology, and University of Ljubljana. Each partner organization contributed 20% to the development of the DataMiningGrid system presented in this paper. We hereby acknowledge the cooperation of all DataMiningGrid partners and collaborators in the DataMiningGrid project. We also thank the following people for their contribution and discussion in the related work section of this paper: Ali Shaikh Ali (Cardiff University, Cardiff), María S. Pérez-Hernández (Universidad Politécnica de Madrid), William K. Cheung (Hong Kong Baptist University, Hong Kong), and Domenico Thalia (Universita' della Calabria, Rende).

References

[1] ADaM Web site.
URL `http://datamining.itsc.uah.edu/adam/`

[2] $^{my}Grid$ Web site.
URL `www.mygrid.org.uk`

[3] J. Adeva, R. Calvo, Text mining with Pimiento, IEEE Internet Computing 10 (4) (2006) 27–35.

[4] A. S. Ali, O. Rana, I. Taylor, Web services composition for distributed data mining, in: Proc of the 2005 IEEE International Conference on Parallel Processing Workshops (ICPPW'05), 2005.

[5] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. C. Hong, B. C. et al., The design and implementation of grid database services in ogsa-dai, Concurrency and Computation: Practice and Experience 17 (2-4) (2005) 357–376.

[6] A. Au, V. Curcin, M. Ghanem, N. Giannadakis, Y. Guo, M. Jafri, M. Osmond, A. Oleynikov, A. Rowe, J. Syed, P. Wendel, Y. Zhang, Why grid-based data mining matters? Fighting natural disasters on the grid: from sars to land slides, in: S. Cox (ed.), UK e-science all-hands meeting, EPSRC, 2004.

[7] M. Berry, G. Linoff, Data Mining Techniques For Marketing, Sales and Customer Support, John Wiley & Sons, Inc., New York, 1997.

[8] P. Brezany, I. Janciak, A. T. AM, GridMiner: A fundamental infrastructure for building intelligent grid systems, in: The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05), 2005.

[9] R. Brito, W. Dubitzky, J. Rodrigues, Protein folding and unfolding simulations: a new challenge for data mining, OMICS: A Journal of Integrative Biology 8 (2) (2004) 153–166.

[10] M. Cannataro, C. Comito, A data mining ontology for grid programming, in: 1st Int. Workshop on Semantics in Peer-to-Peer and Grid Computing, 2003.

[11] M. Cannataro, D. Talia, P. Trunfio, Distributed data mining on the grid, Future Generation Computer Systems 18 (8) (2002) 1101–1112.

[12] C.-C. Chang, D.-J. Lin, LIBSVM: a library for support vector machines (2001). URL `www.csie.ntu.edu.tw/~cjlin/libsvm`

[13] W. Cheung, X.-F. Zhang, Z.-W. Luo, F. Tong, Service-oriented distributed data mining, IEEE Internet Computing 10 (4) (2006) 44–54.

[14] A. Congiusta, D. Talia, P. Trunfio, Distributed data mining services leveraging WSRF, Future Generation Computer Systems 23 (1) (2007) 34–41.

[15] K. Czajkowski, C. Kesselman, S. Fitzgerald, I. Foster, Grid information services for distributed resource sharing, in: Proc 10th IEEE International Symposium on High-Performance Distributed Computing, 2001.

[16] DataMiningGrid Consortium, The DataMiningGrid project Web site (2006).
URL www.DataMiningGrid.org

[17] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, The KDD process for extracting useful knowledge from volumes of data, Communications of the ACM 39 (11) (1996) 27–34.

[18] I. Foster, Globus toolkit version 4: Software for service-oriented systems, in: H. Jin, D. Reed, W. Jiang (eds.), NPC 2005, LNCS 3779, IFIP International Federation for Information, 2005.

[19] I. Foster, C. Kesselman, J. Nick, S. Tuecke, The physiology of the grid: An open grid services architecture for distributed systems integration.
URL www.globus.org/alliance/publications/papers/ogsa.pdf

[20] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, International Journal of High Performance Computing Applications 15 (3) (2001) 200–222.

[21] W. Gentzsch, Sun Grid Engine: Towards creating a compute power grid, in: Proc of First IEEE International Symposium on Cluster Computing and the Grid, 2001.

[22] E. Gerck, Overview of certification systems: X.509, PKIX, CA, PGP and SKIP. do you understand digital certificates? do you understand digital certificates? (2000).
URL www.thebell.net/papers/certover.pdf

[23] A. Grama, A. Gupta, V. Kumar, Isoefficiency function: A scalability metric for

parallel algorithms and architecture, IEEE Parallel and Distributed Technology 1 (3) (1993) 12–21.

[24] D. Guedes, W. J. Meira, R. Ferreira, Anteater: A service-oriented architecture for high-performance data mining, IEEE Internet Computing 10 (4) (2006) 36–43.

[25] D. Hand, H. Mannila, P. Smyth, Principles of Data Mining, MIT Press, Cambridge, MA, 2001.

[26] M. Haynow, Perspectives on grid: A deeper look at schedulers (2006).
URL `www-128.ibm.com/developerworks/grid/library/gr-sked/`

[27] V. Kravtsov, T. Niessen, V. Stankovski, A. Schuster, Service-based resource brokering for grid-based data mining, in: Proc of Int'l Conference on Grid Computing and Applications, 2006.

[28] V. Kumar, A. Gupta, Analyzing scalability of parallel algorithms and architectures, Journal of Parallel and Distributed Computing 22 (3) (1994) 379–391.

[29] M. Litzkow, M. Livny, Experience with the condor distributed batch system, in: Proc IEEE Workshop on Experimental Distributed Systems, 1990.

[30] S. Ma, J. Hellerstein, EventBrowser: A flexible tool for scalable analysis of event data, in: Proc of Distributed Systems, Operations and Management (DSOM), 1999.

[31] J. Nabrzyski, J. Schopf, J. Węglarz (eds.), Grid Resource Management: State of the Art and Future Trends, Kluwer Academic Publishers, Boston, Dordrecht,

London, 2004.

[32] R. Natarajan, R. Sion, T. Phan, A grid-based approach for enterprise-scale data mining, Future Generation Computer Systesm 23 (2007) 48–54.

[33] M. Pérez, A. Sánchez, P. Herrero, V. Robles, J. P. na, Adapting the weka data mining toolkit to a grid based environment, in: AWIC 2005, LNAI 3528, 2005.

[34] M. Pérez, A. Sánchez, V. Robles, P. Herrero, J. P. na, Design and implementation of a data mining grid-aware architecture, Future Generation Computer Systems 23 (1) (2007) 42–47.

[35] S. A. Sairafi, F. S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopolous, M. Osmond, A. Rowe, iJ. Syed, P. Wendel, The design of discovery net: Towards open grid services for knowledge discovery, International Journal of High Performance Computing Applications 17.

[36] C. Shearer, The CRISP-DM model: The new blueprint for data mining, Journal of Data Warehousing 5 (4) (2000) 13–22.

[37] B. Sotomayor, L. Childers, Globus Toolkit 4: Programming Java Services, Moragan Kaufmann, 2006.

[38] V. Stankovski, M. May, J. Franke, A. Schuster, D. McCourt, W. Dubitzky, A service-centric perspective for data mining in complex problem solving environments, in: H. Arabnia, J. Ni (eds.), Proc of Int'l Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04), 2004.

[39] M. Swain, T. Hunniford, J. Mandel, N. Palfreyman, W. Dubitzky, Reverse-engineering gene-regulatory networks using evolutionary algorithms and grid

computing, Journal of Clinical Monitoring and Computing 19 (4-5) (2005) 329–337.

[40] I. Taylor, M. Shields, I. Wang, A. Harrison, The Triana Workflow Environment: Architecture and Applications, in: I. Taylor, E. Deelman, D. Gannon, M. Shields (eds.), Workflows for e-Science, Springer, New York, Secaucus, NJ, USA, 2007, pp. 320–339.

[41] L. Todorovski, S. Džeroski, Declarative bias in equation discovery, in: Proc of Fourteenth International Conference on Machine Learning, 1997.

[42] L. Todorovski, S. Džeroski, B. Kompare, Modelling and prediction of phytoplankton growth with equation discovery, Ecologial Modelling 113 (1998) 71–81.

[43] J. Trnkoczy, Ẑ. Turk, V. Stankovski, A grid-based architecture for personalized federation of digital libraries, Library Collections, Acquisitions, and Technical Services 30 (2006) 139–153.

[44] J. von Laszewski, I. Foster, CoG Kits: A bridge between commodity distributed computing and high-performance grids, a Java commodity grid kit, in: ACM 2000 Java Grande Conference, 2000.

[45] V. Welch, Globus toolkit version 4 grid security infrastructure: A standards perspective.
URL `www-unix.globus.org/toolkit/docs/development/4.0-drafts/security/GT4-GSI-Overview.pdf`

[46] I. Witten, E. Frank, Practical machine learning tools and techniques. 2nd

Edition, Morgan Kaufmann, 2005.

[47] S. Zhou, LSF: load sharing in large-scale heterogeneous distributed systems, in: Proc of the Workshop on Cluster Computing, 1992.

[48] H. Zhuge, Semantics, resource and grid, Future Generation Computer Systems 20 (1) (2004) 1–5.

Short Biographies

Vlado Stankovski

Mr Vlado Stankovski, M.Sc., is researcher at the Department of Civil Informatics at the Faculty of Civil and Geodetic Engineering of the University of Ljubljana. His past and current research interests include the application of machine learning techniques and evolving Internet technologies to engineering and medical problems.

Martin Swain

Dr Martin Swain received an M.Phys degree in Physics from the University of Manchester in 1996, and MSc. and Ph.D. degrees in intelligent computing systems and bioinformatics from the University of Aberdeen in 1997 and 2001 respectively. His research interests are in biophysics, systems biology, data management and grid computing.

Valentin Kravtsov

Mr Valentin Kravtsov received his Bachelor degree in Software Engineering at the Technion – Israel Institute of Technology. His M.Sc. research topics include grid technologies and distributed and parallel computing.

Thomas Niessen

Mr Thomas Niessen received his Master Degree in Computer Science at the University of Applied Sciences Bonn-Rhein-Sieg in 2005. His research interests include distributed computing, especially grid technology, data mining, and databases.

Dennis Wegener

Mr Dennis Wegener works as research fellow at the Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Department Knowledge Discovery, in Bonn, Germany. His research interests include data mining and grid computing.

Jörg Kindermann

Dr. Jörg Kindermann is a mathematician and linguist. After graduation at the University of Bielefeld he worked at the German National Research Center for Computer Science. Since 2001 he is a senior research scientist at the Fraunhofer Institute IAIS. His research interests are text and multimedia mining, high performance computing, and statistical learning algorithms.

Werner Dubitzky

Prof. Werner Dubitzky holds a Chair in Bioinformatics and is Head of the Systems Biology Research Group at the University of Ulster. His research interests include bioinformatics, systems biology, data and text mining, artificial intelligence and grid technology.

Fig. 7. **Vlado Stankovski**

Fig. 8. **Martin Swain**

Fig. 9. **Valentin Kravtsov**

Fig. 10. **Thomas Niessen**

Fig. 11. **Dennis Wegner**

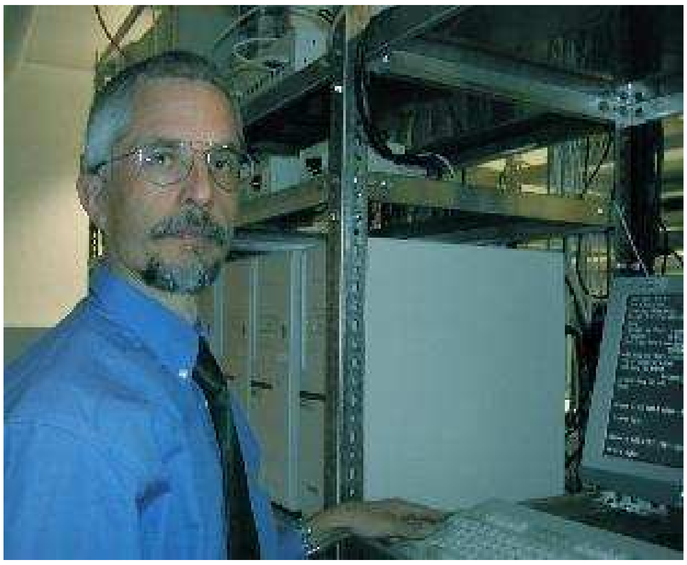


Fig. 12. **Joerg Kindermann**

Fig. 13. **Werner Dubitzky**