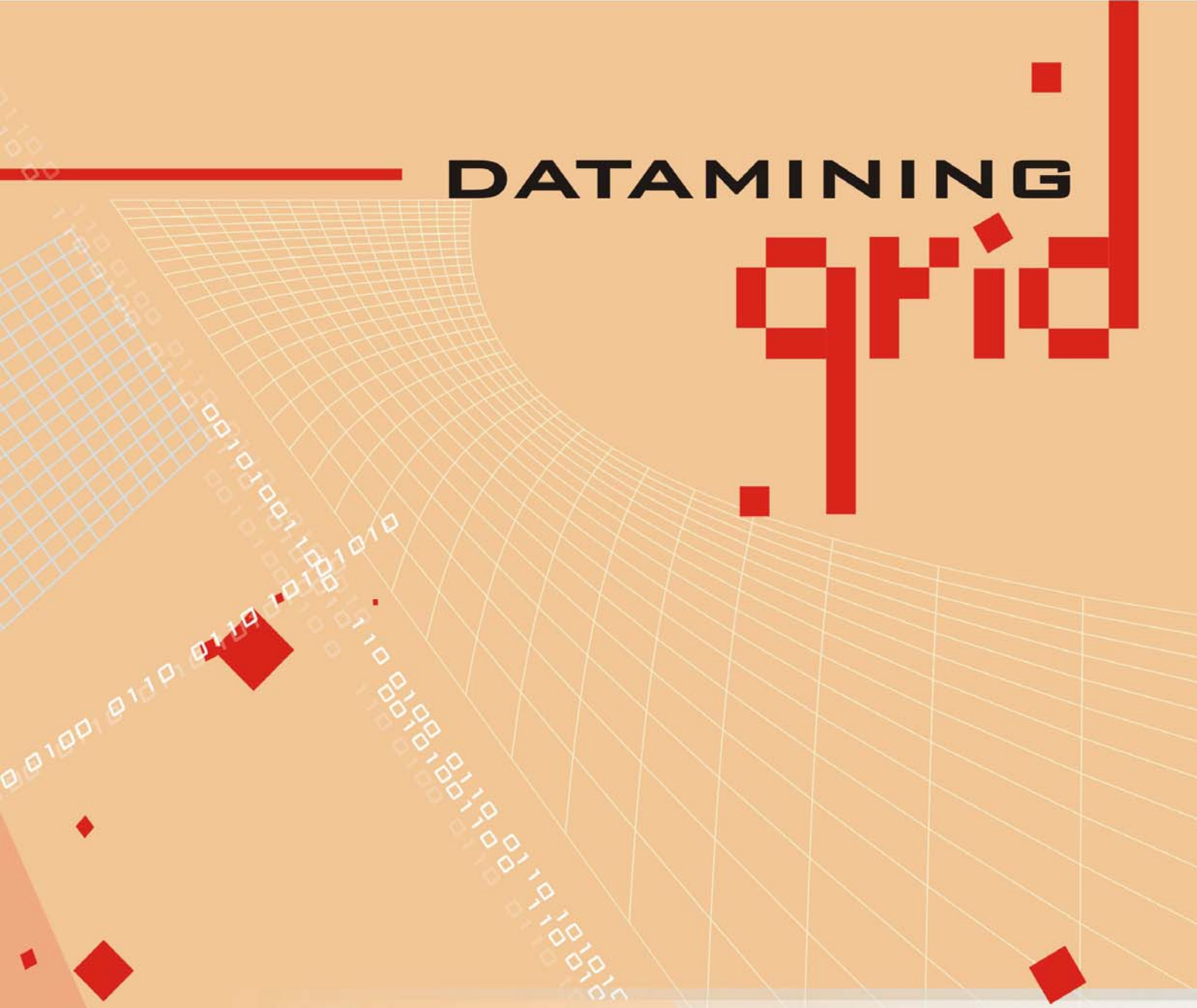

Deliverable D11(1): Common requirements analysis, specification and evaluation of DataMiningGrid interfaces and services

DATA MINING TOOLS AND SERVICES FOR GRID COMPUTING ENVIRONMENTS

Deliverable D11(1): Common requirements analysis, specification and evaluation of DataMiningGrid interfaces and services

Responsible author(s): Vlado Stankovski, Jernej Trnkoczy
Co-author(s): Markus Ackermann, Nataša Atanasova, Werner Dubitzky, Jürgen Franke, Thomas Hunniford, Jörg Kindermann, Boris Kompare, Nahum Korda, Valentin Kravtsov, Michael May, Nataša Fidler Mis, Thomas Niessen, Gerhard Paaß, Matthias Röhm, Assaf Schuster, Martin Swain, Ran Wolff

Project funded by the European Community under the "Information Society Technology" Programme Contract IST-2004-004475

Revision history

Deliverable administration and summary		
Project acronym: DataMiningGrid		**ID:** IST-2004-004475
Document identifier:	DataMiningGrid-del-D11(1)-s-v14	
Leading Partner: LJU in collaboration with all Partners		
Report version: 14		
Report preparation date: 30 November 2004		
Classification: Public		
Nature: Report		
Author(s) and contributors: Vlado Stankovski, Jernej Trnkoczy (LJU) in collaboration with all Partners		
Status:		Plan
		Draft
		Working
		Final
	X	Submitted
		Approved

The DataMiningGrid © Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

Date	Edited by	Status	Changes made
-	DoW	Plan	"a report template will be defined that all reports will follow"
03.10.2004	jt	draft	Original proposal - formatting
12.10.2004	vs	draft	Added contents
14.10.2004	jt	draft	Added data in headers and footers
19.11.2004	jt	draft	Modified sections, added contents
22.11.2004	vs	working	Release for Partner comments
24.11.2004	jt	working	Partner comments included
28.11.2004	vs	working	Release for final comments
30.11.2004	jt	final	Final comments included
30.11.2004	vs	final	Checked for consistency, finalized
30.11.2004	vs	final	Approved by Quality Manager
30.11.2004	wd	submitted	Submitted by Project Co-ordinator

Note that other documents may supersede this document. A list of newest public DataMiningGrid deliverables can be found at the http://www.DataMiningGrid.org/dissemination .

Deliverable D11 shall be released in Months 3, 9 and 24 so the deliverable numbers shall be D11(1), D11(2) and D11(3)).

Report version is the version of the document – that is the number ranging from 01 to 99. The version starts at the beginning (number 01) when the document status changes.

Classification of the document can be of following: PU (Public), PP (Restricted to other program participants), RE (Restricted to a group specified by the Consortium) or CO (Confidential, only for members of the consortium). Deliverable classification for each deliverable is defined in DoW.

Nature of the deliverable is R(Report), P(Prototype), D(Demonstrator), O(Other) and is defined in DoW for each deliverable.

Status of a document can be: Draft (issued for contributions from Partners), Working (issued for comments), Final (approved by deliverable quality manager= person responsible for that particular deliverable), Submitted (approved by project quality manager and submitted to European Commission), Approved (approved by European Commission).

Copyright

Citation

Vlado Stankovski, Jernej Trnkoczy (2004) et al. Deliverable D11(1). DataMiningGrid© Consortium, University of Ljubljana, www.DataMiningGrid.org

Acknowledgements

The work presented in this document has been conducted in the context of the project **IST 2004 004475 DataMiningGrid.** DataMiningGrid is a 24-month project that started on September 1st, 2004 and is funded by the European Commission as well as by the industrial Partners. Their support is appreciated.

The Partners in the project are University of Ulster (UU), Fraunhofer Institute for Autonomous Intelligent Systems (FHG), DaimlerChrysler (DC), Israel Institute of Technology (TECHNION) and University of Ljubljana (LJU). The content of this document is the result of extensive discussions within the DataMiningGrid© Consortium as a whole. This report owes to a collaborative effort of the above organizations.

More information

Public DataMiningGrid reports are available through DataMiningGrid public web site www.DataMiningGrid.org.

Executive summary

This document specifies technical requirements and the goals for the development of DataMiningGrid tools and services. It also specifies possible usage scenarios with the purpose of establishing more detailed technical requirements[1]. The individual parts of deliverable D11 are collected in a single, coherent requirements document that will be a major input for the system and component design.

The Partners identified and elaborated DataMiningGrid requirements in the following areas:

- Identifying (locating) DataMiningGrid resources by using metadata,
- Accessing and selecting subsets of data,
- Data transfer,
- Data (pre-) processing,
- Data mining tasks,
- Text mining and ontology learning,
- Workflow editing and submission,
- Data privacy, security and governance,
- Integration of domain knowledge,
- Grid infrastructure and middleware functionality,
- Usability, response times and user-friendliness.

All individual tasks relating to requirements analysis and specification have contributed to this joint deliverable; therefore, D11 is the product of the following tasks:

- Task T12: Common requirements specification;
- Task T21: Requirements analysis for data access, transfer, and manipulation and
- Tasks T61-T64, that is demonstration tasks contributed by all Partners.

D11 has been composed from contributions by all Partners.

[1] End-users specific requirements will be elaborated in deliverable D61.

Table of contents

1 Introduction

The need for data mining in grid computing environments is motivated by describing several representative use cases. Some of these use cases are based on efforts currently under way in industry and academia, others demonstrate more long-term possibilities.

Section 3: Requirements presents a set of features that should be present and gives motivations for those features. Section 4: Objectives – Research Goals Beyond the Scope of the DataMiningGrid Project describes a list of features that might be useful for many use cases, but may not necessarily be addressed by the DataMiningGrid project.

The detailed specification of the DataMiningGrid interfaces will take into consideration:

- The requirements that are contained in this document,
- Review comments on this document from public feedback, invited experts and working group members,
- Specifications of (or proposals for) grid data mining tools and services that meet many of these requirements,
- DataMiningGrid requirements recognized by other projects from the concertation framework of the same strategic objective “Grids for Complex Problem Solving”, such as the SIMDAT, inteliGrid and other.

1.1 Grid computing and data grid systems

Grid computing could be viewed as a generic enabling technology for distributed computing. It is based on a hardware and software infrastructure that provides dependable, consistent, pervasive and inexpensive access to computing resources anywhere and anytime. In their basic form, these resources provide raw compute power (CPU cycles) and massive storage capacity (magnetic disk or other mass storage devices) [Fos99]. These two Grid dimensions were originally dubbed Computational Grid and Data Grid, respectively. However, since the inception of Grid technology [Bax02], the term resource has evolved to cover a wide spectrum of concepts, including “physical resources (computation, communication, storage), informational resources (databases, archives, instruments), individuals (people and the expertise they represent), capabilities (software packages, brokering and scheduling services) and frameworks for access and control of these resources (OGSA - Open Grid Services Architecture, The Semantic Web)” [Fos02]. Using a Grid to share resources, researchers and small enterprises can gain access to resources they cannot afford otherwise. Research institutes, on the other hand, can leverage their investment in research facilities by making them available to many more scientists.

Initially, the research community’s focus was placed on Computational Grids. These have today reached maturity. Toolkits such as Globus [Globus04], UNICORE [Rom02], Condor [Condor04], and AVAKI [Avaki04] offer a wide range of services starting from job management, data transfer (for input and output),

and various security primitives such as authorization and secure channels. Built on top Globus services, a number of Data Grid projects were initiated about three years ago. The most notable of these is the EU-funded DataGrid project (initiated 2001) [DataGrid04].

Current Data Grid systems offer good solutions for file-based data access and data management problems: they allow a client to locate data, select data, takes care of data replication, etc. However, many scientific and commercial applications are highly dependent on data stored in more complex database management systems, providing more sophisticated access and processing of data. Therefore, recent research has been focusing on (semantic) Grid database access and integration services [GGF04], notably the DAIS standard (see below). These developments and the ever-increasing need to exploit the growing amounts of data across many sectors, give now rise to the development of generic Grid infrastructure (protocols, services, systems, tools) facilitating the automated analysis and interpretation of large and inherently distributed data. However, distributed, Grid-enabled environments imply new levels of complexity along conceptual, technical, legal, ethical, and other dimensions (see section 3 on research and development challenges).

1.2 State of the art in grid computing

In many of today's distributed computing applications, the idea of a stateful (as opposed to stateless) process is extremely important. Data values, and the results of operations on those values, must persist (i.e. maintain their informational state over certain periods of time). Given the inherent heterogeneity of modern platforms, maintaining state is a non-trivial concept, resulting in the development of standards in the area. The idea of web services has emerged to address heterogeneous distributed computing based on existing Internet-based standards (e.g. XML)[Fos04]. Based on this concept, the Open Grid Services Architecture (OGSA) [Fos02] was developed to provide a consistent manner for describing services in Grid Environments. The Open Grid Services Infrastructure (OGSI) is one implementation of this architecture, which is now being reworked under the Web Services Resource Framework. Computational resources, storage resources, databases and programs can all be considered as services. Typically, the development of a standard in any computer application area is two-fold. It involves

1. The definition of service interfaces and
2. The definition of protocols to invoke these interfaces.

In the area of data management the Data Access and Integration Services (DAIS) [GGF04] working group, as part of the Global Grid Forum, aims to develop and promote standards for accessing data across Grids with an emphasis on existing systems. OGSA-DAI is a collaborative program developed with the aim of meeting the needs of the UK e-Science community by producing an open source database access and an integration middleware for Grid applications. While it developed separately from DAIS, it aims to be fully compliant with the standards developed by them. As data access in data mining applications is of

fundamental importance the output from DAIS and OGSA-DAI are relevant to all data mining services that will be developed under the DataMiningGrid project.

Although the OGSI/OGSA standards have helped to provide grid-based services, several criticisms have been made with regards to its usability. Perhaps one of the most significant criticism is that it is too object-oriented and in violation of the "pure" the web services concept (on which it was originally based). Pure web services are supposed to have no associated state or instances. As a result, the OGSI architecture is moving towards the Web Services Resource Framework (WSRF) [Czaj04]. This framework, initially inspired by OGSI, basically restructures OGSI concepts to fit more closely with the current web services architecture. Due to fundamental changes in the exchange of messages between the two approaches, WSRF-compliant services will not interoperate with OGSI services.

Much effort is made by both the scientific and business communities to develop standard grid middleware, allowing for easy development of grid-enabled applications and ensuring their interoperability. However with the emergence of the OGSA standards, calling for the web services-based approach, there is much confusion and feeling of uncertainty among the grid researches and developers.

The Globus Toolkit 3.0, an OGSI based descendant of the widely adopted Globus Toolkit 2.x, required significant refactoring of the working grid environments. However recent dramatic changes of OGSI towards further integration with Web Services (WSRF) render the previous work unusable. Such unexpected change in the core of the grid middleware makes it unclear how the latest developments will impact other standards and the whole standardization process at GGF. Currently, the grid interoperability standard is WS-I+.

One of the tough challenges faced by any grid-related project today is to maintain a consistent picture of the latest developments and to comply with the constantly evolving new standards. "gridification" of services will need to take onboard and adhere to existing and emerging grid data mining standards, such as those discussed in the next section.

1.3 Research and development challenges

As a result of the unprecedented technological advances in recent years, fundamentally new, intrinsically distributed complex problem solving environments are emerging. These developments are prompting a range of new data mining research and development problems.

These can be classified into the following broad challenges:

- **Distributed data**. The data to be mined are increasingly stored in distributed computing environments on heterogeneous platforms. Consequently, development of algorithms, tools, and services is required that facilitate the mining of inherently distributed data within complex problem solving environments.

- **Distributed operations**. Emerging grid computing environments will see more and more data mining operations and algorithms be made available on the grid. To facilitate a seamless integration of these resources into distributed data mining systems for complex problem solving, novel algorithms, tools, grid services and other IT infrastructure need to be developed.
- **Massive data**. Development of algorithms and systems for mining large, massive and high-dimensional data sets (out-of-memory, parallel, and distributed algorithms) is needed, as more and more data mining problems in complex problem solving environments are faced with giga- and tera-scale data sets.
- **Data types**. The phenomena analysed in complex problem solving scenarios are captured in increasingly complex data sources, structures, and types, including natural language text, images, time series, multi-relational and object data types. grid-enabled mining these data will require the development of new methodologies, algorithms, tools, and grid services.
- **Data privacy, security, and governance**. Automated data mining within distributed computing data Grids raises serious issues in terms of data privacy, security, and governance. Related issue involve ethical and legal aspects, good practices, and audits. grid-based data mining technology will need to support researchers and analysts in using the data mining tools and services in a legal, ethical, secure, privacy-preserving, and auditable way.
- **Domain knowledge**. The data to be mined in complex problem solving scenarios will increasingly require the integration of existing domain knowledge into the mining process, particularly in knowledge-intensive domains. This knowledge is often dispersed across geographically distributed sites and organisations, either in digital form (ontologies, metadata, knowledge-based or simulation systems) or provided interactively by human experts. grid-enabled data mining technology will need to provide services and tools to support knowledge-aided data mining. (see also last bullet point).
- **User-friendliness**. Ultimately, data mining in a distributed grid computing environment must hide technological grid details from the user. To facilitate this, new software, tools, and infrastructure development is needed in the areas of grid-supported workflow management, resource identification, allocation, and scheduling, and user interfaces.
- **Resource identification and metadata**. A key characteristic of emerging grid-based complex data mining environments is that data sets, data mining operations and relevant domain knowledge resources will be dispersed around a potentially very large number of sites and nodes. Powerful metadata services resource identification tools and systems are required to navigate, select, and use the available resources effectively.

1.4 DataMiningGrid underlying philosophy

The underlying philosophy (approach) of the project is depicted in Figure 1. The Consortium decided to use a so-called “three legged stool” methodology for

project development process. This methodology basically consists of three main concepts:

- Use-case-driven process – all the stages in the process are driven by use cases,
- Architecture-centric – architecture has to be outlined at a very early stage of the project and it evolves over time,
- Iterative process of improvements of the various models (use case model, requirements model, analysis model, and design model). In the DataMiningGrid project we improve these models at least in three iterations.

The requirements are chosen based on the aspects of the use cases that the Consortium considered most important. As such, one should not assume that that the DataMiningGrid components will directly support every aspect of the use cases.

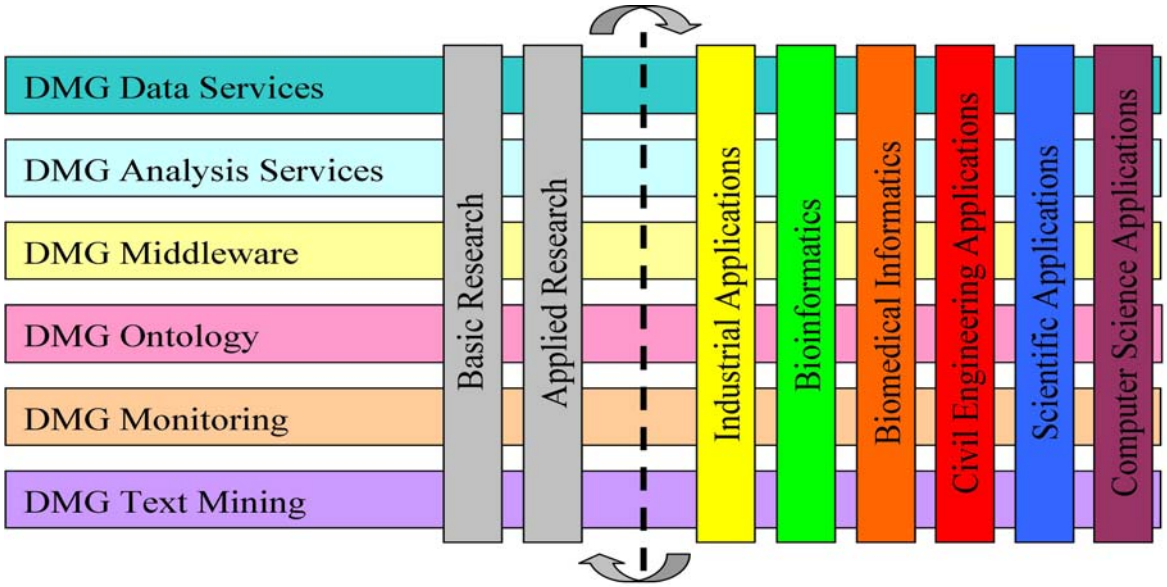

Figure 1: DataMiningGrid underlying philosophy

2 Representative use cases

DataMiningGrid can be used to improve existing data mining techniques and may enable new uses of data mining on the grid. In this section we describe some representative use cases. Note that this is not an exhaustive list, but instead a cross-section of interesting use cases. These use cases will serve as a guideline in choosing requirements for the DataMiningGrid tools and services.

2.1 Genetic algorithms for gene regulatory reengineering

A genetic algorithm is used to model a natural phenomenon; gene regulatory networks will be simulated in silico on the grid in order to discover the genetic configuration that reproduces observations made by in vivo biological experiments.

Many different genetic networks will need to be computationally simulated, and the genetic algorithm will be used to design and evolve these network simulations so that the optimum network configuration is discovered. Here the optimum network is the network that best reproduces the observations generated by a real, biological genetic network. Having access to large computational resources will be essential for this demonstrator, as the search space is vast, and the more simulations that can be run, the better will be the search through the possible genetic configurations.

Each genetic network simulation can be run independently of the others, and so a batch scheduling system such as Condor, that can access the computational power of idle machines, would be excellent for this use case. The simulations can be distributed over a cluster of machines running Condor, and the data produced by the simulations will need to be collected and processed by the genetic algorithm, which will then evolve new simulations based on the previous simulations. Finally, the optimum genetic network configuration will be presented to the user.

2.2 Information integration of life science data: an integrated approach to protein subcellular localization prediction

To investigate natural phenomenon from the bioinformatics area there is great need to access data in multiple databases, which all have significant syntactic, technological and semantic heterogeneity. DataMiningGrid mechanisms will be used to facilitate this process, and in particular they will be used to integrate two databases containing information that can be used to predict proteins' locations within a cell's subcellular compartments.

In this use case the main challenge for the DataMiningGrid will be integrating heterogeneous data sources for data mining. To allow for effective data mining, the data has to be stored and adequately organized; this task will be addressed by means of data warehouse technology. Data integration using warehousing technology has already been investigated by UU in the context of this research

project [Stur03]. Here we will be mostly concerned with semantic heterogeneity as our biological databases will use different conceptualizations or ontologies to model data.

Once the biological databases have been integrated, a classification algorithm will be applied, in this case probably a decision tree, to datasets derived from these databases.

2.3 Text mining use cases

The text mining use cases described under T61 of the Technical Annex can be subdivided into the following text mining framework, which is shown in the picture below (Figure 2. Text mining framework). Three different scenarios are sketched for a huge (and distributed) collection of electronic texts.

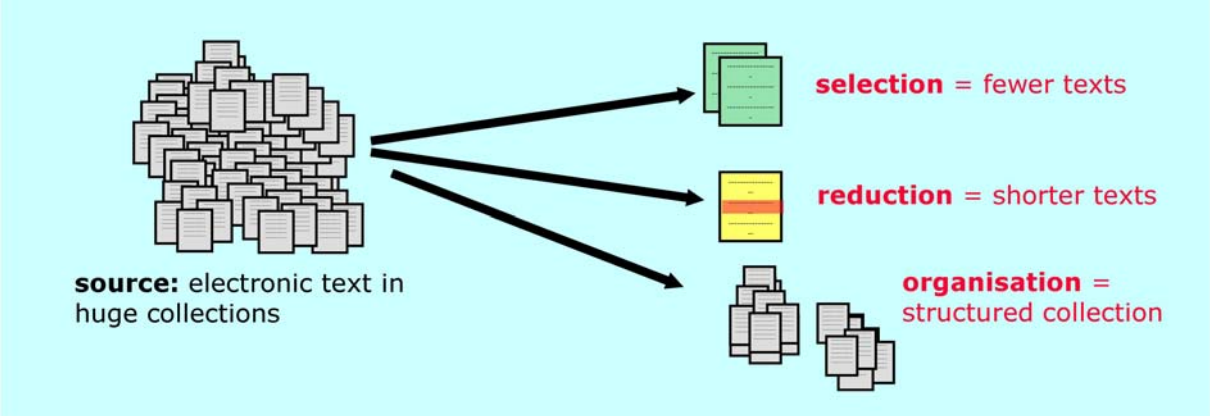

Figure 2. Text mining framework

- Selection (T61.3) – The typical application of search engines to find the relevant texts for a query, i.e. to select the appropriate text(s).
- Reduction (T61.3) – Finding the relevant passages in documents which contain the answer to a query.
- Organisation (T61.1) – The building of problem relevant subsets for the whole collection. Structuring the document collection can be done in two different ways:
  - Classification – The structuring process is done due to a given categorization scheme and can be done by rules or training by examples from a given learning set. So the principle structure is given by a natural categorization scheme or by the human trainer of the system.
  - Clustering – The organization process is done automatically to find unknown structures in the collection and the main purpose is to detect the unknown categories of a collection.

In addition to all basic text mining processing steps the subtask **Ontology learning from and for text collections (T61.2)** will focus on an important processing unit for which the grid-enabled version will be very valuable. This subtask is not a demonstrator for its own, because the usefulness of this

demonstrator is not self-contained, but is an essential subtask of the other demonstrators. The learned ontology will support the other subtasks by:

- Improvement of the document representation,
- Improvement of class- and sub-class structure,
  - Improved classification and clustering (T61.1),
- Improved representation and annotation of documents,
- Query expansion,
- Enhance passage retrieval,
  - Improved retrieval (T61.3).

Specific text mining use cases are therefore as follows:

- Fast distributed text classification for quality management,
- Ontology Learning from and for text collections,
- Finding related and similar documents in the intranet,
- Mining digital libraries,
- Mining civil engineering databases,
- Mining old church Slavic documents.

The use cases displayed under T61 of the Technical Annex provide the building blocks needed by the text mining use cases T61.1 – T61.3. They deal with the pre-processing of textual data, document classification, and clustering of documents. The computations are either performed on a central server, or they are distributed.

2.4 Grid monitoring use cases

A key problem in the development and operation of distributed systems is that their behaviour tends to be extremely complex. Traditional monitoring services are based on gathering local logs in each component of the system and on the manual, or software assisted, browsing of these logs by experts. This approach has three basic flaws to it: (1) any centralized approach is necessarily non-scalable with the number of logs, (2) many interesting events exceed the scope of a local log (e.g., distributed denial of service attacks) and can only be detected if all logs are processed and a system-wide picture is obtained, and (3) in order to maintain reasonable log volumes, the events that are interesting must be selected a priori; yet, many times an event only becomes interesting in the context of a given problem, which cannot be predicted.

An alternative approach, which has been tested in small-scale systems, is to use data mining techniques to look for unknown patterns in the logs. This approach allows logging many more events, since these will be processed automatically. It also has the potential to reduce the level of system expertise required from the operator because the same monitoring tool can be used to extract behavioural patterns from the logs of different systems. Nevertheless, until today all such experiments still had two drawbacks: they used centralized data mining techniques and were, thus, non-scalable, and they used the logging facilities

supplied by vendors and thus suffered from the need to fuse the data sources before they can be mined – which is in itself a difficult problem.

In the DataMiningGrid we will investigate a new approach to the monitoring problem. We will use the data mining toolkit that will be part of DataMiningGrid to mine logs that will be extracted specifically for this purpose. Since data mining is integrated with the system, the monitoring task will be distributed. For instance, we will be able to log much more data locally, because most of that data will be discarded of immediately and will pose no performance overhead on the system. This work is expected to have implications for all grid systems

The following generic use case refers to the task T64: Complex Monitoring Problems of the Workpackage WP6: DataMiningGrid Demonstrators in the projects Technical Annex (section 7.6.6).

After submitting a grid job it is of critical importance to allow user to monitor its execution, as well as to constantly diagnose the system. If the system is properly diagnosed, its performance can be improved by correcting the problems that occurred during the job execution, or by resubmitting the job to other resources that function properly. If the job submission was faulty (e.g., due to the incomplete data), the user can be alerted, and the information necessary for the correction of the submission can be provided.

Monitoring in the distributed systems is extremely complex since it must combine information from numerous and diversified resources that simultaneously execute multiple computational tasks. This is why current distributed systems offer very little monitoring functionality. The DataGrid [DataGrid04], for example, developed a monitoring facility called Logging and Bookkeeping that keeps track of the job submission, and enables querying of the current job execution status. However, this facility retrieves only a very limited set of predefined job status descriptions. Moreover, these descriptions are rather cryptic, and contain very little information regarding the problems that caused job failure, and how it could be corrected. Actually, a job that was factually not executed due to the application malfunctioning would be still reported as successful.

It is therefore necessary to provide a significantly more powerful monitoring functionality integrated into the distributed systems that would allow ongoing analysis of the system's condition, and provide critical diagnostic information.

2.5 Data mining distributed medical databases

The medical use case is especially intended to demonstrate the need for privacy and security when mining a real-world medical problem. Many medical studies are more relevant if conducted for a greater geographic area. In this particular use case, endemic goitre and supply with iodine in Slovenian children entering high school is investigated. The specific scientific objectives of this demonstrator are (1) to study the prevalence of insufficient supply with iodine, which is manifested as goitre, among children entering high school in 9 regions of Slovenia, (2) find out in which regions of Slovenia the prevalence of insufficient supply with iodine, manifested as goitre, is especially high, (3) to determine, if

there is a significant difference in nutritional habits between the children with deficient supply with iodine and their peers from the control group from the same region in Slovenia, who have adequate iodine supply, and (4) compare nutritional habits of Slovenian children entering high school with recommended Central European reference values.

Currently, approximately four thousand children are being recruited in nine Slovenian regions and the data is collected in several regional databases. The collected data is being analysed by statistical and data mining approaches. In this respect, the use case is not concerned with massive data neither it needs high-performance computing, however scalability is an open issue when mining larger (distributed) medical databases.

It would be highly beneficial if the data is either not transferred from its original location (for the purpose of data mining) or they could be transferred in a coded manner, thus preserving privacy, security and governance, while at the same time facilitating medical studies on the data.

2.6 Ecological modelling use cases

The ecological modelling use cases are taken from the scientific area of automated ecological modelling, but could be generalized on any scientific area that has the need for automated modelling of systems on the basis of knowledge bases.

Models of dynamic systems are often stated in terms of basic processes that govern the dynamic behaviour of the observed system. Each basic process influence the change of one or more system variables, while the model of a basic process specifies the equations used to model its influence. There are numerous mathematical formulations (models) for each of the processes in a system. When setting a model of a system the expert then makes a choice of a suitable expression that would describe the process best. In automated modelling a knowledge library can take the experts place by offering a variety of models for the same process to the equation discovery tool, which then finds one to describe the system best.

The modelling knowledge is gathered in a library of domain-specific knowledge. Given a specification of modelling task at hand, which includes specification of the observed system variables and processes that are expected to influence the behaviour of the system, we can transform the high-level knowledge from the library into an operational form of grammar that specifies the space of candidate models for the observed system. This is illustrated in the left-hand side of Figure 3: An automated modelling framework based on the integration of domain-specific models. Once we have a grammar, we can use equation discovery system Lagramge to heuristically search through the space of candidate models, match each of them to measured data by fitting the values of the constant parameters, and find the one that fits measurements best. This model is the output of Lagramge application.

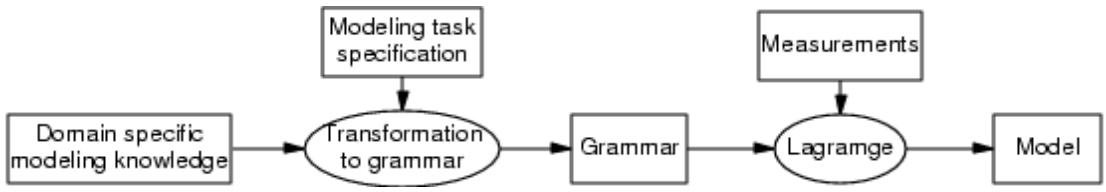

Figure 3: An automated modelling framework based on the integration of domain-specific models

The ecological use case is concerned with the task of building mathematical models of ecosystems, in particular population dynamics in aquatic ecosystems. Population dynamics studies the structure and dynamics of populations, where each population is a group of individuals of the same species sharing a common environment (ecosystem). We consider models of the dynamic change of population concentrations (or densities) that take the form of ordinary differential equations (ODE or ODE models). There are two main aspects of building an ODE model of a real-world ecosystem. First, we have to establish an appropriate ODE structure (the structure identification task). Second, we have to identify acceptably accurate values for the constant parameters of the ODE model (the parameter estimation task). Each of these aspects will be implemented on the grid as a separate computational service. However, a basic service for simulating ODE models is also necessary, that will allow the user to simulate and compare behaviours of the different models.

3 Requirements

The use cases motivate a number of technical requirements for the development of DataMiningGrid tools and services. The DataMiningGrid Consortium currently feels that the **technical requirements** described below are **essential** to the DataMiningGrid© technology. Each requirement **includes a short description** and is **motivated** by one or more use case(s) from the previous section.

Functional requirements lists the set of operations that the DataMiningGrid services and tools must be able to perform in order for the services and tools to be considered functional. **Non-functional requirements** are additional criteria that generally focus on the system properties, such as environmental and implementation constraints, performance, platform dependencies, maintainability, extensibility, reliability etc. of the services and tools. Non-functional requirements are further divided into special (non-functional) requirements and supplementary (non-functional) requirements. The special requirements are related to the specific use cases and need to be handled in subsequent system models, such as the analysis, design and implementation models. The supplementary requirements are generic requirements and can not be connected to a particular use case or a particular real world problem. They should instead be managed separately in a list of supplementary requirements.

The main classes of functionality required by users were identified as:

- Identifying (locating) DataMiningGrid resources by using metadata,
- Accessing and selecting subsets of data,
- Data transfer,
- Data (pre-) processing,
- Data mining tasks,
- Text mining and ontology learning,
- Workflow editing and submission,
- Data privacy, security and governance,
- Integration of domain knowledge,
- Grid infrastructure and middleware functionality,
- Usability, response times and user-friendliness.

3.1 Identifying (locating) DataMiningGrid resources by using metadata

Functional requirement 1: Provide mechanism for accessing metadata

Rationale: For the client to make decisions about how to handle the data provided by the data service, it is necessary for the client to access the metadata associated with the data. There are different forms of metadata. Application metadata describes properties of primary data that help the user to interpret it. Technical metadata describes how the primary data is stored in physical resources.

Initial Assessment: Application metadata would be most useful if provided the early stages of workflow design. This would suggest that application metadata should be part of the information supplied by the data service registry.

Technical metadata would mainly be used internally by the data services to construct data access and mediation workflows.

Motivation: Text mining use cases especially DC Use Case for Subtask T61.3: Finding related and similar documents in the intranet. Document repositories on the intranet contain metadata describing the kind, format, up-to-datedness and accessibility of their data.

Functional requirement 2: Provide mechanism for locating data services

Rationale: Data services are distributed all around the world. Locating all the required data services using ad-hoc search techniques is unlikely to be successful. By providing an automatic mechanism for locating data services, appropriate data services will be found quickly and easily.

Initial Assessment: This requires a data discovery service. Data discovery services identify data items by matching specified characteristics, attributes, or meta-data such as:

- A description, summary, or overview of the data source,
- Data provenance: how and where the data was generated i.e. what scientific instruments were used to collect the data, or what processing has been applied to the data,
- Who is the creator, the owner, or the last modifier,
- Physical metadata such as the data size, access control policies, transfer rates, error handling, number of disks, and the number of heads,
- It is also important to identify those data sources that are also available as computational resources so that data can be analysed at its source.

Discovering data services can be difficult in OGSA because:

- There can be a very large number of data objects to be discovered,
- Data models can be very complicated,
- It can be necessary to model and view various subsets of a dataset.

The semantic grid can be used to develop metadata ontology, and catalogues of metadata services are used to help locate and discover data.

Motivation: text mining use cases especially DC Use Case for Subtask T61.3: Finding related and similar documents in the intranet. Identification and location of document servers/repositories on the intranet which can or have to be searched.

3.2 Requirements concerning data privacy, security and governance

Functional requirement 3: Authenticate request and provide appropriate feedback.

Rationale: The security of access to data and other resources may be critical for some applications. It will be necessary to authenticate requests and provide appropriate feedback. Not all services are public. For those that are not it is necessary to confirm client identity and to check client access rights. The actions that the client takes next will depend on knowing if the authentication was successful or not, so the client requires appropriate feedback.

Initial Assessment*:* Authentication can be provided via a simple username/password combination or more elaborate techniques can be employed. On the other hand, for public access sites, authentication may not be needed at all.

Many individual datasets may be protected with different authentication techniques. Any service that provides access to these datasets may not only have to deal with the heterogeneous nature of data itself but also with heterogeneous authentication and access procedures.

There may also be restrictions on who can have access to which datasets or datasets may need to be anonymised (substituting untraceable codes in place of personal details) in order to ensure that personal information is not revealed.

Motivation:

- These use cases are the text mining from DC and FHG, general data access cases from UU, and system diagnostics from TECHNION,
- DC text mining use cases: Data and document repositories are distributed over different business units and departments. This implies different access policies on the distributed data. Therefore authentication must not only take place for the provided grid services in general, but the grid service itself must do authentication on a per server/repository basis.

Nonfunctional requirement 1 (supplementary): Useable from firewall-protected environments.

Rationale: To protect systems from unauthorized access, many networks make use of a firewall. This restricts external access to particular ports. Ideally the data services and tools will not be affected by the restrictions set by a typical firewall.

Initial Assessment: Most firewalls will allow the HTTP protocol on port 8080 This is sufficient to allow Web access but not allow unauthorized access beyond

the confines of public directories. Thus any protocol that makes use of port 8080 is likely to be accepted by most firewalls.

***Motivation**:*

- DC text mining use cases: Data and document repositories at DC are distributed over different business units and company departments. The DC intranet is not only protected against the internet by a firewall, but different units within the company intranet are also protected against each others by firewalls,
- Data mining distributed medical databases use case,
- Ecological use case.

Nonfunctional requirement 2 (supplementary): Use secure, encrypted channels.

Rationale: It may be necessary to pass sensitive information across the Internet. In such circumstances the information should be encrypted to ensure that sensitive information remains secure.

Initial Assessment: The inclusion of secure socket layer (SSL) software will encrypt data traffic to and from the server.

Motivation:

- Text mining use cases from DC,
- Data mining distributed medical databases use case.

3.3 Accessing and selecting data

Functional requirement 4: Initiate data service in response to successful request

Rationale: Assuming that a data service will be connection-oriented a session is required that will maintain the current state of the data service.

Initial Assessment: Data services can also be connectionless, like Web servers. In this case only one instance of the data service is needed, but each request must be self-contained and include authentication details where necessary. This means that a single request has the potential to be very complicated. On the other hand, a connection-oriented data service will maintain a current state, like FTP. Each individual request is simpler because its context does not have to be restated every time it is passed to the data service.

Motivation: All use cases.

Functional requirement 5: Data service provides access and mediation services to one or more datasets

Rationale: A data service may provide access to one or more datasets. The client application requires a consistent approach to access and handle the data provided by the data service. A mediation service will provide an integrated view of distributed data to the client application.

Initial Assessment: Data sources need to have structure in order for them to be recognized or understood. This structure should be described by metadata. Data sources are usually file orientated or database orientated. Sources of datasets include:

- File systems e.g. NFS or AFS as a flat file or collection of flat files such as ASCII text files that have been formatted according to some clear convention or documents from which data can be extracted and then presented in a structured manner,
- Structured or semi-structured XML data,
- Database management systems: particularly relational (e.g. ORACLE or MySQL),
- Abstract or virtual data that has been derived from a subset of one or more data sets,
- Directories of grid services i.e. data that is used to describe available resources, to support the operation of the grid, and to describe its state and configuration.

Motivation: Most, if not all, use cases will require access to data for mining.

Functional requirement 6: Data service processes basic queries from client

Rationale: The data service must accept queries that allow the client to select data and perform other basic operations.

Initial Assessment: Users want to access both data and metadata from the located source(s) and select a data subset for processing or mining. For the client to make decisions about how to handle the data provided by the data service, it is necessary for the client to access the metadata associated with the data. The way in which data is structured will affect the manner in which it can be accessed:

- Data access can involve both reading data from, and writing data to, a data source. These operations will be implemented in different ways for different resources,
- Data access may be needed to update data sources, and to ensure the consistency of all data replicas,
- Data sources have different access mechanisms:

- File system based commands like open, close, read, write. Data subsets can be selected using file processing scripts,
- Relational query mechanisms like SQL, XQUERY, XPATH,
- Mechanisms for hierarchical storage systems.

Diverse sources will result in a diversity of access mechanisms. To enable robust generic access mechanisms we will want:

- Uniform methods to access a data source e.g. use GridFTP, although it is also good for multiple mechanisms to exist,
- To be able to transform data amongst the different data types that are stored in the different data sets,
- To be able to mediate between different data models and database schemas,
- Federated or virtual databases may be required in order to integrate various data sources into a transparent global schema. Individual data sources may still need to be available for direct access,
- Specific views of a data source may need to be provided.

It is often desirable to be able to select a subset of the data accessed from a data source before it is transferred or stored elsewhere. Data selection may be an interactive operation and require sophisticated tools to guide the user:

- At the simplest level a data subset can be selected using relational database query mechanisms,
- At the most complex level there are many different ways of filtering, combining, or processing data, and these methods can require sophisticated code to be executed.

Motivation: All use cases.

3.4 Data transfer

Functional requirement 7: Data service must be able to transfer datasets from one server to another

Rationale: By its very nature the grid is geographically distributed. Therefore, in order to access the resources available at different locations, it must be possible to move datasets from one geographical location to another.

Initial Assessment: Moving data requires no knowledge of the data structures. The basic unit of data transfer and access is:

- Primitives: floats, integers, characters and arrays, images or objects,
- Files: these are uninterrupted sequences of bytes.

GridFTP is the fundamental data access and transport mechanism and it provides a uniform interface to different file storage systems. When moving data it is necessary to consider:

- Maximizing file sharing, minimize replication and to monitor shared file space i.e. perform garbage collection,
- Dealing with failure and restarting transfers,
- Synchronising updates to all replicas,
- Allowing different types of file transfer:
  - The simultaneous transfer of multiple files with each job individually monitored and managed,
  - Reliable data transfer i.e. GridFTP with some enhanced features.

Motivation: All use cases.

3.5 Data (pre-) processing

Functional requirement 8: Include additional functions such as data cleaning operations and data transformation operations

Rationale: Operations on large amounts of data should be kept as close to the data as possible in order to reduce data transfer overheads. Ideally such operations would be integrated with the data services and tools.

Initial Assessment: Additional grid data service functions may execute specific algorithms on the data, such as filling in missing values or transforming the data in different ways (such as normalising the data). This can be done more efficiently by the data service since fewer data transfers are required.

At the extreme level, all the data mining tasks could be performed by the grid data service. Programs (probably either scripts or Java programs) could be uploaded to the grid data service and used to perform tasks on the data.

Motivation:

- Genetic algorithms for gene regulatory reengineering use case,
- Information integration of life science data: an integrated approach to protein subcellular localization prediction use case,
- Text mining use cases,
- Data mining distributed medical databases use case.

Functional requirement 9: Data processing taking place near where data is located

Rationale: For large amounts of data, there may be high overheads of transferring the data from where it is store to where it is processed. There might

also be cases, where the right to move or distribute the data to other servers may be restricted due to data privacy or copyright reasons. Where possible, as much processing should take place as near to where the data is located as possible in order to keep data transfer times to a minimum and to avoid inconvenience to other users due to the increased traffic caused by data transfers.

Initial Assessment: Processing data close to its source may be important in order to:

- Scale computation,
- Reduce the dataset's size before it is transferred,
- Realize virtual datasets i.e. a new dataset that has been derived in some way from one or more data sources. These results could then be accumulated in other data collections: this is similar to virtual data warehousing,
- Format the dataset (into XML for example) before it is transferred,
- Access and process data which can not be redistributed or transferred,
- Process highly dynamic data, where data changes faster, than transferring the data to another place would take.

Motivation:

- Text mining use cases: Text data should be converted in place from their original document format (different binary and proprietary file formats like MS Word, Powerpoint, PDF, …) to a data format suitable for doing text mining and retrieval (e.g. ASCII, XML, vector-representation, …). If the conversion takes place on the data repository, the converted data can be cached for future queries and analyses,
- Genetic algorithms for gene regulatory reengineering use case,
- Information integration of life science data: an integrated approach to protein subcellular localization prediction use case.

3.6 Data mining tasks

Functional Requirement 10: Decoupling WEKA components

Rationale: Decoupling WEKA components (data mining services, etc.) for optimal extensibility and platform/language independence

Initial Assessment: Weka [Weka04] is a widely known and accepted data mining application, especially in academic and research areas. It contains many state of the art data mining algorithms. It also provides a workflow editor capable of creating sophisticated data mining workflows. As this editor encompassed nearly all data mining algorithms provided in Weka, it is desirable to have Weka being able to execute on the grid in order to use this editor to its full extend.

Motivation: Weka's components can be used to build one integrated system encompassing demonstrators from several or all Partners. Furthermore, with Weka being able to execute on the grid users can choose between many more additional analysis services.

3.7 Text mining & ontology learning

All text-mining demonstrators will be realized on decentralized document sources and the processing of the documents will also be realized decentralized in a grid environment. The requirements relating to the "text mining" and "ontology learning" use cases have already been covered in sections 3.2 to 3.4 (accessing distributed data services, data preprocessing and transfer).

3.8 Workflow editing and submission

Functional requirement 11: Ability to prepare workflows by graphical editor and submit workflows for execution

Rationale: Data mining is a complex process consisting of many tasks for pre-processing, analysis, and visualization. In order to compile these tasks for automatic execution, a graphical workflow editor providing such capability is needed. Also such a visual tool will reduce error rates compared to traditional batch scripting. Furthermore as the workflow editor will not execute the workflow, it has to be able to submit it to some machine where it is executed.

Initial Assessment: In order to contribute to the system's usability for end users the workflow editor has to support various demonstrators while hiding as many grid specific aspects as possible from end users. This demands for the editor to provide a generic interface for adding new services during runtime by using the Data & Analysis Discovery/Location services as explained in functional requirements #1 and #2 to present available services to the user.

The workflow editor will also perform basic validation of the workflow chain ensuring syntactical soundness.

These functions require the computational and data services to provide extensive descriptions about themselves. It has to be carefully evaluated which information is needed in order to keep data exchange and overall complexity to a minimum while still ensuring full functionality.

Execution of the workflow is not part of the editor. Therefore, it is necessary to submit the workflow to some sort of manager, which starts the execution and monitors the progress.

Motivation: The basic idea is to provide an extensible workflow editor, which can be used in many demonstrators.

3.9 Requirements concerning integration of domain knowledge

Functional requirement 12: Ability to store models in a knowledge base

Rationale: The data to be mined in complex problem solving scenarios will increasingly require the integration of existing domain knowledge into the mining process, particularly in knowledge-intensive domains. This knowledge is often dispersed across geographically distributed sites and organisations, either in digital form (ontologies, metadata, knowledge-based or simulation systems) or provided interactively by human experts. grid-enabled data mining technology will need to provide services and tools to support knowledge-aided data mining.

Initial assessment: Storing formulae in a knowledge database could be highly beneficial for some machine learning algorithms, e.g. Lagramge.

Motivation: Ecological modelling use case.

3.10 Grid infrastructure and middleware requirements

Functional requirement 13: Middleware able to execute generated workflows including conditional control flows

Rationale: Data mining is complex process including many sub-tasks. As some may only be executed if specific conditions are fulfilled and others may be repeatedly executed a specific number of times, the workflow editor has to provide conditional control flows.

Initial Assessment: Conditional control flows are essential to the usability of the workflow editor, especially if this editor is supposed to support several demonstrators. Without such control flows the workflow editor would only be of limited use.

Motivation:

- Text mining use cases: Distributed document clustering and training of classifiers takes place as iterative process, being repeated until a certain quality criteria is met,
- Ecological use case.

Nonfunctional requirement 3 (supplementary): The DataMiningGrid services and tools should be compatible with existing grid infrastructure.

Rationale: Using the existing grid infrastructure will reduce development time and will make implementation and installation quicker and easier. However this will only be possible if the existing grid infrastructure is capable of supporting the required services and tools.

Initial Assessment: Where possible, existing protocols (grid interoperability profile WS-I+) and infrastructure should be used. This may mean compromising

on some aspects of the project, but will enhance compatibility and reduce the risk of project failure.

Motivation: All use cases.

Nonfunctional requirement 4 (supplementary): Grid infrastructure, test-bed for the DataMiningGrid project.

Rationale: Grid infrastructure should be developed by the project Partners for technology testing purposes.

Initial Assessment: Where possible, existing protocols and infrastructure should be used. This may mean compromising on some aspects of the project but will enhance compatibility and reduce the risk of project failure.

Motivation: All use cases.

Functional Requirement 14: Provide monitoring services.

Rationale: Constant system diagnosis is of critical importance in complex, distributed systems. In such systems it is extremely difficult to track and monitor job execution, and to understand what causes failures and poor performance. An effective ongoing analysis of the system's condition is necessary in order to enable corrective actions, and improve system's performance.

Initial Assessment: Ongoing analysis of the system's condition enables early detection of the occurring problems, detects points of failure, and exposes weaknesses that potentially or factually harm the system's performance. The corrective actions and the performance improvement based on this analysis need to be automated to the maximal extent catering for the dynamic and often unpredictable modifications and alterations occurring in the system. The user needs to be notified regarding the status of her job submissions, and provided with the sufficient information to carry out corrective actions, if these are within her capacity as a user (for example, job resubmission with the complete and correct data). The system administrator needs to be updated regarding the reoccurring problems and potential weaknesses, and provided with the sufficient information to reconfigure the system if necessary.

Motivation:

- Grid monitoring use cases,
- Ecological modelling use case.

3.11 Usability, response times and user-friendliness

Nonfunctional requirement 5 (supplementary): Data services and tools response time is within a reasonable time period.

Rationale: The user will wish to know if an operation has succeeded or failed within a reasonable time period.

Initial Assessment: Response time when client-application connects to a service should be within seconds (10 seconds is realistic response time for job submission confirmation. However, resource allocating facilities of the current distributed systems operate in minutes, rather then seconds. For example, EDG Resource Broker takes on average 5 minutes to execute a simple echo job). Response time when client-application sends a query should also be within seconds in order to inform the user if the query can be processed or not (30 seconds is suggested as a reasonable time). For the DC Text-Mining Use case "Find related documents" even shorter response time guarantees seem reasonable. Otherwise, system should give the user regular progress reports.

Time to transfer a large amount of data will depend on the amount of data to be transferred (no limit can be placed on this).

Motivation: All use cases.

4 Objectives – research goals beyond the scope of the project

In addition to the set of features that should be part of the DataMiningGrid technology (as defined by requirements in the previous section), there are other features that would be useful for many use cases. These requirements can be classified as nice-to-have requirements. These useful, but non-essential requirements will be addressed by the Consortium if possible, but the Consortium may decide that there are good reasons for excluding them or leaving them to be implemented by a later project. Some of these objectives are not fully defined, and as such need further clarification if they are to be addressed by the DataMiningGrid© Consortium. Note that the order of the objectives below does not imply relative priority or degree of consensus.

Functional requirement 15: Provide mechanism for locating computational services

Rationale: Similar to the data services, the analysis services are also distributed all over the grid. Since ad-hoc search techniques are inadequate to locate such distributed services, an automatic mechanism for locating them is needed in order to find them in a quick, easy, and convenient way.

Initial Assessment: This requires an analysis discovery service. Similar to data discovery services, analysis discovery services identify analysis services by matching specified characteristics, attributes, or meta-data such as:

- A description, summary, or overview of the analysis service,
- A description of the input and output data,
- The version of the service,
- Who is the creator, the owner, or the last modifier.

Motivation: All the use cases which will need to locate appropriate computational service(-s) will benefit from automatic service discovery.

Nonfunctional requirement 6 (supplementary): Data services and tools must be scalable

Rationale: Scalable means that larger datasets should not take a significantly longer time to process than just one dataset.

Initial Assessment: Where possible, data should be divided into subsets and sent to distributed processes. The data can then be processed in parallel and, in theory, should not take significantly longer than the time it would take one data subset to be processed on by one process. In practice there will be additional overheads due to scheduling of processes and distribution of datasets.

Furthermore there may be other bottlenecks, for example if all the data must pass through a single gateway.

Where data is already distributed, it may be possible to avoid bottlenecks by transferring the data directly to the distributed process without passing through the client or a specific service node. However, this raises some technical problems in how to clean the data and prevent repetitions in the data.

Motivation: All use cases.

Nonfunctional requirement 7 (supplementary): DataMiningGrid services and tools must be transparent.

Rationale: The end-user should not need to know about or understand the low-level details of how the technology works. However, users should always be aware of the possible intricacies of distributed systems, like latency, availability and reliability (See [Spol00] and [Spol02]).

Initial Assessment: It is assumed that end-users will be conversant with how to set up data mining workflows. However, the end-user will not necessary be conversant with the structure of the grid technology that underlies the data services and tools. Data services should be presented to the user as if it were a file or any other source of data.

Motivation: All use cases.

Nonfunctional requirement 8 (supplementary): Extensible and "future-proof"

Rationale: The design of the DataMiningGrid services and tools should be flexible enough to allow additional functionality to be included without inhibiting backward compatibility.

Initial Assessment: The key to making a system "future-proof" is simplicity. For example, a simple set of functions and operations that can be combined to perform any task are better than a smaller set of functions that require more arguments and are more likely to require change in the future. If necessary, new functions can be introduced but these should follow the same principal.

Motivation: All use cases.

Nonfunctional requirement 9 (supplementary): Grid infrastructure allows interoperability between heterogeneous programming environments and different operating systems

Rationale: The existing data mining components of the project Partners run on different operating systems (Linux and Windows) and are written in different and incompatible programming language. To reduce development time it must be possible to adopt them to the grid with out having to port them to another programming environment or operating system.

Initial Assessment: Grid interoperability standards should guarantee this requirement. It may be necessary to write glue/bridge code in some cases.

Motivation: All use cases.

Nonfunctional requirement 10 (supplementary): Data service interactions should be kept simple.

***Rationale*:** A complex protocol is more difficult to implement and more likely to fail. The additional functionality provided by a more complex protocol can often be achieved by using several simpler commands.

Initial Assessment: A simple text-based language that is both simple and extensible. Every command has a response in order to confirm the command has been successfully processed.

Motivation: All use cases.

5 Conclusions and future work

While investigating the generic data mining requirements of emerging grid-enabled problem solving environments based on a selected set of representative application sectors and problem domains, the DataMiningGrid© Consortium aims to develop algorithms and tools facilitating grid-based data mining services for future and emerging complex problem solving environments.

The deliverable D11(1) was prepared jointly by the Partners to address some important requirements for data mining in grid computing environments. As such, it represents only a subset of the most important requirements that the Consortium has to address at the beginning of the DataMiningGrid project. More requirements will certainly arise during the next phases of the project.

Due to the impending data influx in many sectors, the need for data mining is likely to increase. With the advent of increasingly flexible and powerful grid-computing infrastructures, it is high time to think of and to instigate the development of grid-enabled data mining services. Such services would facilitate:

- a dynamic and secure way of accessing, retrieving, and manipulating (join, subset selection, filtering, etc) data sets from heterogeneous and distributed sources,
- a dynamic and secure association of these data sets to operations provided by data mining servers available in a distributed computing environment, i.e. the grid,
- a dynamic and secure execution of these operations on the data sets,
- a dynamic and secure pipelining of such operations and the resulting intermediate data sets,
- a dynamic and secure allocation and addition of new data mining services and operations (servers), new databases and data sets to the grid,
- a highly interactive, intuitive, and secure way for users to define, execute, monitor, and manage a data mining workflow in such a distributed data mining environment.

Along with these developments the DataMiningGrid© Consortium shall demonstrate the deployment of the developed web-based, grid-enabled, data mining applications, modelling tools and services in a carefully selected sets of representative application sectors (see Figure 1). The selected applications and technologies include: re-engineering of gene regulatory networks via distributed genetic programming, analysis of biological databases for gene/protein annotation, data mining based monitoring of grid systems (including analysis of recurring failure and descriptive analysis), distributed text classification and ontology learning for customer relationship management and quality management, finding related and similar documents in the intranet, mining of digital (scientific) libraries, analysis of distributed medical databases for endemic goitre and iodine deficiency studies, distribution of algorithms for mining of data in the context of sanitary engineering, and mining of literature databases (including equation discovery, data check/filtering).

6 References

[Avaki04] AVAKI at http://www.avaki.com/

[Bax02] Baxter R, A Complete History of the Grid at www.nesc.ac.uk/talks/sdmiv/SDMIV-25Oct2002- Baxter.pdf

[Condor04] The Condor Project at http://www.cs.wisc.edu/condor/

[Condor-G04] http://www.cs.wisc.edu/condor/condorg/

[Czaj04] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. David Snelling and S. Tuecke, "From Open Grid Services Infrastructure to WSResource Framework: Refactoring & Evolution," http://www-fp.globus.org/wsrf/OGSI%20to%20WSRF%201.0.pdf , Tech. Rep. 1.0, 12/2/2004, 2004.

[DataGrid04] The DataGrid Project at http://eu-datagrid.web.cern.ch/eu-datagrid/

[Dee90] Deerwester, S. S. & Dumais, T. K.Landauer, G.W. Furnas, and R.~A. Harshman (1990). Indexing by latent semantic analysis, in: Journal of the American Society of Information Science, 41(6):391--407, 1990.

[DQP04] www.ogsadai.org.uk/dqp/

[Dub01] Dubitzky, W., Krebs, O., and Eils, R. (2001), 'Minding, OLAPing, and Mining Biological Data: Towards a Data Warehousing Concept in Biology', Proc. Network Tools and Applications in Biology (NETTAB), CORBA and XML: Towards a Bioinformatics Integrated Network Environment, Genoa, Italy, pp78-82.

[ELDAS04] www.edikt.org/eldas

[Fer03] Ferreira, L., Jacob, B., Slevin, S., Brown, M., Sundararajan, S., Lepesant, J., Bank, J. (2003) Globus Toolkit 3.0 Quick Start http://www.redbooks.ibm.com/redpapers/pdfs/redp3697.pdf

[Fos02] I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, at www.gridforum.org/ogsi-wg/drafts/ogsa_draft 2.9_2002-06-22.pdf

[Fos04] I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, T. Storey, W. Vambenepe and S. Weerawarana, "Modeling Stateful Resources with Web Services," http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf , Tech. Rep. 1.0, 01/20/2004, 2004.

[Fos99] I. Foster, C. Kesselman, The GRID, Morgan Kaufmann Publishers, Inc., San Francisco, 1999.

[GGF04] The Global Grid Forum Database Access and Integration Services Working Group at http://www.gridforum.org/6_DATA/dais.htm

[Globus04] The Globus Alliance at http://www.globus.org/

[Gra97] Ingo Graf and Ulrich Kreßel and Jürgen Franke, Polynomial Classifiers and Support Vector Machines, 1997, 00 -- 00, Inter. Conf. on Artificial Neural Networks, Zürich

[Gru93] Gruber, T. R. (1993): Toward principles for the design of ontologies used for knowledge sharing, in: Formal Analysis in Conceptual Analysis and Knowledge Representation, Kluwer 1993.

[Hof01] Hofmann, Thomas, Unsupervised learning by probabilistic latent semantic analysis, in: Machine Learning,42:177--196, 2001.

[Joa98] Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features, in: Proceedings of the Tenth European Conference on Machine Learning (ECML '98), Lecture Notes in Computer Science, Number 1398

[Lan97] Landauer, Thomas K. & Dumais, Susan T. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge Psychological Review 104 (2): 211--240, 1997.

[Leo02] Edda Leopold & Jörg Kindermann (2002): Text Categorization with Support Vector Machines. How to Represent Texts in Input Space?; in: Machine Learning 46, pp. 423 - 444.

[Man99] Manning, C. D. & Schütze, H. (1999): Foundations of Statistical Natural Language Processing, MIT Press: Cambridge MA, London.

[OGSA-DAI04] www.ogsadai.org.uk/docs/current/doc/DAIOverview.html

[Park03] Park, K., and Kanehisa, M. (2003), 'Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs', Bioinformatics, Vol. 19, pp. 1656-1663.

[Rom02] M. Romberg, "The UNICORE Grid Infrastructure", Scientific Programming Special Issue on Grid Computing, 2002, 10, pp. 149-158.

[Sal83] Salton, G.& McGill, M. J. (1983) Introduction to Modern Information Retrieval, McGraw Hill, New York.

[Spol00] Spolsky Joel; Three Wrong Ideas From Computer Science, August 22, 2000; www.joelonsoftware.com/articles/fog0000000041.html

[Spol02] Spolsky, Joel; The Law of Leaky Abstractions, November 11, 2002; http://www.joelonsoftware.com/articles/LeakyAbstractions.html

[Stan04] Stankovski V., May M., Franke J., Schuster A, McCourt D., Dubitzky W., A Service-Centric Perspective for Data Mining in Complex Problem Solving Environments, H.R. Arabnia and J. Ni (eds) Proc of Int'l Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'04), Vol II, pp. 780-787, 2004.

[Stork04] www.cs.wisc.edu/condor/stork/papers/stork-icdcs2004.pdf

[Stur03] Sturgeon, B., McCourt, D., Cowper, J., Palmer, F., McClean, S., and Dubitzky, W. (2003), 'Can the Grid Help to Solve the Data Integration Problems in Molecular Biology?', 3rd International Symposium on Cluster Computing and the Grid (CCGRID 2003) , Tokyo, Japan, pp. 594-600.

[Tom99] M. Tomassi. Parallel and distributed evolutionary algorithms: A review. In K. Miettinen, M. Makela, P. Neittaanmaki, and J. Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, pages 113-133. J. Wiley and Sons, Chichester, 1999.

[Weka04] – Weka Homepage; http://www.cs.waikato.ac.nz/ml/weka/

[Weka99] – Witten, Ian.H, Frank, Eibe; Data Mining Practical Machine Learning Tools with Java Implementations; Morgan Kaufmann; 1999

Appendix A: Applications description

This appendix contains detailed description of applications and data sources within the DataMiningGrid Consortium. It contains technical requirements for use of these applications as stand alone applications, what is their input and what is the output etc. This was prepared, so that the Partners become familiar with the (tentative) software tools and technologies that shall be used during the project.

UU application farm and data sources

UU will demonstrate two use cases for which the following applications are considered:

Use case 1: Genetic algorithms for gene regulatory reengineering

The essential aim of the use case is to develop a demonstrator that uses a parallel evolutionary algorithm approach to evolve gene regulatory networks from expression time series data on the GRID.

There are three different approaches to parallelize evolutionary algorithms [Tom99]:

1. Master-Worker model: one master process manages the population and hands out individuals to evaluate a number of workers. Ideally you have one worker per individual:

   - The master node has to recognize available nodes in the Grid and use them as workers.
   - The master node has to transfer the individual to each worker, execute the individual (in this case an executable simulation of a gene-regulatory network) and calculate the fitness.
   - After the evaluation, each worker node has to return the fitness value with the individual's identification to the master node.

2. Island distributed evolutionary algorithms: Semi-independent groups of individuals or subpopulations (demes) evolve separately with a loose coupling to each other:

   - An organizer node has to recognize available nodes in the Grid and use them as deme nodes (demes represent isolated subpopulations).
   - Individuals migrate with between demes.
   - At regular intervals the organizer node has to be informed about the development state of each subpopulation (e.g. report best fitness value).

3. Cellular genetic algorithms: Each individual of the population is placed on one grid-node. The fitness is evaluated simultaneously for all individuals and

selection, reproduction and mating take place locally within a small neighbourhood:

- An organizer node has to recognize available nodes in the grid and assign one individual to each node.
- Synchronization is needed for the fitness evaluation.
- Individuals migrate locally. This requires a spatial mapping of each individual node within the grid (or the mapping could be implemented in the organizer node).

If one approach can be implemented then all approaches are possible. However using a master-worker or a cellular-GA approach makes only sense if the time consumption of the data transfer is low in relation to the time needed for the evaluation. Further, a master-worker approach still has all "ugly" properties of a sequential run i.e. sticking to local maxima would be still a problem. Therefore, it would be best to start with island distributed evolutionary algorithms (approach 2). If the networks become large and the evaluating the simulations requires more time, then this approach may be extended by combining island distributed evolutionary algorithms with the master worker approach (approach 1 with approach 2).

Software and database requirements:

- A scheduling system like Condor for managing and scheduling distributed computing jobs,
- In-house Java code for bioinformatics applications,
- Eclipse: this application is the IDE used to develop Java applications. The technical reasons for using this application are:
  - It is free open source software supported by IBM,
  - It runs under both Windows and Linux,
  - It uses the standard Sun Microsystems software development kit.
- Sun Microsystems SDK: the Sun Microsystems Software Development Kit provides the compilers and documentations needed to create Java applications. The technical reasons for using this are:
  - The Sun Microsystems SDK is the standard development kit for Java,
  - Java is machine independent, and therefore portable,
  - A database may not be needed, and even if it is required it will not be an important element of this use case.

Use case 2: Information integration of life science data: an integrated approach to protein subcellular localization prediction

This use case is concerned with developing computational methods to accurately predict the location of proteins within the cell. Such tools are needed in order to decipher the mass of data being generated by large scale sequencing projects.

It is possible to identify three computational approaches to predicting protein subcellular localization:

- Prediction by signal i.e. recognizing protein sorting signals from the genetic or amino acid sequence,
- Identifying sequence homology to a protein with a known subcellular location,
- Prediction by content i.e. deviations in the protein amino acid composition differ for the various subcellular compartments. This approach has lead to a number of prediction methods based solely on sequence composition [Park03].

However, there is a considerable need to identify and develop more general and accurate solutions to protein subcellular localization prediction, and to this end this use case will integrate databases containing the proteins' crystal structures with databases dedicated to understanding and describing their amino acid sequences. This will enable proteins to be classified by their three dimensional structure and surface properties, as well as by their amino acid composition.

Software and database requirements:

For our demonstrators we will need to use public bioinformatics databases (actually databanks):

- Protein Databank (PDB),
- Swiss-Prot.

We will be able to install copies of these databanks on to our own machines. They supply data as ASCII text files with clear formatting conventions. Oracle is being considered as the database application for use at UU and in the UU demonstrators to store sample data. This is because:

- It is well supported by Oracle,
- It is highly flexible and provides an ideal platform for all database needs,
- In order to classify the data we will use decision tree software that is being developed at UU by Brian Surgeon.

1. EXISTING SOFTWARE FOR DATA SERVICES:

Data services will be required to identify data sources, access them, select subsets of data from the data sources, construct data sets, and transfer the resultant dataset around the grid.

Data access and integration is an essential element of all data grids, and a number of solutions to this problem have been developed by other grid projects. In order to reduce the development time of the data services work-package DataMiningGrid will use existing data management solutions where appropriate. Many of these data-management solutions are still under development and the developers are keen to gain feedback from anyone using these solutions. Collaborating with these projects will help to disseminate DataMiningGrid expertise and aid in the construction of underlying grid technology developed specifically for data-mining applications.

A non-exhaustive list of existing data services solutions is given here.

1.1. Spitfire

Spitfire was developed by the EU DataGrid project and provides a set of grid enabled middleware services for access to relational databases. In order to provide reliable and simple access to datasets, Spitfire decouples the client and RDBMS using a mediator. Custom code clients, browsers and command line tools can read and write over HTTP(S) into any RDBMS. It is a good system to use for gaining access to SQL databases but other data sources such as file systems are not supported.

1.2. Stork

Stork [Stork04] has been developed by the Condor project. It is a specialized scheduler for data placement activities in grid computing. Stork allows data placement tasks to be queued, scheduled, monitored and managed in a fault tolerant manner. By interacting with higher level work flow managers such as the Condor project's DAGMan, both storage and computational resources can be scheduled together.

Stork currently has support for:

- Data transfer protocols such as FTP, GridFTP, HTTP, and DiskRouter,
- Data storage protocols such as SRB, UniTree, and NeST,
- Data management middleware such as SRM.

The transfer protocol used to transfer data may be decided at run-time by using Condor ClassAds to describe all the resources that Stork will be interacting with. If one transfer protocol fails then Stork can change to alternative transfer protocols. Stork is also able to allocate and deallocate space on storage resources using NeST and it can ensure that network links do not get overloaded.

Although Stork provides a robust solution to data transfer, it has can only support simple data access facilities, and currently users can only move around complete files.

Condor-Stork is a robust and mature solution, but it only offers a solution to the problem of transferring files. It cannot provide access to databases, and it does not provide any solutions to problems involving data identification or manipulation.

Although it may be possible to develop additional functionality into Condor-Stork, it is necessary to ask whether investing resources into grid middleware development is a priority for the DataMiningGrid, especially when many of these problems have already been solved by systems such as OGSA-DAI or Eldas.

1.3. GGF Data services solutions (OGSA-DAI and ELDAS)

The Data Access and Integration Working Group (DAIS-WG) within the Global Grid Forum (GGF04) is producing the Grid Data Services Specification (GDSS). The GDSS defines a Grid Data Service (GDS) standard for accessing and integrating data stored in multiple types of data storage systems (e.g. relational databases, XML file systems).

There are two implementations of the GDSS:

- OGSA-DAI,
- ELDAS.

Both of these systems are developed in Scotland at the National e-Science Centre (NESC), and represent two different architectural approaches to implementing the DAIS standards.

An email that summarised the differences between the two systems was provided by Rob Baxter, the software manager for both Eldas and OGSA-DAI, and it is given here:

- OGSA-DAI is designed for a web container host (tomcat) and Eldas is designed for an EJB/J2EE host (JBoss in particular),
- OGSA-DAI is dependent on Globus Toolkit 3 but a "pure webservices" (WS-I) version will be available soon; Eldas already has a "pure webservices" version and can be installed with no dependence on Globus,
- OGSA-DAI (currently) has richer functionality i.e. activities like data transformation and compression, enhanced data delivery - while Eldas (currently) has a greater emphasis on ease-of-installation and use,
- OGSA-DAI is enhancing its usability; Eldas is enriching its functionality.
- OGSA-DAI is open source whereas Eldas (currently) is not.

Whether to use OGSA-DAI or Eldas comes down really to which web service/Java container architecture you want to use - EJB or non-EJB - and whether you require open source software. There are plans to develop both platforms in the future, although for funding reasons the roadmap for OGSA-DAI is clearer.

1.3.1. OGSA-DAI

The OGSA-DAI [OGSA-DAI04] aims to provide an extension to the Open Grid Services Architecture (OGSA) specifications in order to allow data resources to be incorporated within an OGSA framework. OGSA-DAI components can be used as basic primitives in the creation of sophisticated higher-level services that offer the capabilities of data federation and distributed query processing within a Virtual Organization (VO).

OGSA-DAI is motivated by the need to integrate data sources and resources into an OGSA-compliant architecture. In particular it is required to:

- Obtain information about data that may be distributed amongst several heterogeneous database environments, and interpret the data held in databases.
- Locate data that may be distributed, or replicated, over many different types of databases (relational, XML, some functionality for file systems etc.), the locations of which may not be known beforehand.
- Integrate different data models from distributed databases.
- Access that data through uniform interfaces.
- Integrate data from various sources to obtain the required information.

These requirements in turn give rise to the following constraints. Any solution must be able:

- To provide a service for the registration and subsequent discovery of databases with the required data.
- To support access to these databases and subsequent interactions with the databases.
- To provide control over the structure of the results, following a database interaction.
- To provide control over the method and location of data delivery, following a database interaction.
- To ensure that the type of database interactions and the specified data transport methods are independent of the type of database and the data model.

Within OGSA-DAI, grid data services (GDS) are used as interfaces to data resources; the capabilities of the data resources are determined by the GDS, and these are published to the service registry. Such capabilities include information on the resource that a GDS connects to, the database management system, driver and schema, the type of queries that may be executed, the format of the resultant data, and the XML schema that defines the documents (known as GDS-Perform documents) that are past to and from the GDS.

The GDS-Perform documents specify the actions to be performed on a data resource (for example an SQL or XML query, or a database update, or a data transfer for data delivery and receipt) and the data that is extracted from that data resource. Data can be transformed when it is stored within the XML of the GDS-Perform document by using XSL (Extensible Stylesheet Language) Transformations (XSLT).

Both the US (GSF) and Europe (EG) have agreed to agree on standards, and the future releases of OGSA-DAI will be released in collaboration with OMII (the Open Middleware Infrastructure Institute).

OGSA-DAI has much more functionality than Stork. Although it can be more difficult to install, OGSA-DAI provides services for data identification, data manipulation and data transfer. It can be used with relational databases, XML or file systems (with more limited data access services),

and it is able to select subsets of data from these data sources and then transfer the subset: this can be an advantage when using large files because Stork, as it is, could only transfer the entire data set. A requirement that the Data Services work package of the DataMiningGrid project must be able to meet involves performing SQL joins when accessing data from databases. OGSA-DAI can be used to integrate different databases, and apply such SQL statements across the two databases. It is not clear how functionality like this could be developed in Condor-Stork without a large commitment of time, resources and risk.

Using OGAS-DAI can reduce development time, and it is becoming a common solution to grid data-management problems with about 800 registered users worldwide. The OGSA-DAI team now consider the problems of data access to have been solved and are now focussing more on problems of data integration and distributed query processing (DQP), as described below.

DQP: Distributed Query Processing for OGSA-DAI

DQP [DQP04] is a distributed query processor that acts over web and grid services: it integrates query processing technology with a service-based grid. DQP extends the OGSA-DAI architecture with two new services - the Grid Distributed Query Service (GDQS) and the Grid Query Evaluation Service (GQES). The OGSA-DQP should yield significant programmer productivity and performance benefits for large-scale data intensive applications.

Grid Distributed Query Service (GDQS) is a high-level data integration service that interacts with OGSA-DAI. It combines data analysis with data access and integration by enabling calls to web services to be incorporated within a distributed query.

The supported OQL query types include:

- Select, From, Where queries in general
- Limited forms of nested queries
- Some forms of OQL-style aggregation (count, sum, max, min, avg)
- Typed function invocation
- Conjunctive predicates (i.e. connected by the logical AND operator)
- Equality-based join conditions
- Unnesting of collection-typed attributes
- Comparative operators are supported (=, !=, >, <, <=, >=, like)

The transform activities are the same as those supported by OGSA-DAI and include compression (i.e. zip and gunzip) and transforming data in XML documents using XSLT.

The following query [from http://www.mygrid.org.uk/] is used as an example to illustrate the GDQS query submission procedure:

```
select p.ORF, go.id, calculateEntropy(p.sequence)
from p in protein_sequences, go in goterms, pg in protein_goterms
where go.id=pg.GOTermIdentifier and p.ORF=pg.ORF and
p.ORF like "YBL06%" and
go.id like "GO:0000%"
```

The query contains two separate join operations each joining two tables from different databases (on different servers) and applies entropy analysis on protein sequences obtained from one of the tables using a web service. The calculateEntropy method is an operation defined by an EntropyAnalyserService. Note that the parameter to this method is a column from protein sequences table.

The OGSA-DAI team are working with the GridMiner data mining project in order to help solve the problems of data-integration, and these are being implemented in DQP

OGSA-DAI has much more functionality than Stork. Although it can be more difficult to install, OGSA-DAI provides services for data identification, data manipulation and data transfer. It can be used with databases, XML or file systems (with more limited data access services), and it is able to select subsets of data from these data sources and then transfer the subset: this can be an advantage when using large files because Stork, as it is, could only transfer the entire data set. A requirement that the Data Services work package of the DataMiningGrid project must be able to meet involves performing SQL joins when accessing data from databases. OGSA-DAI can be used to integrate different databases, and apply such SQL statements across the two databases. It is not clear how functionality like this could be developed in Condor-Stork without a large commitment of time, resources and risk.

1.3.2. ELDAS (Enterprise Level Data Access Services)

Eldas [ELDAS04] is a core Edikt (e-Science Data Information and Knowledge Transformation) technology and aims to be a commercial quality software system that has been produced using industrial software engineering practices. It has been developed using J2EE and Enterprise Java Bean technologies, and it has adopted industry standard Web Services.

The Eldas project concentrates more on data integration rather than data access, and it provides solutions to data federation and performing joins on data from distributed, heterogeneous data sources. It provides access to heterogeneous, distributed data sch as Relational Database Management Systems (RDBMSs), XML databases, ASCII, and binary flat files (using BinX).

Davy Virdee of the Edikt team email a description of Eldas, and this is given here:

Eldas is "easy to use and install" - taking minutes to install, rather than hours, and is streamlined for users who want quick and easy access to databases via Web or grid services, albeit with constrained functionality: at present Eldas 1.0 allows SQL queries to be passed to MySQL databases. Eldas 1.1 (to be released late 2004) extends this functionality to allow connection to Oracle, DB2, SQLServer and Postgress databases; as well GSI message level security for Grid Services and HTTPS security for web services.

Its conception was originally based on the GDSS specification, but since this is a moving target, we have diversified slightly. For example, an Eldas Data Service Factory can connect to many data resources, whereas a Grid Data Service Factory has a one-to-one mapping with a particular data resource. We did not stick to "grid services", as we foresaw the emergence of WS-RF. Eldas is based solidly on Web Services and J2EE technologies, which are industry standards.

Eldas is intended for users who want to set up access to their databases with minimal effort, and is optimised for performance. One of the applications that uses Eldas is the EdSkyQuery project, an astronomy data mining project that handles Giga-bytes of data.

Eldas has been developed by Edikt, which is software engineering project funded by the Scottish Higher Education Council. Our "mission" is to facilitate Scottish "e-Science", and to this end, we developed Eldas for use in our applcations projects, and for general release to the academic and wider community.

FHG application farm and data sources

1. APPLICATIONS:

- **Weka** [Weka04]
  - Current usage:
    - ✓ Used in many data mining projects as stand-alone application
  - Reasons:
    - ✓ Wide range of data mining algorithms
    - ✓ Build-in workflow editor
    - ✓ Widely accepted
    - ✓ Open source
    - ✓ Easy to modify (good software design)
  - Technical requirements:
    - ✓ JavaVM 1.4
- **Text Mining Tool**
  - Current usage:
    - ✓ Used in text mining project with DC
  - Reasons:
    - ✓ Ability to efficiently train & classify large numbers text documents
    - ✓ Sophisticated methods for text mining
  - Technical requirements:
    - ✓ JavaVM 1.5
    - ✓ Fast harddrives
    - ✓ Min 20GB storage space

2. DATABASES:

Oracle is being considered as the database application for use at FHG if a database should be required. The reasons for this choice are:

- It is well supported by the manufacturer and many other companies.
- It is a highly flexible and powerful system providing many extensions and complying with all relevant standards.

DC application farm and data sources

1. APPLICATIONS

- **DC Text Mining Toolbox**
  - Current usage:
    - Used in several internal text mining and information retrieval projects
  - Reasons:
    - Provides efficient methods for text classification, clustering and retrieval
    - Developed and used over many years within research department
  - Technical requirements:
    - Written in plain C
    - Windows 2000/XP environment
    - 512MB RAM (application and data dependent)
    - 20GB storage space (application dependent)
    - Works with flat files on the filesystem, so no database technology is needed

2. DATABASES:

- **Dpa**:
  - about 250 classes
  - 4 years with about 400.000 texts each
  - German
  - there is a time stamp with the data, so detecting new topics can be processed (essential for QNA data)
  - short texts, therefore similar to the internal quality data from DC
  - combined processing possible with the Wortschatzprojekt (University of Leipzig) (german texts)
  - have to be purchased, FHG ask, whether it is possible to distribute the data within the DMGrid project

- **Reuters (700000 texts)**
  - there is a time sequence in the data.
  - Shorter than the Wipo data, therefore more similar to Dpa data
- **Wipo patent data**

- Patent collection of the World Intellectual Property Organization
- Several hundred classes hierarchically organized
- German and English
- 60.000 documents/year

DC internal data includes:

- **QNA (quality data)**
  - about 200 classes
  - 80.000 texts/day
  - short English texts (ca. 250 bytes)
  - Text Mining team at DC asks whether it is possible that FHG can use the data outside DC (confidential data)
- **Internal Patent collection (Derwent Collection)**
  - Text Mining team at DC asks whether it is possible that FHG can use the data outside DC (non confidential data, but not free available)

For each of this data there is no underlying database technology. The data is in flat files.

TECHNION application farm and data sources

APPLICATIONS:

- **Weka**, as in the farm of FHG.
- **Condor**
  - Current usage: more than 1000 pools around the world.
  - Reasons:
    - ✓ well supported (Annual funding of $5M).
    - ✓ stable.
    - ✓ contains data management and batch tools we can use.
    - ✓ open source.
    - ✓ easy to modify.
    - ✓ has versions consistent with Globus, Unicore, others.
    - ✓ portable across many OS and hardware.
  - Technical requirements: ractically nothing special.
- Optional: **OGSA-DAI**, currently still under study. Not yet clear how stable and how useful.

Condor [Condor04] is a specialized workload management system for compute-intensive jobs. Like other full-featured batch systems, Condor provides a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. Users submit their serial or parallel jobs to Condor, Condor places them into a queue, chooses when and where to run the jobs based upon a policy, carefully monitors their progress, and ultimately informs the user upon completion.

While providing functionality similar to that of a more traditional batch queuing system, Condor's novel architecture allows it to succeed in areas where traditional scheduling systems fail. Condor can be used to manage a cluster of dedicated compute nodes (such as a "Beowulf" cluster). In addition, unique mechanisms enable Condor to effectively harness wasted CPU power from otherwise idle desktop workstations. For instance, Condor can be configured to only use desktop machines where the keyboard and mouse are idle. Should Condor detect that a machine is no longer available (such as a key press detected), in many circumstances Condor is able to transparently produce a checkpoint and migrate a job to a different machine, which would otherwise be idle. Condor does not require a shared file system across machines - if no shared file system is available, Condor can transfer the job's data files on behalf of the user, or Condor may be able to transparently redirect all the job's I/O requests back to the submit machine. As a result, Condor can be used to seamlessly combine all of an organization's computational power into one resource.

Condor-G [Condor-G04] is the job management part of Condor. Condor-G lets you submit jobs into a queue, have a log detailing the life cycle of your jobs, manage your input and output files, along with everything else you expect from a job queuing system. Condor-G gets its name from how it talks to the resource management part. Instead of using the Condor-developed protocols to start running a job on a remote machine, Condor-G uses the Globus Toolkit(tm) to start the job on the remote machine. Condor-G provides a "window to the Grid" for users to both access resources and manage jobs running on remote resources. Use Condor-G to look across the Grid and see instantly how your jobs are doing. You can trust Condor-G to keep watching your jobs while your attention is elsewhere.

The Condor-G system leverages recent advances in two distinct areas: (1) security and resource access in multi-domain environments, as supported within the Globus Toolkit, and (2) management of computation and harnessing of resources within a single administrative domain, embodied within the Condor system. Condor-G combines the inter-domain resource management protocols of the Globus Toolkit and the intra-domain resource and job management methods of Condor to allow the user to harness multi-domain resources as if they all belong to one personal domain.

Condor-G provides the grid computing community with a powerful, full-featured task broker. Used as a front-end to a computational grid, Condor-G can manage thousands of jobs destined to run at distributed sites. It provides job monitoring, logging, notification, policy enforcement, fault tolerance, credential management, and it can handle complex job-interdependencies. Condor-G's flexible and intuitive commands are appropriate for use directly by end-users, or for interfacing with higher-level task brokers and web portals.

LJU application farm and data sources

1. APPLICATIONS:

- Automated modelling framework: Lagramge application

- Applications used for processing medical databases:
  - Prodi 5.0 (www.nutri-science.de/de/nutri-science/aktuell/index.htm)
  - SPSS 12.0 – statistical programme http://www.health.state.mo.us/dnhs_pdfs/R_NPE_M5-05_instruct.pdf
  - FFQ more information on:http://ffq.fhcrc.org/pdf/msel_sample.pdf, , http://www.bus-sante.ch/alim_visu.html, http://www.abdn.ac.uk/deom/ffq/, http://www.srl.cam.ac.uk/epic/nutmethod/7dd.shtml, http://menu.pbrc.edu/ffq/

2. DATABASES

Usual data base technologies, e.g. MS Access, mySQL, Oracle or similar.

Appendix B: Detailed schematic use cases descriptions

These use cases were collected from a selected set of real-world applications from different domains. These applications will be used as demonstrations and testing of the DataMiningGrid components. More use cases shall be collected by the end-users group which is currently under formation.

1. UU use cases:

1.1. Identify and select available grid data services

Identifier	*Identify and select available grid data services*
Goals in Context	*Identifying available grid data services and selecting grid data services.*
Actors	*User, user interface (client-side application), grid middleware, grid data service registry*
Triggers	*Need to select grid data service*
Included Use Cases	*Data request*
Specialised Use Cases	
Pre-conditions	• *Grid data service registry is known and online* • *Grid data service factories have registered with the grid data service registr* • *Grid data services have access to datasets.*
Post-conditions	
Basic Flow	1. *User accesses list of grid data services held in grid data service registry.* 2. *Client-side application sends query to grid data service registry requesting available grid data services and associated metadata.* 3. *Client-side application displays list of matching grid data services and associated metadata to user.* 4. *Client-side application contacts data services to establish their capabilities.* 5. *User selects grid data services to use.* 6. *Client-side application stores grid service handle as part of workflow.*
Deviant Flow(s) (non-exhaustive)	*Failure – Access to grid data service registry is denied, network communication is down, grid data service registry is not working.* *Unable to fulfil request – no grid data services are available.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 1: Identify and select available grid data services

1.2. Requesting data from grid data service

Identifier	*Requesting data from grid data service*
Goals in Context	*Requesting protein data by user from single grid data service*
Actors	*User, user interface (client-side application), grid middleware, grid data service registry, grid data service, dataset, data consumer*
Triggers	*Need to select data and transfer to data consumer*
Included Use Cases	*Data services: provide data selection and access*
Specialised Use Cases	
Pre-conditions	• *User has access rights to data service* • *Client application has grid service handle for grid data service* • *Grid data service is accepting requests for grid data service instance.*
Post-conditions	*Data consumer accepts grid data transfer*
Basic Flow	1. *User initiates compilation and execution of workflow script containing requests to access grid data* 2. *Client-side application connects to grid data* 3. *Client-side application sends a query for data to the grid data service.* 4. *The grid data service executes the query and accesses the requested data* 5. *The data is formatted into a standard data representation* 6. *The data is transferred to the data consumer*
Deviant Flow(s)	*Failure – Client-side application is denied access to grid data service, network communication is down, grid data service is not working* *Unable to fulfil request – grid data service is unable to process request because it is malformed or it requests data that cannot be supplied by grid data service* *Grid data service failure – grid data service is unable to access datasets containing the data it requires, the datasets are malformed or not working* *Data transfer failure – grid data transfer fails because client application will not accept the data, network communication is down, client application is not working*
Importance and Frequency	*Extremely important – no data means no data mining.*
Additional Requirements	*Grid data services workflow templates must be constructed either manually or using an automated process.*

Use case 2: Requesting data from grid data service

Data mining application near data source

Identifier	*Data mining application near data source*
Goals in Context	*Processing data at the site of the data source*
Actors	*User, user interface (client-side application), grid middleware, compute job scheduler, a prepared compute job, a data source, grid data service registry*
Triggers	*Users wants to analyse data sources*
Included Use Cases	*Data mining*
Specialised Use Cases	*Data mining at the data source*
Pre-conditions	• *User has logged on to the service registry* • *Data sources exist that allow compute jobs to be uploaded and executed on them* • *Grid middleware is available, a scheduler for compute jobs is available,* • *User has read, write and execute permissions on the data source* • *User has prepared a compute job that consists of data access, selection, and processing operation* • *Only a single data source is queried.*
Post-conditions	*Processed data can be viewed or transferred to another system of the Users choice*
Basic Flow	1. *User selects and identifies a data source that is also a compute resource* 2. *User submits a compute job to the scheduler and specifies the machine that it must run on* 3. *The compute job is uploaded to the data source by the scheduler* 4. *The compute job is executed locally on the data source* 5. *The job can be monitored by the User at any time* 6. *The user is informed when the job is finished*
Deviant Flow(s) (non-exhaustive)	*Failure – access to the data source is denied, execution rights at the data source are denied, network communication is down, the compute job scheduler is down, the compute job crashes, compute resources on the data source are insufficient to execute the compute job*
Importance and Frequency	*Important and frequent for some applications, not at all important for other applications*
Additional Requirements	

Use case 3: Data mining application near data source

2. FHG use cases:

2.1. Finding analysis services

Identifier	*Finding analysis services*
Goals in Context	*Exporting information about available grid servers and analysis services offered by them*
Actors	*User, User interface (e.g. Workflow Editor), Administrator, LDAP server with data stored in it*
Triggers	*Needed by workflow to collect data about servers, offered services and resources available*
Included Use Cases	
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the server (maybe anonymous bind?)* • *User interface knows about data structure* • *Information shared by server are prepared by Administrator or servers are enabled to register automatically on their own*
Post-conditions	*List of all analysis services currently including descriptions of them (e.g. input data, output data, etc.) available*
Basic Flow	1. *Server keeps data concerning available analysis services* 2. *User (or any program) issues search for available services* 3. *Requested data are returned and presented to user and/or used during construction of submission scripts*
Deviant Flow(s)	*Failure:* • *Server access is denied -> user must provide correct authentication / authorization* • *Network communication is down -> try again after appropriate period of time has passed or provide "last known configuration" to construct workflow chains "offline"*
Importance and Frequency	*Extremely important – server is asked by user interface at least once during every execution*
Additional Requirements	*Mechanism of updating available data – manually by server administrator or by set of automatic scripts*

Use case 4: *Finding analysis services*

Workflow editor

Identifier	*Workflow editor*
Goals in Context	*Definition of workflow and production of job submission script*
Actors	*User, user interface (client-side application), data source(s), grid-middleware*
Triggers	*Need to define the actions for a data mining task*
Included Use Cases	• *FindAnalysisServices* • *FindDataServices (UU)*
Specialised Use Cases	
Pre-conditions	• *User has local workflow editor* • *User has authentication / authorization for data access* • *Grid middleware installed on all participating machines*
Post-conditions	*Workflow chain itself and generated workflow script are stored client-side and/or server-side for later modifications, re-runs, etc.*

Basic Flow	*1. User starts local workflow editor (client-side)*
	If (new data mining task)
	2. Workflow requests available data and analysis services
	3. User defines workflow graphically (selection of data sources, pre-processing, dm-algorithms)
	4. Editor validates workflow chain (logically, access to data sources)
	5. Editor produces appropriate script for job submission
	Else
	6. User loads old workflow
	7. Editor validates workflow chain (logically, access to data sources)
	End If
	8. User saves the workflow on the client and/or the server
	9. Transfer of script to a designated server for execution
	10. Execution of script on server

Deviant Flow(s) (non-exhaustive)	*Failure:* • *Chain incorrect -> User must revise workflow chain* • *Data access denied -> User must specify correct authentication / authorization* • *Server not ready -> e.g. restart server*
Importance and Frequency	*Extremely important* *Workflow editor will be used every time a data mining task is to be executed.*
Additional Requirements	

Use case 5: *Workflow editor*

2.2. Pre-process documents for specific classifier

Identifier	*Preprocess documents for specific classifier*
Goals in Context	*Produce input features for a document classifier/clusterer*
Actors	*User, Grid middleware, User interface (client-side application), Server-side scripts, text mining module*
Triggers	*TrainSpecificClassifier, UseClassifierForSpecificClassAttribute, ClassifyDocument, ClusterDocuments*
Included Use Cases	*Workflow*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the server* • *Rquested data can be accessed by server-side scripts* • *Gid middleware is available on both sides* • *Taining data is available(on server or on client grid computer)* • *Optionally: news database has been preprocessed (see use case 61.3-3)* • *Cnversion from server-side data format and the format requested by the user is available*
Post-conditions	*Requested data are stored on the server in format specified by the user and are available to download by any GridFTP-enabled application*
Basic Flow	13. *User logs in to the server by GUI* 14. *Client-side application lists kind and range of data stored on the servers* 15. *User specifies training data on different servers* 16. *Optionally: User specifies target class.* 17. *The user specifies a procedure for selecting data (e.g. sampling equal number of positive/negative examples for a document class).* 18. *The data is selected, possibly from databases/files on different servers and stored on these servers.* 19. *The list of primary terms is extracted on the servers (words, …),* 20. *Optional: Primary terms are transformed/enhanced by stemming, stopword omission, ngram generation, character quadgram generation, syllable generation, chunking, POS-tagging, latents semantic analysis, ontology lookup on the servers yielding new terms* 21. *Counts of terms are determined on the servers.* 22. *Optional. Term lexica are compiled of a specific server.* 23. *Optional: The relevant terms are selected on the specific server (feature selection).* 24. *Optional: New weights for terms are determined on the specific server (e.g. tfidf).* 25. *Counts of terms are reweighted on each of the servers.* 26. *New terms are stored on each server..* 27. *Optionally status of job can be checked by the User at any time* 28. *User is informed when the job is finished*
Deviant Flow(s)	*Failure – Server access is denied, network communication is down, grid*

(non-exhaustive)	*middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 6: Preprocess documents for specific classifier

2.3. Train specific classifier

Identifier	*Train specific classifier*
Goals in Context	*Produce news stories classifier*
Actors	*User, Grid middleware, User interface (client-side application), Server-side scripts, text mining module*
Triggers	*Use Classifier For Specific Class Attribute, Classify Document*
Included Use Cases	*Workflow, Pre-process Documents For Specific Classifier*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the serve* • *requested data can be accessed by server-side scripts* • *grid middleware is available on both sides* • *training data is available(on server or on client grid computer)* • *Optional: document collection has been (partially) preprocessed (use case PreprocessDocumentsForSpecificClassifier)* • *conversion from server-side data format and the format requested by the user is available*
Post-conditions	*Requested data are stored on the server in format specified by the user and are available to download by any GridFTP-enabled application*
Basic Flow	1. *User logs in to the server by GUI* 2. *Client-side application lists kind and range of data stored on the servers* 3. *User specifies training data, possibly on different servers* 4. *The user specifies a procedure for selecting data (e.g. use case PreprocessDocumentsForSpecificClassifier).* 5. *Optionally: The selection procedure is specialized to a specific fold of a cross validation run.* 6. *For each class the data is selected - possibly from databases/files on different servers - and sent to a server dedicated to train the specific classification model (if necessary),*

	7. *The classifier is estimated on the specific server.* 8. *Statistics on classifier performance on the training data are collected and stored.* 9. *Optional: The classifier output is transformed to a probability.* 10. *Optional: Estimate classifier performance by cross validation.* 11. *The classifier data structure is stored in repository together with the list of primary terms, terms and weights.* 12. *Optionally status of job can be checked by the User at any time* 13. *User is informed when the job is finished*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 7: *Train specific classifier*

2.4. Use classifier for specific class attribute

Identifier	*Use classifier for specific class attribute*
Goals in Context	*Classify documents*
Actors	*User, Grid middleware, User interface (client-side application), Server-side scripts, text mining module*
Triggers	
Included Use Cases	*Workflow, TrainSpecificClassifier, ClassifyDocument*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the server* • *Requested data can be accessed by server-side scripts* • *Grid middleware is available on both sides* • *Classification model has been trained* • *News database has been preprocessed* • *Conversion from server-side data format and the format requested by the user is available*
Post-conditions	*Requested data are stored on the server in format specified by the user and are available to download by any GridFTP-enabled application*
Basic Flow	1. *User logs in to the server by GUI*

	2. *User specifies unlabelled test data on different servers.* 3. *The data is selected, possibly from databases/files on different servers and stored on these servers.* 4. *The stored pre-processing module and the trained classifier is transmitted to the servers holding the unlabelled data.* 5. *The selected data is pre-processed on the servers and derived terms are stored on the servers.* 6. *The classifiers is applied and the class probability/score is calculated on the servers and stored on these servers.* 7. *Optionally status of job can be checked by the user at any time* 8. *User is informed when the job is finished*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 8: Use classifier for specific class attribute

2.5. Classify documents

Identifier	*Classify document*
Goals in Context	*Classify documents*
Actors	*User, Grid middleware, User interface (client-side application), Server-side scripts, text mining module*
Triggers	*UseClassifierForSpecificClassAttribute*
Included Use Cases	*Workflow, TrainSpecificClassifier*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the server* • *Requested data can be accessed by server-side scripts* • *Grid middleware is available on both sides* • *Unlabelled test data is available* • *Class scores/probabilities for each class attribute have been determined* • *Conversion from server-side data format and the format requested by the user is available*

Post-conditions	*Requested data are stored on the server in format specified by the user and are available to download by any GridFTP-enabled application*
Basic Flow	1. *User logs in to the server by GUI* 2. *User specifies unlabelled test data with scores/probabilities on possibly different servers* 3. *The decision module collects the scores/probabilities for each document from the servers.* 4. *The decision module assigns one class, several classes or no class to each document and optionally computes class probabilities.* 5. *The assigned classes/probabilities are stored on the specific server (or the originating servers).* 6. *Optionally: The results for one or more documents are collected. Summary statistics and document contents and assigned classes are visualized on a specific client.* 7. *Optionally status of job can be checked by the User at any time* 8. *User is informed when the job is finished*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to acomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 9: *Classify documents*

2.6. Cluster documents

Identifier	*Cluster documents*
Goals in Context	*Produce clusters of Documents*
Actors	*User, Grid middleware, User interface (client-side application), Server-side scripts, text mining module*
Triggers	
Included Use Cases	*Workflow, PreprocessDocumentsForSpecificClassifier*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the server* • *Requested data can be accessed by server-side scripts* • *Grid middleware is available on both sides* • *Training data (unlabeled) is available*

	• *Conversion from server-side data format and the format requested by the user is available*
Post-conditions	*Requested data are stored on the server in format specified by the user and are available to download by any GridFTP-enabled application*
Basic Flow	1. *User logs in to the server by GUI* 2. *Client-side application lists kind and range of data stored on the servers* 3. *User specifies (unlabeled) training data on different servers.* 4. *The user specifies a procedure for selecting and preprocessing of data (e.g. use case PreprocessDocumentsForSpecificClassifier)* 5. *The data is selected and pre-processed on different servers.* 6. *The cluster procedure (and possibly the data) is distributed on suitable servers and one iteration is performed.* 7. *Cluster processes synchronize on a central server* 8. *Check on the central server if results are sufficient, else go to 6..* 9. *The cluster attributes are stored on the central server or with the documents on the different servers.* 10. *Optionally: Cluster results are collected on a central client and visualized with a specific GUI.* 11. *Optionally status of job can be checked by the User at any time* 12. *User is informed when the job is finished*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 10: *Cluster documents*

2.7. Access learned ontology

Identifier	*Access learned ontology*
Goals in Context	*Access Ontology / Wortschatz*
Actors	*User, Grid middleware, User interface (client-side application), Server-side scripts, text mining module*
Triggers	*ClusterDocuments*
Included Use Cases	*Workflow*
Specialised Use	

Cases	
Pre-conditions	• *User has the key accepted by the server* • *Requested data can be accessed by server-side scripts* • *Grid middleware is available on both sides* • *The ontology (i.e. cluster attributes) are stored on the server* • *Conversion from server-side data format and the format requested by the user is available*
Post-conditions	
Basic Flow	1. *User logs in to the server by GUI* 2. *User specifies query data for the ontology* 3. *the ontology module collects the requested items on the server* 4. *the collected data are sent back to the user*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*
Additional Requirements	

Use case 11: *Access learned ontology*

3. DC use cases

3.1. Pre-processing distributed document collections

Identifier	*Pre-processing distributed document collections*
Goals in Context	*Exporting information about available grid servers and services offered by them*
Actors	*Expert, Administrator, Grid middleware, User interface (client-side application), Document Servers, Index Server, Server-side scripts, text mining modules(server-side application)*
Triggers	*Needed by User Interface to collect data about servers, offered services and resources available*
Included Use Cases	*Data mining: Checking of available grid resources by user interface*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the server* • *User interface knows about data structure*

	• *Information shared by server are prepared by Administrator or servers are enabled to register automatically on their own* • *Requested data can be accessed by server-side scripts* • *Grid middleware is available on all acting client and server sides* • *Optional: Ontology has been learned* • *Document collections on document servers have been registered to (cf. Use Case: Register document server)* • *Conversion from server-side data format and the format requested by the user is available*
Post-conditions	*Requested data are stored on the servers in format specified by the user and are available to download by any GridFTP-enabled application*
Basic Flow	1. *User logs in to the server by GUI* 2. *Client-side application lists kind and range of data stored on the servers* 3. *User specifies document collection on different servers* 4. *Terms to be evaluated (Words, Quadgrams) are determined* 5. *Optional: Primary terms are transformed/enhanced by stemming, stopword omission, ngram generation, chunking, POS-tagging, latent semantic analysis or ontology terms on the specific server yielding new terms.* 6. *Counts of terms are determined on each server separately.* 7. *The term lexica are merged on a specific server* 8. *The relevant terms (feature selection) are selected on the specific server.* 9. *Optional: Counts of terms are reweighted, e.g. by tfidf on each server separately by the information gained in 8.* 10. *Inverted files are generated on each server to be accessed by use case GridWhatsRelatedSearchEngine or the use cases for classification and clustering* 11. *Optionally status of job can be checked by the User at any time* 12. *User is informed when the job is finished*
Deviant Flow(s)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Extremely important – basic use case for the others in T61*
Additional Requirements	

Use case 12: Preprocessing distributed document collections

Grid whats related search engine

Identifier	*Grid whats related search engine*
Goals in Context	*User satisfies information need on distributed technical document collections, e.g. patents*
Actors	*User, Grid middleware, User interface (client-side application), Document Servers, Index Server, Server-side scripts, text mining modules(server-side application)*
Triggers	*Need to choose which of all available data should be prepared and downloaded by the user or any grid application*
Included Use Cases	*By Preconditions:* • *Register document server to GridWhatsRelated* • *Pre-process Document collections* • *Compute Central Index* *By Basic Flow:* • *Access Learned Ontology* • *Access Wortschatz* • *Access Data on the Grid*
Specialised Use Cases	
Pre-conditions	• *User has the key accepted by the Index Server and some or all of the document servers* • *Requested data can be accessed by server-side script* • *Grid middleware is available on all acting client and server sides* • *Optional: Ontology has been learned* • *Document collections on document servers have been registered to Grid what is related and pre-processed (cf. Use Cases: Register document server, and: Pre-process Distributed Document collections)* • *Central Index on Index Server has been computed (cf. Use Case: Compute Central Index)* • *Conversion from server-side data format and the format requested by the user is available*
Post-conditions	*User has list of URIs of matching documents/passages with relevancy values and these documents are accessible by their URI by any GridFTP enabled application*

Basic Flow	1. *User logs in to the system by GUI*
	2. *User formulates and inputs query either*
	  a. *as list of keywords, or*
	  b. *by providing a document in fulltext*
	  c. *as URI of a known document from any of the document servers*
	  d. *as URI of a so far unknown document from any accessible server*
	3. *Query is sent to Index Server*
	4. *Index Server computes on his index which document servers the query should be sent to (i.e. which document servers could carry matching*

	documents) OR computes the set of matching documents (this depends on the kind of index!!) In the latter case, jump to step 8. *5. Index server distributes the user query to the set of document servers which come into question and waits for answers* *6. Each document server, which receives user query from index server does:* *a. Check, if user has necessary access rights to access this servers documents* *b. If yes to a., query local index and compute ranked result list consisting of URIs to matching documents/passages* *c. Send local result list back to index server* *7. Index server collects the local result lists and computes a global ranking of the result set* *8. Index server sends the final result set back to the user's client.* *9. Present the result set to the user in a GUI.*
Deviant Flow(s) (non-exhaustive)	*Optional after BF Step 2: Automatic or interactive Query Expansion takes place by accessing the learned ontology (see Use Case: Access Learned Ontology) or by querying Wortschatz (see Use Case: Access Wortschatz). Expanded Query is the input to Step 3.* *If successful, optionally after Step 9: User accesses documents given by result list (see Use Case: Accessing Data on the Grid by URI) OR users starts over again by taking a document URI from the result list as input to new query.* *Optionally during steps 3 to 8: Status of query job can be checked by the User at any time OR user is informed in regular intervals on the progress of the query job* *Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – Index server not reachable, requested data are not stored on the server, data cannot be accessed by server-side scripts.* *Able to accomplish request partially – some of indexed document servers are reachable, others are not. In this case, Index server should compute a partial result set and tell user which document servers are unreachable or unavailable.*
Importance and Frequency	*Important* *Frequency depends on user's information need, appears at arbitrary times*

Use case 13: *Grid what's related search engine*

TECHNION use cases:

3.2. System diagnosis

Identifier	*System diagnosis*
Goals in Context	*System functioning improvements via system condition analysis.*
Actors	• *Collector (resource responsible for collecting availability of resources),* • *WiseMan (resource responsible for updating the factual availability of resources),* • *Resource (individual node in the network, also referred to as "individual resource"), and* • *Negotiator (resource responsible for submitting jobs).*
Triggers	*Scheduled self-triggering.*
Included Use Cases	*None.*
Specialised Use Cases	*None.*
Pre-conditions	*Middleware running with unknown condition.*
Post-conditions	*Complete diagnosis of the system.*
Basic Flow	1. *Logs are parsed locally on the resources by a component included in the middleware, and analysed to produce analysed data.* 2. *Analysed data are pushed from an individual local resource bottom-up to some other resources where these data are re-analysed by including analysed data from other individual resources.* 3. *2 is repeated moving bottom-up until the WiseMan.* 4. *WiseMan finalizes the data analyses, and provides the diagnosis of the system.*
Deviant Flow(s) (non-exhaustive)	*None.*
Importance and Frequency	*Critical importance.* *Frequency is customisable.*
Additional Requirements	*None.*

Use case 14: *System diagnosis*

3.3. Data mining workflow generation

Identifier	*Data mining workflow generation*
Goals in Context	*Executing a data mining workflow design automatically.*
Actors	• *User,* • *Distributed Data Sources, and*

	• *Workflow Generator.*
Triggers	*User who designed a desired data mining workflow wants to execute it.*
Included Use Cases	*None.*
Specialised Use Cases	*None.*
Pre-conditions	• *Data mining workflow design completed* • *the data mining workflow must comply with the data sources and data available in a specific network.*
Post-conditions	*The application ready to be executed provided to the user.*
Basic Flow	5. *The data mining workflow design is submitted to the Workflow Generator in an internal formalism.* 6. *The Workflow Generator parses the design, and transforms it into a series of executable steps. Each step is further transformed into jobs or commands that can be executed on the specific middleware.* 7. *The resulting jobs and commands are provided to the user.*
Deviant Flow(s) (non-exhaustive)	*The data mining workflow design is not fully compliant with a specific network due to the lack of knowledge regarding the available data sources. – The Workflow Generator is required to interact with the user providing the missing information in order to allow the user to complete the design.*
Importance and Frequency	*Very important.* *Launched on demand.*
Additional Requirements	*Data mining workflow design provided from the application provided by FHG.*

Use case 15: *Data mining workflow generation*

4. LJU use cases

4.1. Lake modelling: model induction

Identifier	*Lake modelling: model induction*
Goals in Context	*Induction of aquatic ecosystem models*
Actors	*User, Grid middleware, User interface (client-side application), Database Servers*
Triggers	*induction finished/failed*
Pre-conditions	
Post-conditions	*models induced and stored in the data base*
Basic Flow	1. *Transferring the data set to the server data base for storing data sets* 2. *Transferring the task specification to the server* 3. *Run application (Lagramge)*

	4. *Report the results: the equations will be written in form that allows user to recognise which process models are involved in the model. The models discovered are stored in a data base of models*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done, user has no rights to access data.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*

Use case 16: *Lake modeling: model induction*

4.2. Lake modelling: simulations

Identifier	*Lake modelling: simulations*
Goals in Context	*Simulation of aquatic ecosystem models*
Actors	*User, Grid middleware, User interface (client-side application), Database Servers*
Triggers	*simulation finished/failed*
Included Use Cases	*Lake Modelling: induction of models*
Pre-conditions	• *Data set (for initial values of the system variables as well as values of input/output variables)* • *Model* • *Compatibility between model and data set: all the model variables should appear in the model*
Post-conditions	*simulation stored in the data base*
Basic Flow	1. *User uploads a data set and a model to the server (or choose existing ones)* 2. *Run the simulation code* 3. *Store the results in the data base of data sets*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done, user has no rights to access data.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*

Use case 17: *Lake modeling: simulations*

4.3. Process medical data in distributed databases

Identifier	*Process medical data in distributed databases*
Goals in Context	*Analysis of distributed medical databases for endemic goitre and iodine deficiency studies*
Actors	*User, Grid middleware, User interface (client-side application), grid data service, grid dataset*
Triggers	
Included Use Cases	*Workflow, Requesting data from grid data service*
Pre-conditions	• *User has the key accepted by the server* • *Requested data can be accessed by server-side scripts* • *Grid middleware is available on both sides* • *Conversion from server-side data format and the format requested by the user is available*
Post-conditions	*Calculated decision tree models are stored in database and are available to other users.*
Basic Flow	1. *User logs in to the server by GUI (SECURITY, PRIVACY)* 2. *User specifies data on different servers.* 3. *The data is selected, possibly from databases/files on different servers and stored on these servers.* 4. *The stored pre-processing (checking for illogical data) module is transmitted to the servers holding the unlabelled data.* 5. *The selected data is processed on the servers (with Weka) and derived results are stored on the servers.* 6. *Optionally status of job can be checked by the user at any time*
Deviant Flow(s) (non-exhaustive)	*Failure – Server access is denied, network communication is down, grid middleware is not working at any side.* *Unable to accomplish request – requested data are not stored on the server, data cannot be accessed by server-side scripts, data format conversion cannot be done, user has no rights to access data.* *Able to accomplish request partially – some of requested data are available, but some are not.*
Importance and Frequency	*Important*

Use case 18: Process medical data in distributed databases