Digging Deep into the Data Mine with DataMiningGrid

As modern data mining applications increase in complexity, so too do their demands for resources. Grid computing is one of several emerging networked computing paradigms promising to meet the requirements of heterogeneous, large-scale, and distributed data mining applications. Despite this promise, there are still too many issues to be resolved before grid technology is commonly applied to large-scale data mining tasks. To address some of these issues, the authors developed the DataMiningGrid system. It integrates a diverse set of programs and application scenarios within a single framework, and features scalability, flexible extensibility, sophisticated support for relevant standards and different users.

Vlado Stankovski and Jernej Trnkoczy
University of Ljubljana

Martin Swain and Werner Dubitzky
University of Ulster

Valentin Kravtsov and Assaf Schuster
Israel Institute of Technology

Thomas Niessen, Dennis Wegener, and Michael May
Fraunhofer Institute for Intelligent Analysis and Information Systems

Matthias Röhm and Jürgen Franke
DaimlerChrysler

Data mining facilitates the automated extraction of potentially useful information from large volumes of data. Expanding data volumes and the geographic distribution of applications in many modern knowledge sectors are fueling the need for novel data mining solutions. Recent trends that leverage network-based infrastructures include distributed data mining in peer-to-peer networks, mobile and embedded devices, sensor networks, and both parallel and privacy-preserving data mining.

Data mining in grid[1] computing environments represents a specific incarnation of distributed data mining motivated by resource sharing via local and wide area networks.[2] Increased performance, scalability, access, and resource exploitation are the key drivers behind such endeavors. However, several factors hamper large-scale data mining applications on a grid. To begin with, grid computing is relatively new, and relevant standards and technologies are still evolving. A plethora of data mining technologies and a staggering number of largely varying data mining application scenarios further complicate matters. Finally, data mining clients range from highly domain-oriented end users to technology-aware specialists. To the former, user transparency and ease-of-use is paramount, whereas the latter group needs control of certain detailed aspects of data mining and grid technology.

In the DataMiningGrid project (www.dataminingrid.org), we've aimed to address the requirements of

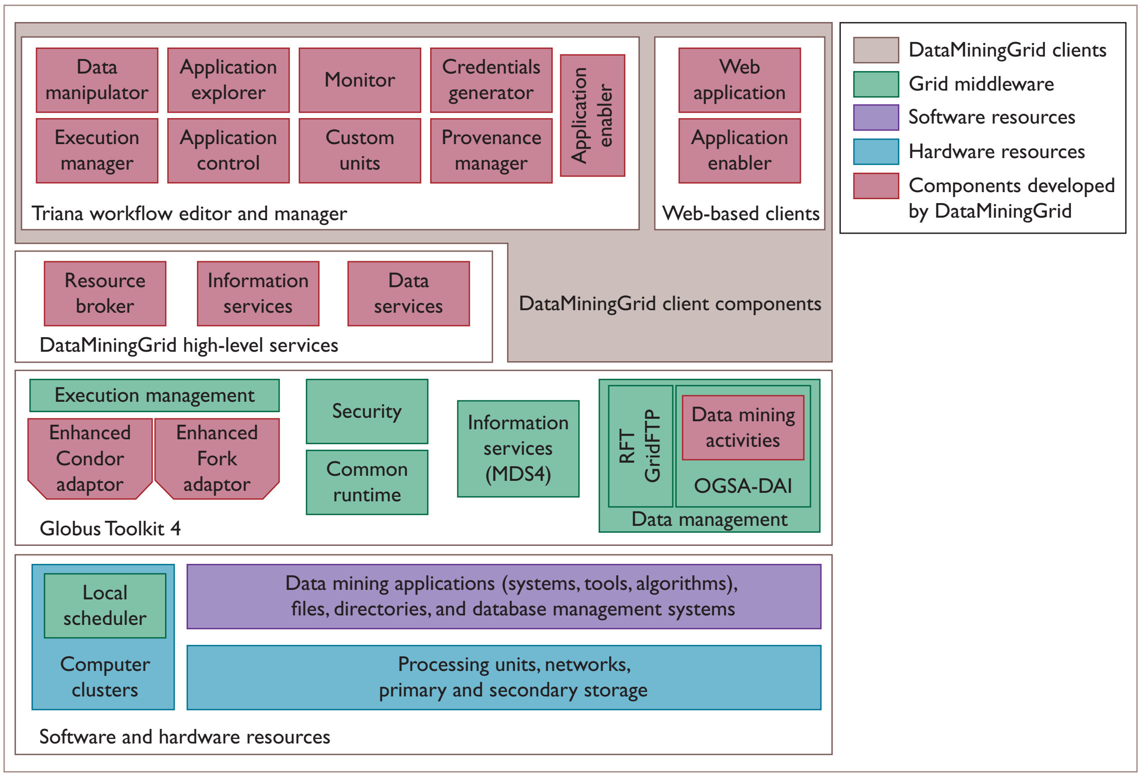

Figure 1. The DataMiningGrid system architecture in four layers. Generally, components in higher layers use the components organized in lower layers. The software and hardware resources layer sits at the bottom, the Globus Toolkit 4 layer depicts some of the system's core grid middleware components, the high-level services layer shows components providing central DataMiningGrid services, and the client components layer depicts the DataMiningGrid applications' client-side components.

modern data mining application scenarios — in particular, those that involve sophisticated resource sharing.[3]

Use Cases and Requirements

In order to determine the requirements of a generic grid system that will support demanding data mining applications, we analyzed use case scenarios from a wide range of application areas. These included ecological modeling, computational biology and bioinformatics, customer relationships and quality management in the automotive industry, performance monitoring and fault diagnosis in network computing systems, and analysis of distributed data in large-scale medical studies. We faced key challenges integrating the requirements of all these complex application scenarios into a single framework:

- *Grid-enabling data mining applications shouldn't require modification of source code.* We had to grid-enable numerous data mining applications, ranging from the well-known toolkit Weka, which runs on a Java Virtual Machine (JVM), to proprietary machine-learning, text mining, and ontology-learning applications that only execute on specific platforms and use special libraries. Bearing in mind the diversity of the application scenarios, we couldn't restrict the system to specific data mining applications, tools, techniques, or algorithms. It also had to support various types of data sources, including database management systems and data sets stored in file systems.
- *Efficiency, novel use, and improved resource exploitation.* Our grid-based data mining environment had to offer one or more of the following benefits: increased performance for data mining applications (speed-up, throughput), high scalability to serve more users and more demanding applications, possibilities for the creation of novel workflow-based data mining applications (for example, combining several data mining tasks to cover preprocessing, processing, and postprocessing stages), and improved exploitation of existing hardware and software resources.

- *Usability.* To ensure that the system was easy to use, we had to hide the grid's intricate technological details from domain-oriented users, but we also provided ways for technically knowledgeable users to define, configure, and parameterize application details.

The collected requirements implied various complex system components, such as a workflow editor and manager, a resource manager, grid-based data management, security mechanisms, an execution management system, and so on. To speed up development and reduce costs, we based the system design on existing open technology.

DataMiningGrid System and Features

To address our requirements, the DataMiningGrid system is designed according to three principles: service-oriented architecture (SOA), standardization, and open technology. SOA promotes the sharing of geographically dispersed business functions in a flexible way. We exploited SOA's main advantages, such as loose coupling, ease and flexibility of reuse, scalability, interoperability, and service abstraction from underlying technologies.[4] To ensure the DataMiningGrid system's seamless evolution, we made it support two important distributed computing standards: the Open Grid Service Architecture (OGSA, www.globus.org/ogsa) and the Web Services Resource Framework (WSRF, www.oasis-open.org).

Figure 1 depicts the DataMiningGrid system architecture in four layers. Generally, components in higher layers use the components organized in lower layers. The *software and hardware resources layer* sits at the bottom, the *Globus Toolkit 4 layer* depicts some of the system's core grid middleware components, the *high-level services layer* shows components providing central DataMiningGrid services, and the *client components layer* depicts the DataMiningGrid applications' client-side components.

Application Description Schema

A metadata schema definition, which we call an Application Description Schema (ADS), manages user interaction with system architecture components to grid-enable existing data mining applications, and helps register and search for software on the grid, match jobs with suitable computational resources, and dynamically create user interfaces. Instances of the ADS are XML documents at various levels of specification, which are passed among system components.

A fully specified instance of the ADS contains

- *general information*, that is, the application's narrative description, program vendor, version, and so on;
- *implementation aspects*, such as the application's operating system (Windows or Linux), programming language, executable files and their storage locations, and requirements, such as CPU architecture, memory, and disk space;
- *parameters* that define the data mining application's execution behavior (at runtime, these parameters specify different command-line options and can help create multiple simultaneous executions); and
- *data mining aspects* to facilitate the application's fast discovery on the grid, including the domain-specific problem addressed; the data mining task, method, algorithm, and software; and the CRoss Industry Standard Process (CRISP) for the application's data mining phase (www.crisp-dm.org).

The ADS is expressive enough to accommodate data mining applications from a wide range of platforms, technologies, application domains, and sectors, and has been tested with our eight different application domains.

Software and Hardware Resources Layer

Software resources include data resources, such as database and file systems, and data mining applications. Typical hardware resources include processing units, primary and secondary storage devices, and computer clusters.

Globus Toolkit 4 Layer

The Globus Toolkit 4 (GT4) layer of the DataMiningGrid architecture provides core grid middleware functionality.[5] GT4 is an open source toolkit for building grid systems provided by the Globus Alliance, which meets the requirements of the OGSA and implements the WSRF. It provides several critical components. GT4's Monitoring and Discovery System 4 (MDS4) is essentially a system for storing and searching dynamically changing distributed XML documents that describe the grid's software and hardware resources and their status. GT4's data management components include grid file transfer functionality (GridFTP, Reliable File Transfer [RTF] service) and data services (OGSA-DAI[6]). The GT4 data access and integration tools (OGSA-DAI component) provide grid-enabled access to files, relational databases, and XML databases. In addition to GT4 data management functions, we use the Java Commodity Grid Kit[7] to let application client machines, which don't have a GridFTP server installation, manipulate and transfer files to and from grid storage servers.

In addition, the DataMiningGrid makes extensive use of GT4's execution management tools to handle the initiation, monitoring, management, and coordination of remote computations. GT4 is used as a front end to either single machines or computational clusters, but it doesn't implement a global scheduling functionality per se. GT4 provides a Web service version of the Grid Resource Allocation and Management (WS-GRAM) interface, which is responsible for either the job's execution on the local single machine or for forwarding and controlling execution of the local cluster manager, such as Condor, load sharing facility, and so on. To overcome the relatively limited abilities of WS-GRAM to interact with local cluster managers, the DataMiningGrid enhances WS-GRAM's execution adaptors by adding data transfer functionality to deal with recursive directory structures and by enabling the execution of Java applications, including the proper handling of JVM parameters, class path information, and so on.

DataMiningGrid High-Level Services

In the DataMiningGrid system, a *job* refers to a DataMiningGrid-enabled data mining application (executables and associated libraries) that needs data files (data from databases must be converted to files before job submission) and appropriate computing resources to be executed, whereas *multi-jobs* are collections of such jobs (for example, parameter sweeps).

We based the resource broker service on the GridBus resource broker,[8] which we modified to adhere to the WSRF standard. (Hereafter, we refer to our GridBus resource broker service simply as the resource broker.) It takes as input an instance of the ADS, which specifies a requested job or multi-job for execution. The resource broker then determines the list of available hardware resources from all administrative domains. From that list, it determines the best matched resources according to user requests and data mining application requirements and then submits the job to the selected sites. In particular, the submission process includes data and data mining application staging and setting up the environmental variables. Afterward, the resource broker monitors jobs and performs data stage-out and clean-up tasks when a job completes.

DataMiningGrid information services are designed to support the discovery, characterization, and monitoring of various resources and services. These services create a registry via the underlying MDS4 to keep records of grid-enabled software resources (that is, ADS instances, which describe data mining applications) and to provide information on other resources (for example, available clusters, storage and CPU capacity, operating systems, and so on). Users can search the registry for software resources according to ADS attributes, such as application names, vendors, versions, functional descriptions, and so on. The resource broker also requires the registry to plan, allocate, and perform job execution.

DataMiningGrid data services involve extensions to OGSA-DAI[6] to provide data-mining-specific functionality for data sets configured as OGSA-DAI data resources, such as XML and relational databases. These are accessed by a set of clients developed with the OGSA-DAI API[6] and can be used to integrate distributed databases, perform data transformations, calculate data summaries, and format data on the fly to support different file formats. File resources aren't usually manipulated using OGSA-DAI but via other DataMiningGrid client components.

DataMiningGrid Client Components

We designed the DataMiningGrid system with three main user groups in mind: administrators who manage the hardware and middleware components; developers who build new grid-based applications; and end users who run the applications. For the latter groups, we envisioned two application client types: a *workflow editor* for more complex and interactive applications, and hard-wired *Web clients*, which provide much simpler interactions. Web clients are intended to be easy to use and are designed specifically for domain-oriented end users with limited interest or knowledge of the underlying data mining and grid technology. The workflow editor is based on Triana[9] and is designed to facilitate fine-grained user control of data mining applications; it supports flexible workflow com-

Related Work in Grid-Based Data Mining

Recently, various systems and approaches to grid-based data mining have appeared in the literature. We briefly review those that are relevant to the DataMiningGrid system here.

GridMiner[1] is designed to support data mining and online-analytical processing in distributed computing environments. It provides a sophisticated distributed data warehousing functionality, implements some common data mining algorithms, including parallel versions, and also supports various text mining tasks. One of the major differences between GridMiner and DataMiningGrid is that the latter complies with the recent trend toward WSRF.

Knowledge Grid (K-Grid)[2] is a service-oriented system providing grid-based data mining tools and services. It can perform wide-ranging data mining and related tasks such as data management and knowledge representation. K-Grid wasn't originally designed to support OGSA and WSRF, but developments are geared toward adding this support. Also, as of the time of this writing, K-Grid isn't available as open source. Whereas the development of the DataMiningGrid was driven by the requirements of a diverse set of applications, a more conceptual view of the knowledge-discovery process has driven K-Grid's design.

Anteater[3] is a service-oriented architecture for data mining that relies on Web services to achieve extensibility and interoperability, and is freely available. However, unlike the other systems, it doesn't support grid standards such as WSRF or OGSA. Moreover, Anteater requires data mining applications to be converted into a *filter-stream* structure. Although this feature greatly increases scalability, the overhead involved is likely to limit the number of applications ported to this platform.

The DataMiningGrid differs in comparison to these other systems because of

- *User friendliness.* It supports different user interfaces, according to the technological capabilities of its users, and we've kept the installation process as simple as possible.
- *Extensibility.* The Application Description Schema allows users to quickly grid-enable existing data mining applications. This is further supported by the workflow editor, which dynamically configures its user interface according to the instantiated schema's parameters.
- *Parameter sweeps.* The Application Description Schema allows the definition of multi-jobs that iterate over application parameters, input files, or directories.
- *Diverse application support.* We've implemented and tested a wide variety of applications from different domains. No modification to the data mining application source code was necessary.
- *Standardization.* The WSRF standard is relatively new and not yet supported by the GridMiner and Anteater systems.
- *Open source.* The GridMiner and K-Grid systems aren't freely available.

Overall, the DataMiningGrid provides a collection of features that have been tested with a diverse set of applications. This makes it a unique and competitive contender for both grid-enabling existing applications and developing new ones.

References

1. P. Brezany, I. Janciak, and A.M. Tjoa, "GridMiner: A Fundamental Infrastructure for Building Intelligent Grid Systems," *Proc. 2005 IEEE/WIC/ACM Int'l Conf. Web Intelligence* (WI 05), IEEE Press, 2005, pp. 150–156.
2. A. Congiusta, D. Talia, and P. Trunfio, "Distributed Data Mining Services Leveraging WSRF," *Future Generation Computer Systems*, vol. 23, no. 1, 2007, pp. 34–41.
3. D. Guedes, W. Meira, and R. Ferreira, "Anteater: A Service-Oriented Architecture for High-Performance Data Mining," *IEEE Internet Computing*, vol. 10, no. 4, 2006, pp. 36–43.

position, parameter settings, input and output data flows, and so on. DataMiningGrid provides the workflow components, which include those for remotely browsing and viewing files, as well as transferring them to and from grid storage servers. Type-checking is applied to connections between workflow components and to any parameters manually entered by users.

Using the DataMiningGrid System

The DataMiningGrid's architecture is best understood by viewing it in action, from the perspective of different types of users. Here, we outline all the necessary steps required to run a simple DataMiningGrid workflow application.

Let's start with an example in which it's necessary to grid-enable an existing data mining application. To do this, the application developer uses a Web-based client called an *application enabler.* Using this client, the application developer uniformly specifies the application's properties and requirements and selects the actual software (executable and libraries) on the local machine. The application enabler then uploads the selected software to a grid storage server, creates a partially filled new instance of the ADS, and registers it with the DataMiningGrid information services. Following this, anyone who has the necessary permissions can instantly discover, configure, and run the data mining application in the grid environment.

Before using the DataMiningGrid workflow components, an end user must have installed Triana,[9] the DataMiningGrid components, and a valid certificate. The end user's part of the DataMiningGrid components consists of Java libraries, which contain the client software used to manipulate data resources, select data mining applications, and orchestrate all the services needed to execute applications in the grid.

Now the end user employs the workflow editor to compose a simple workflow.

The first step involves selecting the data in one of three ways: first, via an OGSA-DAI client that accesses the data services; second, by selecting local files or directories to upload to a default GridFTP server; or third, by specifying files or directories already present on a GridFTP server, such as a preceding run's results.

In the next step, the end user employs a workflow component called an *application explorer* to browse the application registry for the previously grid-enabled data mining applications. When the user selects a data mining application, the application explorer automatically queries the information services and receives the ADS instance of that particular application.

Now the user interface in the workflow editor, which we call an *application control*, is dynamically configured according to the ADS instance's specifications. As described earlier, this interface provides slots for the end user to instantiate fields in the ADS — that is, for specifying mutable options, such as algorithm parameters, data inputs, and so on. By specifying an iteration over an algorithm option, for example, the interface defines a range of numeric variables or a list of strings as parameters for a multi-job. Once the end user specifies all execution options, the ADS instance is passed to the resource broker for execution. The end user now has little involvement in the process until the jobs have completed, except to optionally monitor their execution status.

The resource broker uses the information services to determine the available computational resources. It matches the ADS instance's requirements to descriptions of available resources and selects suitable resources for execution accordingly. The resource broker uses RFT to transfer the executable (for example, a Java .jar file or a C binary and its related libraries) and the uploaded data to each selected grid site, which has a GT4 WS-GRAM installation. WS-GRAM can then use its enhanced Condor adaptor to submit jobs to its local Condor cluster, where they're ultimately executed. The resource broker can execute a multi-job via several different WS-GRAM installations and Condor clusters, which may span different administrative domains. Once the jobs complete, the resource broker will retrieve the job's results from the WS-GRAM installation and transfer them with RFT to a predefined storage server.

Finally, the end user browses and downloads the result files using suitable workflow components.

It's worth mentioning that the resource broker can "ship" data mining applications to targeted machines on the grid. The resource broker determines whether applications are shipped around the grid manually or automatically. This feature greatly adds to the system's flexibility because no pre-installation of applications is required, which supports an important use case scenario: shipping algorithms to data as opposed to shipping data to algorithms. This is extremely useful in many application scenarios we've investigated — for example, if data can't be transferred because of its inherently distributed nature, its large volume, or for privacy issues.

Evaluation

We set up a comprehensive test bed spanning three European countries: the UK, Germany, and Slovenia. Each site provided at least one server with a GT4 installation (two in Germany) and heterogeneous Condor compute clusters, comprising up to 90, 5, and 40 available nodes, respectively, typically with CPU speeds of 1.4 to 3 GHz, RAM of 521 Mbytes to 2 Gbytes, and using both Windows and Linux platforms. In Ulster, we used a 64-node SGI Altix machine with CPU speed of 900 MHz and 128 Gbytes shared RAM. One machine in Germany centrally hosted the resource broker and the information services. We validated the system by performing comprehensive data mining studies based on our main use case scenarios.

One of our applications was topology discovery for gene regulatory networks, which was CPU-intensive and based on a genetic algorithm implemented in Java. It was capable of using every available compute node in the test bed, and thus was able to achieve considerable performance gains.

Analysis of protein folding simulations, on the other hand, was a data-intensive application (data were in the range of 10 Gbytes to 2 Tbytes), and involved distributed data warehouses of molecular simulations.[10] We processed data in situ by shipping algorithms implemented in different languages to the data resources. Removing the need to download large data volumes was important for this application, as was the need to quickly grid-enable new analysis algorithms.

In our medical application scenario, we integrated and queried distributed databases as if they were a single resource by using the data services. Then, we did cross-validation and classification studies using grid-enabled algorithms from the Weka data mining suite. This resulted in a significant increase in productivity, as medical specialists could now easily perform data analyses.

Another scenario — customer relationships and quality management in the automotive industry — involved the distributed analysis of millions of documents by shipping preprocessing algorithms to the document repositories. Proprietary programs for document classification and ontology learning were grid-enabled and combined in complex workflow applications.

Our ecosystem modeling scenario involved a compute-intensive application based on an equation-discovery machine learning program written in C and Python, which we restricted to run under Linux. Due to these requirements, only 18 of the available nodes in the test bed could execute this application, yet the ecologists who weren't familiar with grid technology reported that their applications returned results on average more than six times faster. They also appreciated the ability to construct complex workflows to easily execute complex and error-prone applications, which they previously did manually.

Now, let's review a more detailed explanation of a text mining application concerned with classification. Typical text mining tasks applied to large and fast-evolving text corpora include text classification facilitation, for example, document redirection to suitable subject matter experts, emerging subject discovery, and keyword extraction. In this text classification case study, we measured cross-validation runtimes on newswire data available from the Deutsche Presse-Agentur (DPA).

We performed our experiments on 1,000 documents in the DPA collection from October 2004 (the source XML file was 39 Mbytes) using several grid-enabled Java applications. Each tenfold cross-validation run began with a document preprocessing stage — for example, to remove stop words, digits, and punctuation, and to convert the document collection to a binary format. In each cross-validation experiment, we performed parameter sweeps on the following: the importance weight for corpus terms, the term significance threshold, and the text category used for classification. Jobs with different settings for these three parameters ran in parallel, and each job split the data tenfold and performed training and classification 10 times.

To gather performance information, we ran the experiments identically on a single machine, sequentially, and compared the results with parallel runs on the three Condor pools in the test bed. This led to the following results: speed-up depended on the input data location within each site because the algorithm is data intensive and required constant file reading. For a central file server with NFS and Gigabit Ethernet, we measured a linear speed-up for up to five machines. For all the machines in the test bed, we could maintain linear speed-up if we mirrored input data on local disks within each site by specifying the proper Condor setting. In the case of local data, parallel executions had a quasi-linear speed-up during runtime due to grid overheads concerned with cluster configurations, file transfer, and scheduling.

It seems obvious that emerging large-scale data mining applications will rely increasingly on distributed computing environments. To tackle these and other issues, we developed the DataMiningGrid system, and its main features include high performance, scalability, flexibility, ease of use, conceptual simplicity, compliance with emerging grid and data mining standards, and the use of mainstream grid and open technology. Our DataMiningGrid software is freely available under the Apache Open Source License V2.0 via SourceForge.net, including supporting documentation. Future developments with the DataMiningGrid are concerned with complex applications that can't be easily split into largely independent computational tasks. Such problems require sophisticated grid middleware technology that facilities efficient interprocess communication within grids.

Acknowledgments

European Commission FP6 grant DataMiningGrid, contract number 004475, supported this work.

References

1. I. Foster, C. Kesselman, and S. Tuecke, “The Anatomy of the Grid: Enabling Scalable Virtual Organizations,” *Int’l J. High Performance Computing Applications*, vol. 15, no. 3, 2001, pp. 200–222.
2. A. Kumar, M.M. Kantardzic, and S. Madden, “Guest Editors’ Introduction: Distributed Data Mining — Framework and Implementations,” *IEEE Internet Computing*, vol. 10, no. 4, 2006, pp. 15–17.
3. V. Stankovski et al., “Grid-Enabling Data Mining Applications with DataMiningGrid: An Architectural Perspective,” *Future Generation Computing Systems*, vol. 24, no. 4, 2008, pp. 259–279.
4. P. Plaszczak and J.R. Wellner, *Grid Computing: The Savvy Manager’s Guide*, Morgan Kaufmann, 2006.
5. B. Sotomayor and L. Childers, *Globus Toolkit 4: Programming Java Services*, Morgan Kaufmann, 2006.
6. M. Antonioletti et al., “The Design and Implementation of Grid Database Services in OGSA-DAI,” *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2–4, 2005, pp. 357–376.
7. G. Von Laszewski et al., “A Java Commodity Grid Kit,” *Concurrency and Computation: Practice and Experience*, vol. 13, nos. 8-9, 2001, pp. 645–662.
8. S. Venugopal, R. Buyya, and L. Winton, “A Grid Service Broker for Scheduling e-Science Applications on Global Data Grids,” *Concurrency and Computation: Practice and Experience*, vol.18, no. 6, 2006, pp. 685–699.
9. G. Churches et al., “Programming Scientific and Distributed Workflow with Triana Services,” *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, 2005, pp. 1021–1037.
10. C. Silva et al., “P-Found: The Protein Folding and Unfolding Simulation Repository,” *Proc. 2006 IEEE Symp. Computational Intelligence in Bioinformatics and Computational Biology* (CIBCB 06), IEEE Press, 2006, pp. 101–108.

Vlado Stankovski is a researcher in the Department of Civil Informatics in the Faculty of Civil and Geodetic Engineering at the University of Ljubljana. His research interests include semantic grid technology and data mining. Contact him at vlado.stankovski@fgg.uni-lj.si.

Martin Swain is a research fellow at the University of Ulster. His research interests are biophysics, systems biology, data management, and grid computing. Swain has a PhD in intelligent computing systems and bioinformatics from the University of Aberdeen. Contact him at mt.swain@ulster.ac.uk.

Valentin Kravtsov is a PhD student at the Technion—Israel Institute of Technology. His PhD research topics include grid technologies and distributed and parallel computing. Kravtsov has an MSc in software engineering from the Technion. Contact him at svali_ds@cs.technion.ac.il.

Thomas Niessen recently left Fraunhofer Institute for Intelligent Analysis and Information Systems and is now a software developer at Scopevisio AG. His research interests include distributed computing, especially grid technology, data mining, and databases. Niessen has an MSc in computer science from the University of Applied Sciences Bonn-Rhein-Sieg in Bonn. Contact him at thomas.niessen@gmx.net.

Dennis Wegener works as research fellow at the Fraunhofer Institute for Intelligent Analysis and Information Systems, Department of Knowledge Discovery. His research interests include data mining and grid computing. Contact him at dennis.wegener@iais.fraunhofer.de.

Matthias Röhm is a researcher with the text mining team at DaimlerChrysler. His research interests are in distributed text mining. Contact him at uni-ulm.m.roehm@daimlerchrysler.com.

Jernej Trnkoczy is a researcher in the Department of Civil Informatics in the Faculty of Civil and Geodetic Engineering at the University of Ljubljana. His research interests are in federated digital libraries. Contact him at jernej.trnkoczy@fgg.uni-lj.si.

Michael May is head of the Department of Knowledge Discovery at the Fraunhofer Institute for Intelligent Analysis and Information Systems. His research interests are in data mining and knowledge discovery. Contact him at michael.may@iais.fraunhofer.de.

Jürgen Franke is head of the text mining team at DaimlerChrysler. His research interests include text mining and character recognition. Contact him at juergen.franke@daimlerchrysler.com.

Assaf Schuster is head of the Distributed Systems Laboratory at the Technion. His research interests include distributed and parallel computing. Contact him at assaf@cs.technion.ac.il.

Werner Dubitzky occupies the chair of bioinformatics at the University of Ulster. His research interests include bioinformatics, systems biology, data and text mining, artificial intelligence, and grid technology. Contact him at w.dubitzky@ulster.ac.uk.