Deployment and Exploitation of Grid-enabled Data Management for Engineers

Jasmin Wason, Zhuoan Jiao, Wenbin Song, Andrew Price & Simon Cox

School of Engineering Sciences, University of Southampton, UK
{j.l.wason, z.jiao, w.song, a.r.price, sjc}@soton.ac.uk

Abstract

In this paper we describe the Geodise Database Toolbox, which utilises Web services, XML, databases and Grid technologies to help manage data created by engineering applications running locally or on the Grid. It has been integrated into the Matlab and Jython scripting environments for ease of use, and into other applications via its Java API. The toolbox supports centralised vs. personal data repositories, the former accessed via secure Web services from platform independent client applications. Metadata can be easily defined on files, data structures, collections of related data, and workflows. A distinctive feature is the support for user-defined application specific metadata that can be queried to locate required data efficiently. We describe the toolbox, how it has been deployed and exploited, and indicate that our approach has proved sufficiently generic to be useful in a range of application areas.

1. Introduction

Engineering design search and optimisation (EDSO) aims to achieve improved designs by exploiting engineering modelling and analysis. Variables in a design are systematically modified to increase, or reduce a quality measure called the objective function, whilst ensuring the variables satisfy certain constraints. It often involves computationally and data intensive processes, producing large amounts of data at different locations in a variety of formats. The emergence of Grid computing and Web service technologies provides new opportunities for the engineering community to access an extended range of compute resources and manage the sizable data created by distributed applications more effectively. To help engineers utilise the available compute and data resources on the Grid, the Geodise project [1] have developed computational and database toolboxes to bring the technologies into an environment familiar to engineers. The Geodise Computational Toolbox [2] provides a suite of functions which enable and simplify access to Grid compute resources, and the database toolbox offers similar support for data management and sharing.

In previous papers [3, 4] we described how the Geodise Database Toolbox, wrapped as a set of Matlab [5] functions that can be incorporated into a user's Matlab scripts, has been used in Computational Fluid Dynamics (CFD) and Computational Electromagnetic (CEM) engineering problems. Matlab is a powerful, flexible, and easy to use scripting environment popular with engineers. It also provides a powerful execution engine for our workflows, and is available on a wide variety of platforms and operating systems.

The underlying toolbox is exposed as a Java API so it can be integrated with different scripting environments, such as Jython [6], and other applications that can interact with Java code. For example, the GENIE project [7] supports environmental scientists modelling long term climate change and has used the API to manage data from their framework using a web portal. The Geodise Workflow Construction Environment (WCE) [8] also uses the API to a) archive, query, and retrieve the workflows for reuse and sharing; b) store data for monitoring workflow executions. Recently we have released the toolbox's client code to the Integrative Biology project [9] so that researchers can experiment with using our repository remotely to manage their application data. We have also extended the toolbox so that it can be used more conveniently for monitoring application processes while they are running, so that engineers can intervene to halt or change long running optimisations if necessary.

In this paper we will give an overview of the architecture in section 2 and discuss deployment in section 3. A description of how the toolbox has been used in various applications will be given in section 4 and in section 5 we will demonstrate its use in application monitoring using an example drawn from engineering design practice. Section 6 summarises the conclusions and future work.

2. Architecture Overview

Traditionally, data created from engineering applications is stored in files on file systems with little information to describe them. When the data volume is large, this makes them difficult to search, share and reuse. The limitation of this approach becomes more obvious when a group of people working in different institutions, i.e. in a Virtual Organisation (VO), wish to collaborate to solve a common problem, making use of Grid technology. This can be overcome by attaching additional descriptive information (metadata) to the data, so that it can be located by querying its characteristics rather than having to know its location.

2.1 Metadata

To encourage the use of metadata within the engineer's environment it must be straightforward to specify and sufficiently flexible to contain any terms and nested data structures required to describe the relevant application data. The Storage Resource Broker (SRB) [10] provides a uniform interface for connecting to heterogeneous resources over a network and, with the Metadata Catalog (MCAT), provides dataset access based on characteristics rather than names or physical locations. However, MCAT has limited support for application specific metadata, particularly the complex, nested data structures and data types which are often essential in assisting engineering users to locate data specific to their problems.

To achieve the required flexibility and ease of use we have developed the Geodise Database Toolbox which allows engineers working in the Matlab environment to define metadata conveniently as Matlab structures. Standard metadata (e.g. archive date, file size) are generated automatically by the toolbox so that users only need to concentrate on defining custom metadata specific to their applications, and granting access permissions to other users in the VO if they want to share the data. Metadata is stored in an XML enabled relational database (Oracle 9i [11]): standard metadata is stored in tables for efficient access, whilst user defined metadata is stored as native XML for flexibility, and this can be transparently converted to and from user defined Matlab structures using our XML Toolbox for Matlab [12].

The database is queried using a simple syntax to locate files/variables or groups of data based on their characteristics. Users specify queries as a combination of named metadata variables and comparison operators, with options for exact or wildcard matches and case insensitivity. A user query is converted to a combination of SQL and XPath, and restricted so that it only returns results the user is entitled to access. The returned array of structures may contain all the metadata for each result or a specified subset, for example the user may only want the unique data identifiers, which can then be used to retrieve files from the archive, regardless of where they are stored. Users can incorporate the query function into their Matlab scripts directly or interact with a query GUI supporting hyperlinks for downloading and browsing related data.

A datagroup is used to logically group together related data, such as that used in a process monitoring task, and metadata can be added at the group level so that the entire collection is described. A datagroup may also contain sub-datagroups, and data can belong to multiple datagroups. This gives users the ability to describe and exploit relationships or hierarchies.

Jython [6] is a pure Java implementation of the Python scripting language which, like Matlab, allows seamless integration with any Java code. We were therefore able to implement the database toolbox in Jython by providing a thin set of functions on top of our existing Java client API. We also provide functionality analogous to the XML Toolbox for Matlab, converting between metadata contained in Jython dictionaries or objects and XML. Data types in the XML Toolbox are described by type, dimensions and array index where applicable, and using the same XML syntax for Jython means metadata can be passed between the environments via the database. For example, an engineer may prefer to install Jython for free at home and query the database to monitor the progress of their current job. Alternatively, Jython may be used to archive files and associated metadata from Grid compute resources that do not have access to a Matlab license, then later in the lab the data can be queried, retrieved and plotted for analysis using Matlab.

For applications running in other environments, metadata can be defined using the appropriate data structures and then converted into XML format before calling the toolbox's Java API. For example, in the Geodise WCE metadata is defined initially as Java objects, and then converted into XML before archiving.

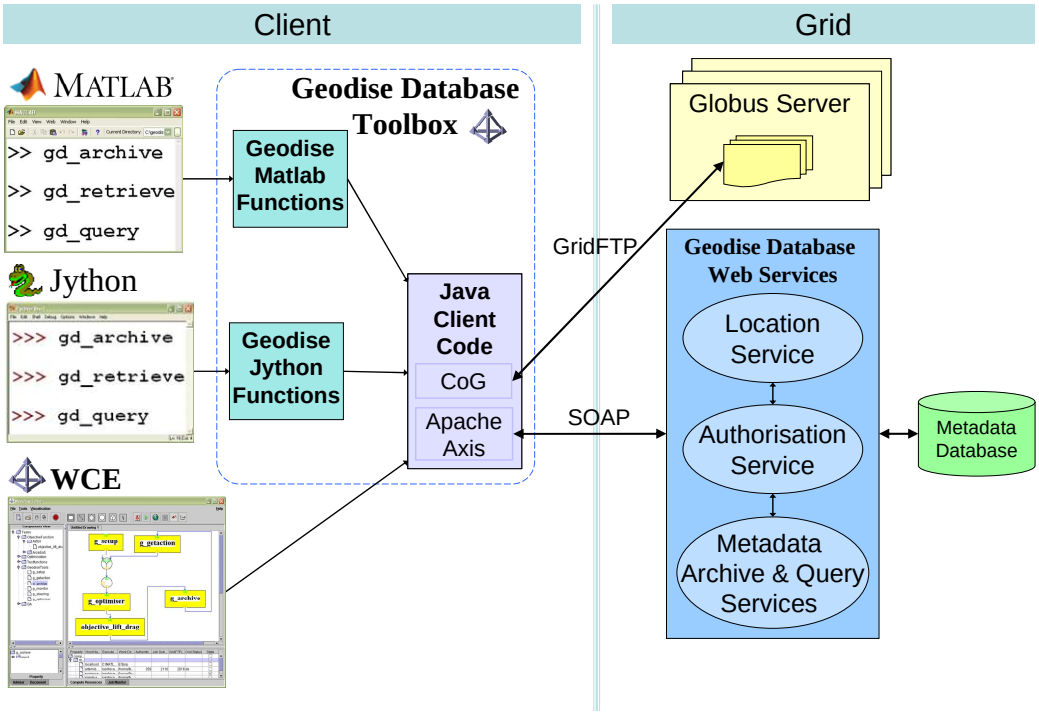

Figure 1 - A GUI or high level set of scripting functions on top of a client side Java API provides an interface to secure data management Web services and file transfer. User scripts may run locally as shown or on remote compute resources with the user's delegated authorisation credentials.

2.2 Web and Grid technologies

Both files and variables (primitive values and data structures) can be archived in the repository with associated system generated and user defined metadata. Files are archived to a user specified remote file store, using GridFTP [13] and assigned Universally Unique Identifiers (UUIDs) [14]. This information is recorded in the database allowing files to be physically located and retrieved by their UUIDs later. Variables are archived as XML in a way similar to the user defined metadata.

All database access is supplied through secure Web services for storage and query of metadata, file location and authorisation, as shown in Figure 1. The Web services are invoked, using Apache Axis [15], from platform independent client code running locally or on Grid resources. To support distributed users and prevent unauthorised access we use a secure method of Web service communication. It allows X.509 certificate based authentication and authorisation using the same delegated credentials that provide a point of single sign-on for computational jobs on the Grid [16]. The user's unique identifier (certificate subject) is extracted from the certificate that signs the SOAP [17] request message and used to authorise database access, record data ownership and allow users to grant access permissions to others.

Projects like Spitfire [18] and OGSA-DAI [19] develop generic middleware to provide database access from Grid applications. Spitfire is a project within the European DataGrid (EDG) project [20]. It provides a thin layer on top of a relational database management system, and uses Web service technology to provide SOAP-based RPC to some user-definable database operations. It is designed to give quick and easy access to data where the access patterns are simple, rather than to be used as a full-fledged data management system. Problems are anticipated for long-running queries, queries returning a very large result set, and other operations that need a large amount of system memory.

We also encountered a problem when the volume of query results was very high. In this situation returning all the results from the Web service as one very large SOAP message can cause Axis to run out of memory when it attempts deserialization on the client. To prevent this we split large results into chunks, which are no more than 2 MB in size by default. The client code pulls the results back in stages from the service, which will indicate when they have all been returned. This is transparent to the user, as the client code passes on the full results when the process is complete. Users can

optionally set the chunk size in a configuration file according to their computer's capabilities.

The OGSA-DAI project produces open source database access and integration middleware for use in the Grid environment. It provides an extension to the Open Grid Services Architecture (OGSA) specifications [21] to allow data resources, such as databases, to be incorporated within an OGSA framework. Its latest release (version 4.0) provides a client toolbox that minimises the specialist knowledge required to interact with OGSA-DAI services and shields the developer from the changes that have been seen in previous releases. OGSA-DAI's provision for asynchronous delivery of query results to various destinations, such as a URL or a file, is a useful feature that we shall investigate using in the future.

3. Deployment

We now discuss some alternative deployment strategies for the client and server tools. The computational and XML toolboxes are prerequisites of the database client toolbox, which also requires classpath additions and a configuration directory to be copied to the user's home directory. The client configuration file can then be altered to specify the location of the database Web services and which Globus [22] enabled server to store files on. We have found that for organisations sharing a Matlab installation the easiest arrangement is to configure these details and the classpath centrally. Then users only need to run a single command to copy the configuration information to their home directory and start using the toolbox, providing they have passed on their e-Science certificate subject to the relevant administrator(s) so they can be authorised to access the database and file store.

Deployment of the server components is more involved, requiring Oracle 9i and an application server (e.g. WebSphere [23]) to be installed together or on separate machines. The database schema is created by running a script, and then user certificate subjects and IDs can be added. The Web services are deployed on the application server, and will access a configuration file to determine the location of the database. A public version of the Web services exists at Southampton so that the database can be accessed by any authorised user, regardless of location.

The Integrative Biology [9] project aims to create an infrastructure allowing biomedical researchers to use Grid computing, databases and visualisation tools to develop complex models of how heart disease and cancer develop. By installing the Geodise database client toolbox they were able to experiment with it, by connecting to a remote repository via the Web services hosted at Southampton, before deciding whether to deploy the database and server components of the toolbox locally.

The database toolbox has been fully deployed in the Computational Engineering Design Centre (CEDC) [24] at Southampton University, where a number of projects use optimisation and design search engineering methods coupled with industrial applications. Engineers are currently using the database to store data associated with structural optimisation and F1 aerodynamic design optimisation problems. A separate installation of the database and services was required at the CEDC due to a policy requiring all systems to remain behind a firewall, to protect commercially sensitive data.

The GENIE project has augmented and deployed the Geodise database code and database schema to meet their specific needs. For example, the standard Geodise database must be flexible enough to handle any user defined metadata, so it is stored as unstructured XML. However, the metadata produced by the GENIE framework is more tightly constrained and can be described by an XML Schema, which improves query efficiency when mapped to the Oracle 9i database.

In response to user feedback we have implemented a standalone version of the database toolbox that can be used when no network connection is available, for example when at home or travelling. A user can switch between the local and central archive by setting a configuration parameter. In standalone mode the code calls local class versions of the Web services to access a personal database, and uses the local file system rather than a remote Globus file store. At present the personal edition of Oracle 9i database must be installed, but we are looking at more lightweight XML enabled database alternatives. Data stored in the local archive can later be uploaded to the central repository using simple synchronisation based on the unique handles.

We have also implemented an administrative data cleanup facility which allows information in the database (and corresponding files) to be deleted by performing a query to find metadata meeting certain restrictions. This has not yet been included in the deployed toolbox, as we intend to further develop the tool to minimise the risk of accidental deletion of important data. Users have requested the ability to remove

unwanted data, (e.g. from test runs) and we would like to provide a tool that allows them to mark data for deletion, or expiry after a certain period. This would hide the data from ordinary query or retrieval commands, but not immediately delete it, providing a chance for recovery. An administrator can inform users when the repository is due to be cleaned up, which will involve deleting, or taking offline to a long-term data store, any flagged data.

4. Applications

We shall now describe some of the different ways in which the database toolbox software and architecture have been used. The industrial application of engineering design optimisation using Computational Fluid Dynamics (CFD) is a subject of ongoing research in the Geodise project. Three-dimensional engine nacelle optimisation is a complex example of this, which aims to reduce the ground noise generated by the engine fan when an aircraft takes off. Metadata has been used extensively in these studies to define the model and manage data. Metadata was attached to large binary files with proprietary formats, enabling the use of queries to search for the required files without having to open them. Datagroups were used to organise the data in a hierarchy, providing a logical abstraction for the model files, which are conventionally scattered in the file system. The introduction of more efficient data management approaches and tools also encourages the reuse of previously explored models by engineers working to deliver better designs within restricted timescales.

The GEM project [25] is developing software to improve the design of optical components for next generation integrated photonic devices. Optimisations involve a large number of initial designs and parameters to vary, giving rise to many solutions and large amounts of data. All the solutions may yield valuable information for future simulations and need to be preserved. Post-processing relevant data became easier when the database toolbox was used. The conventional file system approach can make it very difficult to find data relating to particular simulation runs, where there often is no way to search for an individual data range, or user description.

The GENIE project has recently used the database toolbox to manage data during tuning studies of an Earth system model. In addition to being a convenient resource for monitoring progress, sharing data, and post-processing, the database also provides a level of fault tolerance during the studies. When an optimisation is run it will typically execute about 1000 model runs and record all valid evaluations in the database. Occasional Grid job failures inevitably occur during such an optimisation so the database provides a convenient resource for analysing or re-running the study. Although failures are handled by the optimisation algorithm it is sometimes desirable to re-evaluate the failed points. In this case the GENIE code will retrieve all the values recorded in the database and resume the study from the point of failure. This mode of working is particularly efficient for optimisations, which could take days to re-compute if started from scratch.

The architecture described permits data archiving, querying and retrieval from any location with web access, Java support, the required APIs and a valid Globus proxy certificate. Consequently, in addition to migrating the toolbox to Jython, we could also integrate it into our GUI based Workflow Construction Environment (WCE) [8] with ease. The WCE helps users construct workflows visually. Constituent components can be dragged from a hierarchy of tasks on to the main workflow view and connected to each other by specifying the relationships between input and output parameters. Each workflow is saved in two formats, one described in XML for future reuse in the graphical editor, and the other (more importantly) as a Matlab script. This script can run on compute resources with Matlab support, and be reused and edited outside the WCE without engineers learning new workflow formats. Both the XML and the Matlab versions of the workflow can be stored locally or sent to the Geodise archive with additional metadata, calling the database toolbox's Java API. The WCE verifies that the resources are accessible to the user and configured properly before executing the workflow script which may archive user configured data as it is running. The archived workflows and result data can be later retrieved from the database for reuse, modification, and sharing among collaborators.

5. Process Monitoring

In deploying our toolbox to users, we have found the Grid accessibility of the database has made the implementation of remote monitoring and steering capabilities possible. Optimisation processes involving high-fidelity analysis codes are time consuming and therefore it is desirable to monitor the progress of the search as it is running and make adjustment if necessary. One

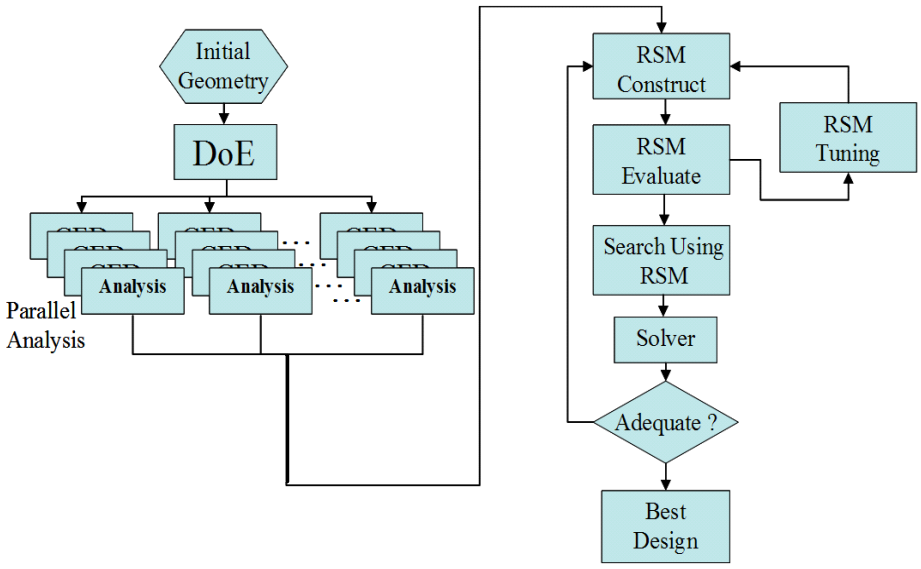

Figure 2 - A typical optimisation workflow in which a Response Surface Model (RSM) is built from objective functions evaluated on points returned from a Design of Experiment (DoE).

of the most common tasks in EDSO applications is to monitor the progress of the optimisation process by plotting the objective function against the number of iterations. This gives a useful indication of whether additional analysis operations are yielding improvement to the design. More complicated tasks include monitoring the evolution of a complex Response Surface Model (RSM) as more analyses are performed and allowing the user to drill into interesting designs and further examine related geometries or analysis results.

The decoupling of process and storage of data means customised monitoring information can be retrieved and post-processed independently and asynchronously from the running script. This is not possible in most integrated packages which implement monitoring on the local machine controlling the optimisation using their internal data storage. We have extended our database toolbox to further support monitoring tasks by attaching job related metadata to specialized datagroups. Each group can contain data related to a particular job and has a name, start time and automatic index number, in addition to the usual custom metadata. This allows a user to more easily query for information on their most recent job, or the latest job that matches certain other criteria. The monitoring group provides a place for users to archive application specific data in a way that allows corresponding monitoring scripts to retrieve the required information and display the current progress of a particular job.

A typical optimisation workflow as shown in Figure 2 is used on a two dimensional optimisation problem. The figure shows how an RSM can be built from objective functions evaluated at points returned by a design of experiment (DoE). A user can monitor the parallel evaluation of the design points to find out how many points have been evaluated and whether they have all been successful. Monitoring can also take place while the response surface is built, searched, and updated with the best point found on the RSM. For problems with more than two design variables, other visualisation techniques can be used after retrieving the search history from the database. Figure 3 (a) shows the original points returned by DoE and an additional single point generated by the update process. Figure 3 (b) and (c) show the initial and updated response surfaces models.

The above example shows how the database toolbox and repository have provided a convenient way to deposit and monitor variables as the search progresses. Process steering can be achieved similarly. A running script retrieves the latest variables deposited in the database by a separate steering program, and makes decisions based on the values to control the design search. The RealityGrid [26] project has developed an API that provides application steering capabilities. File-based or socket-based IO is used to steer the application from a steering client, rather than the database approach described here.

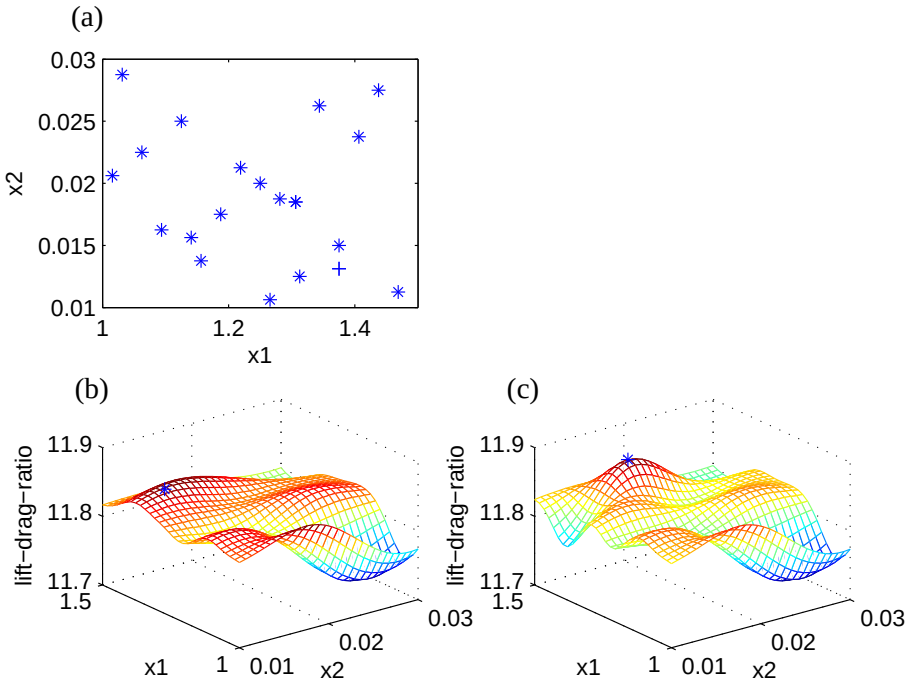

Figure 3 - Monitoring of optimisation process using Geodise database toolboxes.

6. Conclusions and Future Work

The database toolbox described here has played an active role in managing data created by EDSO applications running on the Grid. Its support for metadata, especially user-defined application specific metadata, and its ease of use has made it an appealing tool to be integrated into the daily working environment of engineers, and scientists from other disciplines.

Furthermore, as users wish to share metadata and data, the use of a consistent semantic framework will become more important. Geodise has developed a set of domain specific ontologies to describe engineering applications, and engineers can use these, with the help of tools like the ontology annotator [27] to describe functions, data and processes. We shall further enrich the toolbox by developing tools to help engineers create a semantic framework for their metadata and also to publish and share important Matlab data structures that can be used as templates to generate or validate other data structure instances. This will utilize previous work [28] on generating and managing XML Schemas to describe data structures.

An important issue for future consideration is the provision of a redundant failover system for the metadata services and database. The current central database may be extended to a replicated architecture by using, for example Oracle Stream, a high-speed tool that allows synchronization of replicated databases; and file replication management can be achieved by wrapping Globus replica management libraries [29] with possible extensions as has been shown in the Replica Manager [30] implemented by the EDG project.

7. Acknowledgement

This work is supported by the Geodise e-Science pilot project (UK EPSRC GR/R67705/01), and GENIE [7] is funded by the UK Natural Environment Research Council (NER/T/S/2002/00217).

8. References

1. Geodise (Grid Enabled Optimisation and Design Search for Engineering) Project: http://www.geodise.org
2. M.H. Eres, G.E. Pound, Z. Jiao, J.L. Wason, F. Xu, A.J. Keane, and S.J. Cox. *Implementation and utilisation of a Grid-enabled Problem Solving Environment in Matlab*. Future Generation Computer Systems (in press).
3. W. Song, A.J. Keane and S.J. Cox. *CFD-Based Shape Optimisation with Grid-Enabled Design Search Toolkits*. Proceedings of UK e-Science All Hands Meeting 2003, 619-626.

4. J.L. Wason, M. Molinari, Z. Jiao and S.J. Cox. *Delivering Data Management for Engineers on the Grid*. Euro-Par 2003 Parallel Processing, Lecture Notes in Computer Science, 2003, 413-416.
5. Matlab 6.5. http://www.mathworks.com
6. J. Hugunin. *Python and Java: The Best of Both Worlds*, Proceedings of the 6th International Python Conference, San Jose, California , 1997
7. GENIE (Grid ENabled Integrated Earth system model) project, http://www.genie.ac.uk
8. F. Xu and S.J. Cox. *Workflow Tool for Engineers in a Grid-Enabled Matlab Environment*. Proc. UK e-Science All Hands Meeting 2003, 212-215.
9. Integrative Biology Project, http://www.integrativebiology.ox.ac.uk
10. A. Rajasekar, M. Wan and R. Moore. *MySRB and SRB - Components of a Data Grid*, 11th International Symposium on High Performance Distributed Computing (HPDC-11) Edinburgh, Scotland, 2002.
11. Oracle 9i Database, http://otn.oracle.com/products/oracle9i
12. M. Molinari. *XML Toolbox for Matlab*, GEM/Geodise, 2004, http://www.soton.ac.uk/~gridem/xml_toolbox
13. B. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel and S. Tuecke. *Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing*, Proceedings of the IEEE Mass Storage Conference, 2001.
14. P. J. Leach and R. Salz. *UUIDs and GUIDs*, Network Working Group Internet Draft, 1998, http://hegel.ittc.ukans.edu/topics/internet/internet-drafts/draft-l/draft-leach-uuids-guids-01.txt
15. Axis, The Apache Software Foundation, http://ws.apache.org/axis/
16. I. Foster, C. Kesselman, G. Tsudik and S. Tuecke. *A Security Architecture for Computational Grids*. Proc. 5th ACM Conference on Computer and Communications Security Conference, 1998, 83-92.
17. M. Gudgin, M. Hadley, N. Mendelsohn, JJ Moreau and H.F. Nielsen. *Simple Object Access Protocol (SOAP)*, W3C Recommendation 24 June 2003 http://www.w3.org/TR/soap12
18. Spitfire, European DataGrid Project, http://edg-wp2.web.cern.ch/edg-wp2/spitfire
19. OGSA-DAI, Open Grid Services Architecture-Data Access and Integration http://www.ogsadai.org.uk
20. European DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid
21. I. Foster, C. Kesselman, J. Nick and S. Tuecke. *The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration*. June 2002. http://www.gridforum.org/ogsa-wg
22. The Globus Toolkit, http://www.globus.org
23. IBM WebSphere Application Server, http://www-3.ibm.com/software/webservers/appserv
24. Computational Engineering and Design Centre, Southampton University, http://www.soton.ac.uk/~cedc
25. The GEM (Grid Enabled electroMagnetic optimisation) project, http://www.soton.ac.uk/~gridem
26. J. M. Brooke, P. V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning, and A. R. Porter. *Computational Steering in RealityGrid*. Proceedings of the UK e-Science All Hands Meeting 2003.
27. L. Chen, N.R. Shadbolt, C. Goble, F. Tao, S.J. Cox and C. Puleston. *Managing Semantic Metadata for the Semantic Grid*. Submitted to ISWC2004, Hiroshima, Japan, 2004.
28. Z. Jiao, J.L. Wason, M. Molinari, S. Johnston, and S.J. Cox. *Integrating Data Management into Engineering Applications*. Proceedings of UK e-Science All Hands Meeting 2003, 687-694.
29. Globus replica management, http://www.globus.org/datagrid/replica-management.html
30. P. Kunszt, E. Laure, H. Stockinger, and K. Stockinger. *Advanced Replica Management with Reptor*. In 5th International Conference on Parallel Processing and Applied Mathematics, Poland, September, 2003. https://edms.cern.ch/file/404070/1/CP27.pdf