OGSA-DAI Status and Benchmarks

Mario Antonioletti[1], Malcolm Atkinson[2], Rob Baxter[1], Andrew Borley[3], **Neil P. Chue Hong**[1], Patrick Dantressangle[3], Alastair C. Hume[1], Mike Jackson[1], Amy Krause[1], Simon Laws[3], Mark Parsons[1], Norman W. Paton[4], Jennifer M. Schopf[2,5], Tom Sugden[1], Paul Watson[6] and David Vyvyan[3]

1. EPCC, University of Edinburgh, JCMB, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK.
2. National e-Science Centre, Universities of Edinburgh & Glasgow, Edinburgh EH8 9AA, UK.
3. IBM United Kingdom Ltd, Hursley Park, Winchester S021 2JN, UK.
4. Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK.
5. Mathematics and Computer Science Div., Argonne National Laboratory, 9700 S. Cass Ave., Argonne, IL, USA.
6. School of Computing Science, University of Newcastle upon Tyne, Newcastle upon Tyne NE1 7RU, UK.

Abstract

This paper presents a status report on some of the highlights that have taken place within the OGSA-DAI project since the last AHM. A description of Release 6.0 functionality and details of the forthcoming release, due in September 2005, is given. Future directions for this project are discussed. This paper also describes initial results of work being done to systematically benchmark recent OGSA-DAI releases. The OGSA-DAI software distribution, and more information about the project, is available from the project website at www.ogsadai.org.uk.

1 Introduction and Overview

OGSA-DAI [1] is a widely used piece of middleware infrastructure enabling client applications to perform a set of activities on a remote data resource, e.g. a relational database or a file. These activities are aggregated into a single *request document* to minimise the number of client-service interactions required to obtain the desired result. The use of request documents also avoids unnecessary data movement by placing the computation next to the data.

We provide a set of core activities that implement the basic functionality needed to interact with a data resource, and it is easy for users to add new activities that operate within the OGSA-DAI framework. There are three types of activities: *Statement activities* that wrap the user's query into a format understood by the underlying database to perform basic functions such as update, query, joins, etc.; *Transformation activities* that change the formatting of the data into output more suitable for a given client (for example, changing raw XML results into output for project web site, or compressing the data); *Delivery activities* that allow 3rd party delivery techniques such as GridFTP or SMTP.

The first OGSA-DAI distribution was released in January 2003. In the last two years there have been six major and three minor releases all of which have been based on the OGSI infrastructure [2].

After release 5.0, December 2004, a move was made towards supporting two additional platforms, WS-I and WSRF. We define a WS-I-based platform to be one that only uses standards explicitly included in the WS-I Basic Profile [3], for example, a vanilla Apache Tomcat and Axis infrastructure or the WS-I+ [4] -based OMII 2.0 distribution. Our second new supported platform is the WSRF-based platform, as implemented by the Globus Toolkit 4.0.

In May 2005 we released three related OGSA-DAI distributions: release 6.0 of the OGSI-based distribution, which will be the last release for this platform, and release 1.0 of both OGSA-DAI WS-I and OGSA-DAI WSRF.

This paper discusses the effort made to migrate to three supported platform infrastructures, the current and planned features for these versions, some performance data, and data about current usage and projects.

2 Migration

When the move to three supported platforms began, we restructured the code significantly to have a common core with interfaces to allow for the different versions. The infrastructure had to be refactored to provide for this core component in addition to WSI, WSRF and OGSI-specific components, where each component consists not only of code but also XML Schema, user documentation, configuration files and build tools.

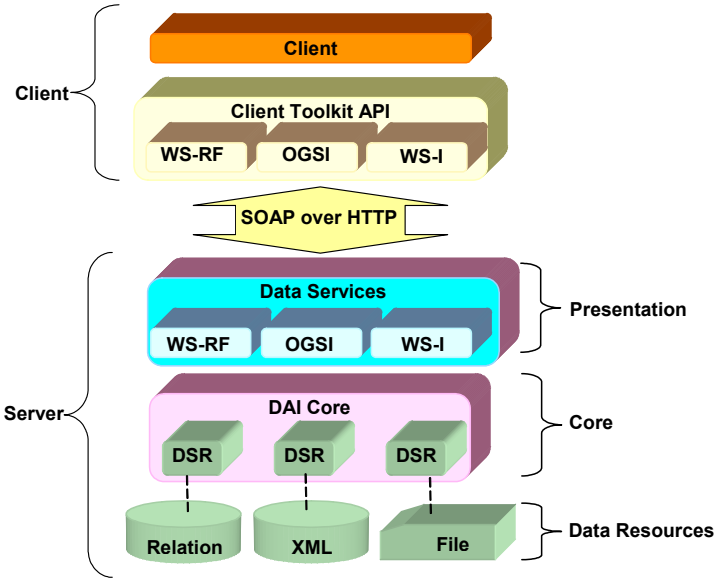

Figure 1: Schematic representation of OGSA-DAI.

We took advantage of this refactoring to implement a change in the service model. This is a first step towards the new OGSA-DAI architecture, the design of which is being led by Malcolm Atkinson and is outlined in [5].

2.1 Refactoring the Server

In order to make the development and maintenance tractable, we restructured the OGSA-DAI server code into a three-tiered implementation, shown in Figure 1..

The Presentation Layer consists of a set of *Data Services* that perform two functions. First, a Data Service accepts a version-specific message from the client, and strips the payload from the message to create a version-neutral *perform document* that consists of the set of activities to be performed. Second, the Data Service decides which *Data Service Resource (DSR)* in the Core Layer is the proper one for the activities and submits the document to that DSR.

The Core Layer, sometimes called the Processing Layer, consists of a set of DSR's that act as a front-end to a *Data Resource*. Each DSR implements the core DAI functionality which includes overseeing the coordination of the activities for a specific Data Resource. A Data Resource can be anything from a database to a simple file. A DSR may also expose additional capabilities such as data transport-related operations and can also cache data for retrieval by third-parties (if the data service resource is configured to support asynchronous data delivery). When completed, the DSR sends the response back to the client, in the form of a *response document,* through the Presentation layer. The DSRs are the same for all three platforms (OGSI, WSRF, and WS-I) but specific to a Data Resource type.

For now, Data Services and DSRs are deployed dynamically and have lifetimes bound to that of the container. Additional DSRs can be associated (or dis-associated) with a Data Service, but these operations require the container to be re-started before these changes are registered by the service. The service model and the functionality it exposes will continue to evolve as elements of the new architecture [5] are incorporated into future releases of OGSA-DAI.

2.2 Refactoring the Client

On the client-side, the multiplicity of platforms is hidden from the user. The client toolkit (CTk) API has been refactored to abstract away the differences between the different messaging infrastructures. It takes the information about the activities from a user, constructs the performance document, and then wraps it with the headings for the platform in use. This implementation-specific SOAP message is then passed on to the Presentation Layer. If a client needs to know what type of OGSA-DAI service is being used, the *getVersion* operation will return the OGSA-DAI distribution type (WSRF, WS-I or OGSI) and its version number.

The changes made to the CTk greatly increase the usability when developing and using the OGSA-DAI middleware. By having a single client interface for all platform versions of OGSA-DAI we were able to use the same system tests across the three platforms, thus mitigating some of the effects of having to support three platforms. The client toolkit has also proven to be of value by hiding changes in specifications from client tools.

2.3 Effort

In addition to extensive code restructuring for both the client and server, the adaptation to three platforms also included changing the XML schema used additional documentation, changes in configuration files, and additional build tools. The overnight automated test system also had to be extended to run tests over three builds rather than one. In all, we estimate that this refactoring took approximately two developer months to complete, compared to two weeks to port just the code..

However, in addition to these upfront costs, there are many ongoing overheads. Moving from one to three distributions results in an increase in the time to prepare and test releases as well as in satisfying our ongoing user support commitments. It also means that we must address the expectation that, for example, OGSA-DAI WSI services be fully inter-operable with OGSA-DAI WSRF services. In addition, commitments to release OMII-compliant versions of OGSA-DAI and to bundle OGSA-DAI with the Globus Toolkit yield additional constraints upon the team's time. This inevitably has constrained the effort available to develop new functionality or address areas of concern such as performance.

For this reason, together with the finite developer effort available, it has become necessary to discontinue further development of the OGSI-compliant OGSA-DAI distribution. It is hoped that the community will be able to converge on a single infrastructure and thus OGSA-DAI need only support a single service layer. However, it does not appear as though this will happen in the near future so we are forced to drop support for OGSI in order to achieve a balance between supporting emerging specifications and extending and improving OGSA-DAI.

3 Other Current and Planned Features

The main change in the OGSI distribution for 6.0 was the refactoring of the code base. It also had a number of bug fixes. Other changes and additions in functionality that affect all three distributions are discussed in this section.

3.1 Control Flow

In the current release we have implemented control flow constraints to be expressed in a perform document. This dictates the processing order of activities that should facilitate the description of more complex interactions that require a definite temporal ordering to make sense, for example a new table needs to be created before it can be populated.

The control flow capability is enabled through two new elements: a *sequence* element signifies that any activities or activity pipelines contained within it will be processed sequentially, one after the other; and a *flow* element allows any activity or activity pipeline contained in to be processed concurrently using different processing threads. These two elements can be nested within each another. These elements provide the client with some control over the order in which activities are processed – the activity model previously worked in a much more haphazard manner. Thus it is possible to specify that one activity does not start until its predecessor has completed or that several activities should be performed in parallel.

3.2 Differences Between Versions

The functionality included in the WS-I and WSRF releases is, more or less, equivalent but not yet on a par with that available in the OGSI based release. Two notable exceptions are:

- **Lack of concurrency support**: only one client can interact with a data service at time. Previously, in the OGSI-based model, the factory pattern could be used to create a new service, a Grid Data Service (GDS), on demand allowing concurrent access to the data resource.
- **Lack of security support**: Grid credentials are not used by the services to map the role data from the DN to a data base user and password. Also, transport and message level security have not been tested as thoroughly as the previous OGSI version.

Both of these are expected to be addressed in the next OGSA-DAI release, due in September 2005.

3.3 Future Plans

In addition to unifying the functionality of the current versions, several other additional features are planned for the next release. These include:

- **Implementation of the GGF DAIS WG [6]**: part of the motivation of OGSA-DAI has always been to not only provide a piece of middleware to provide access to data but also to try and standardise the way that data is accessed through web service interfaces. Hence, the OGSA-DAI group has been very active in the production of a set of standards through GGF. A candidate standard recommendation is expected shortly after GGF14 (June 2005). We plan to implement this standard as part of the standardisation process.
- **Tighter integration with OGSA-DQP**: OGSA-DAI has mainly addressed the issue of data access, with much less effort spent on data integration. The OGSA-Distributed Query Proecess (DQP) Project [7] has been adding data integration functionality using OGSA-DAI services. OGSA-DQP works closely with the myGrid project [8]. In the next release of OGSA-DAI we plan to have a closer integration of the two products and embed some of the DQP functionality into the OGSA-DAI framework.
- **Benchmarking of OGSA-DAI code**: In order to better understand the performance limitations of OGSA-DAI, and to be able to recommend deployment guidelines, we are benchmarking the OGSA-DAI distributions. Initial results are discussed in later in this paper.

4 Benchmarking OGSA-DAI

Although some effort has previously been expended by the OGSA-DAI team doing performance analysis and optimisation of the OGSA-DAI code [14], a more systematic approach is now being undertaken to produce a benchmark suite that will run automatically. The recorded results will be made public to encourage selection of priorities and focused improvements and to inform the community on best practice regarding getting good performance out of OGSA-DAI services. Here we present some preliminary results concentrating on delivery of data from an OGSA-DAI data service.

Initial studies have shown that the performance of OGSA-DAI can vary markedly depending on which of the various activities and delivery mechanisms are used. The default behaviour of OGSA-DAI is to return result data within a SOAP response document. Two useful alternatives to this default are to return the data using FTP or to make the data available via an HTTP URL using a servlet. Both of these delivery mechanisms bypass the overhead of SOAP but may require slightly more work from the client developer.

The first OGSA-DAI benchmark measures the time taken to send an SQL query to the server and iterate through each of the rows returned, summing the values in one of the columns. The data set used was the OGSA-DAI test dataset '*littleblackbook*' which can be generated by code included in the OGSA-DAI distributions. This dataset contains one table with four columns as shown in Table 1. In all cases, the server was already running. For the JDBC tests, the time to create the connection was not included in the benchmark. The software stack used included OGSA-DAI WSRF 1.0, Globus Toolkit WS Core 4.0.0, Apache Tomcat 4.1.29 and MySQL 4.1.9. Table 2 shows the specifications of the server and client machines.

Column name	*Column type*
Id	Int
Name	varchar(64)
Address	varchar(128)
Phone	varchar(20)

Table 1: *Littleblackbook* test table

The results of the benchmark are show in Figure 2. Results obtained by connecting directly to the MySQL database on the server using JDBC are also included for comparison, although it should be noted that this approach returns data without conversion to the WebRowSet format so cannot be directly compared.

	Server	*Client*
OS	Windows XP	Windows 2000
Processor	Intel Pentium M	Intel Pentium 4
Speed	1.60GHz	2.40GHz
RAM	1GB	0.5GB
Java VM	1.4.2_04	1.4.2_04
Network	100Mbit shared network	

Table 2: Client and server specification

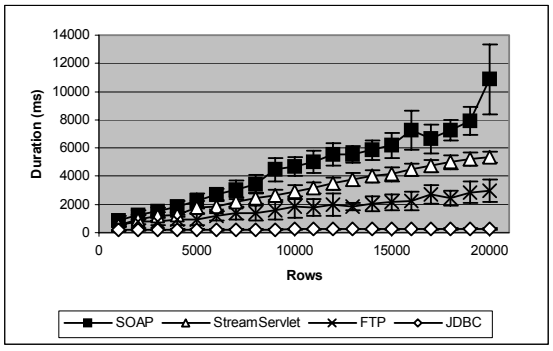

Figure 2: OGSA-DAI and JDBC performance – retrieving rows from relational database. Error bars show standard deviation

OGSA-DAI's *deliverToStream* activity allows users to specify that data should be retrieved from a servlet thus allowing a simple HTTP request to be made without incurring the overhead of SOAP. SOAP is still used for the request and response as normal but the SOAP response does not contain the result data.

A performance gain can be achieved by bypassing the SOAP overhead when delivering the data, as shown when we compare the results for the default SOAP delivery mechanism and the use of the *deliverToStream* activity. Use of the *deliverToStream* activity requires the client to make an additional request to retrieve the data. To remove this additional request while retaining the performance improvement of bypassing SOAP we plan to provide SOAP with attachments [15] and MTOM [16] delivery options in later versions of OGSA-DAI.

Figure 2 also includes the results obtained using OGSA-DAI's *deliverToURL* activity to send query results to an FTP server. It is unfair to compare the FTP results directly against the other approaches because the FTP results do not include the client iterating through the rows of the result set. Despite this it is useful to include the results to emphasise that OGSA-DAI can be used to extract results from a database and return these as files using FTP or GridFTP. In this case the files are in the XML WebRowSet format but it would be easy to include an activity at the server that converts these to another format before delivering the data.

The results shown in Figure 2 for the SOAP and Stream Servlet implementations use the OGSA-DAI Java client toolkit to iterate through the result returned by the query. The client toolkit returns objects that implement the Java *ResultSet* interface. By default the client toolkit returns a forward-only *ResultSet* and it is the results of using this *ResultSet* implementation that are shown in Figure 2. OGSA-DAI provides a forwards-backwards *ResultSet* implementation. The OGSA-DAI benchmarks compare the performance of both implementations.

Figure 3 shows the performance results when using the default SOAP delivery mechanism with the two *ResultSet* implementations. The forwards-backwards implementation is significantly faster than the forwards-only implementation. These results are echoed for the *StreamServlet* delivery mechanism as shown in Figure 4.

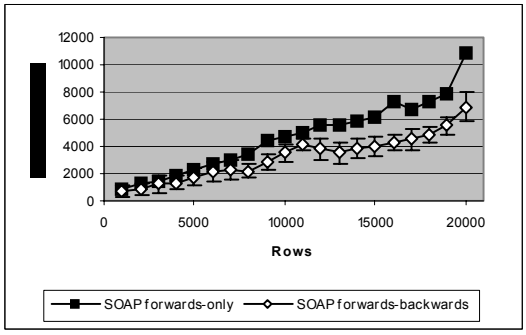

Figure 3: Performance of SOAP delivery using different *ResultSet* implementations.

It is surprising that the forwards-backwards *ResultSet* implementation performs so much better than the forwards-only implementation. The forwards-backwards implementation parses the XML result data and creates a DOM object representing all the data. The forwards-only implementation attempts to only parse the XML result data one row at a time when the client requests the row. The forwards only implementation has a considerably smaller memory footprint than the forwards-backwards implementation and both require only a single pass through the XML data. Given this we would expect the large performance difference between the two is surprising and will be investigated in more detail by the OGSA-DAI team.

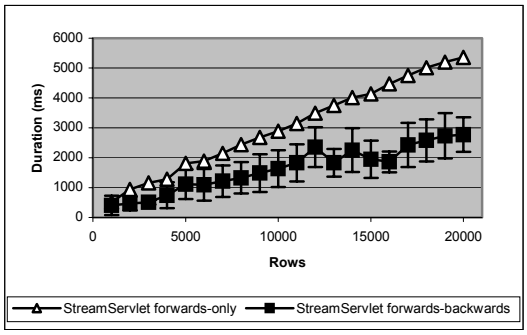

Figure 4: Performance of Stream Servlet delivery using different *ResultSet* implementations

Table 3 includes a selection from the above graphs in numerical form to allow for easier comparison.

Delivery	*ResultSet*	*Rows*	*Duration (ms)*	*Std Dev*
SOAP	FO	10,000	3,671	644
SOAP	FO	20,000	10,881	2,478
SOAP	FB	10,000	3,529	637
SOAP	FB	20,000	6,907	1,046
Servlet	FO	10,000	2,885	477
Servlet	FO	20,000	5,047	401
Servlet	FB	10,000	1,635	618
Servlet	FB	20,000	2,773	576
JDBC	-	10,000	236	30
JDBC	-	20,000	275	46

Table 3: OGSA-DAI performance benchmark results.

The OGSA-DAI benchmark work has started to lead towards a much better understanding of the performance of OGSA-DAI.

5 Projects and Collaborations

Current download statistics for the different distributions are shown in Table 5. One must be careful in interpreting these figures but they do demonstrate that there is still an interest/demand for the OGSI based releases. Release 5.0 is still required for use with the 2.0 release of the OGSA-DQP package. Releases 1.0 to 3.1 of OGSA-DAI and release 1.0 of the OGSA-DQP package are no longer available for download from the OGSA-DAI web site as these are no longer officially supported.

OGSA-DAI Version	*Downloads*
Total OGSI (R1→R6)	4392
OGSI R6.0 only	201
WSRF (1.0 only, excluding GT4.0 downloads)	228
WS-I (1.0 only)	90

Table 5: Download stats (as of 29/06/05)

Downloads of the WSRF release are currently outstripping the WS-I release. This may be in part by the momentum gathered by the association with the GT4.0 release. The WS-I version will become part of the OMII distribution in July 2005, which might contribute a similar effect. Within Globus, OGSA-DAI is in the process of migrating from a Technical Preview to a Core Component of the Globus Toolkit.

It is also instructive to examine what versions of OGSA-DAI projects are using or will adopt and what their upgrade path is likely to be if they are already using an OGSI version of OGSA-DAI. A small number of projects were contacted, some of these having come to existence since those reported to be using OGSA-DAI at the previous AHM meeting, [9].

In a lot of cases a specific version of OGSA-DAI has been used and there is no intent to upgrade either because the project has come to an end or the software is perceived to be stable and there is no perceived need to upgrade lest this break something. Some of those projects that are active or that are starting:

- *Cancer Biomedical Information Grid* (cabig.nci.nih.gov) [10] have data services that leverage of OGSA-DAI R5.0. Their next phase may move to use GT4.0 and the WSRF version of OGSA-DAI but this remains to be decided.
- *Lead* (www.lead.ou.edu) have been using OGSA-DAI release 3.1 but plan to migrate to the WS-I version soon.
- *myGrid information repository project* (www.mygrid.org.uk/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=47) used one of the technical preview version of the OGSA-DAI WS-I distributions. Future development plans aim to stick with the WS-I version of OGSA-DAI.
- *Data Mining Grid* (www.datamininggrid.org) is using GT4.0 and the WSRF version of OGSA-DAI. Interestingly they also plan to try

use the Triana [11] workflow editor and integrate this with GT4.0 and OGSA-DAI.

- *Grid Miner* (www.gridminer.org) are using OGSA-DAI release 5.0 and do not have current plans to migrate, although when they do it will probably be to the WSRF version when it is deemed to be mature.
- *SIMDAT* (www.simdat.org) are using a version of the WS-I implementation that operates with GRIA [12].
- *GOLD* (www.goldproject.ac.uk) are also intending to use the WS-I version of OGSA-DAI. So these projects are using the activity framework to tailor code to their own requirements and deploy it through OGSA-DAI.

This list of projects seems to reflect the divided nature of the community with both WS-I and WSRF based projects being well represented. This view was reflected at the third OGSA-DAI users' group meeting [13], held at NeSC with no dominance of preference of one infrastructure over the other.

A number of major projects, including the EU FP6 projects SIMDAT, inteliGrid and DataMiningGrid, are currently consulting the team as part of their evaluation or as part of their design concerning their use of OGSA-DAI. The OGSA-DAI team are in an e-Science sisters project, DIALOGUE, which is developing a strategy for combining multiple data integration systems. The team continues to contribute to standardisation and three GGF DAIS recommendation documents are planned by mid 2005.

6 Conclusions

Two years after its first release, OGSA-DAI continues to mature and expand in its functionality. Both the user and contributor base are growing, with major projects in the USA, Europe and Asia adopting OGSA-DAI for their Grid applications. Although there has been a small step backwards to introduce support for WS-I and WSRF based platforms, this has led to improvements that ease future development.

Work has been carried out to understand the performance bottlenecks of OGSA-DAI and already some interesting results have been gathered. Further work on profiling the software, in conjunction with improvements suggested by the new architecture will hopefully lead to future versions of OGSA-DAI that can address the more complex issues of data integration.

***Acknowledgements*:**
This work is supported by the UK e-Science Grid Core Programme, the Open Middleware Infrastructure Institute, and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-ENG-38. We also gratefully acknowledge the input of our past and present partners and contributors to the OGSA-DAI project including: EPCC, IBM UK, IBM Corp., NeSC, University of Manchester, University of Newcastle and Oracle UK.

7 References

[1] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. P. Chue Hong, B. Collins, N. Hardman., A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. W. Paton, D. Pearson, T. Sugden, P. Watson and M. Westhead. *The design and implementation of Grid database services in OGSA-DAI.* Concurrency and Computation: Practice and Experience 17(2): 357-376.

[2] S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderpilt, *Open Grid Services Infrastructure, Version 1.0*, June 27, 2003, GFD.15.

[3] K. Ballinger, D. Ehnebuske, C. Ferris, M. Gudgin, C.K. Liu, M. Nottingham, and P. Yendluri (ed.) Basic Profile Version 1.1, Web Services Interoperability Organization Final Material, 2004-08-24. http://www.ws-i.org/Profiles/BasicProfile-1.1.html.

[4] M. Atkinson, D. DeRoure, A. Dunlop, G. Fox, P. Henderson, T. Hey, N. Paton, S. Newhouse, S. Parastatidis, A. Trefethen, P. Watson, J. Webber.*Web Service Grids: An Evolutionary*

Approach.
http://www.omii.ac.uk/web_service_grids.htm.

[5] M. Atkinson, K. Karasavvas, M. Antonioletti, R. Baxter, A. Borley, N. Chue Hong, A. Hume, M. Jackson, A. Krause, S. Laws, N. Paton, J. Schopf, T. Sugden, K. Tourlas, P.Watson. *A New Architecture for OGSA-DAI*. AHM 2005.

[6] See http://forge.gridforum.org/projects/dais-wg for more details.

[7] See http://www.ogsadai.org.uk/dqp,

[8] See http://www.mygrid.org.uk.

[9] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, N. P. Chue Hong, B. Collins, J. Davies, D. Fitzgerald, N. Hardman, A. C. Hume, M. Jackson, A. Krause, S. Laws, N. W. Paton, T. Sugden, P. Watson, M. Westhead and D. Vyvyan. *OGSA-DAI Status Report and Future Direction*. Proceedings of the UK e-Science All Hands Meeting 2004, September 2004.

[10] W. Sanchez, B. Gilman, M. Kher, S. Lagou, P. Covitz. *caGRID White Paper*. July 23, 2004. See http://cabig.nci.nih.gov/guidelines_documentation/caGRIDWhitepaper.pdf.

[11] See http://www.trianacode.org.

[12] *GRID Resources for Industrial Applications*, see http://www.gria.org.

[13] Notes and presentations from the meeting available at http://www.ogsadai.org.uk/docs/UG3/

[14] M. Jackson, M. Antonioletti, N.P. Chue Hong, A.C. Hume, A. Krause, T. Sugden, and M. Westhead. *Performance Analysis of the OGSA-DAI Software*. Proceedings of the UK e-Science All Hands Meeting 2004, September 2004.

[15] J.J. Barton, S. Thatte, H. F. Nielsen. *SOAP Messages with Attachments*. W3C Note 11 December 2000. http://www.w3.org/TR/SOAP-attachments

[16] M. Gudgin, N. Mendelsohn, M. Nottingham, H. Ruellan. *SOAP Message Transmission Optimization Mechanism*. W3C Recommendation 25 January 2005. http://www.w3.org/TR/soap12-mtom/