

# Eldas (Enterprise Level Data Access Services)

Stephen Rutherford, Aileen Campbell, Alan Gray, Brian Hills, Davy Virdee,

Rob Baxter, Denise Ecklund

Edikt, National e-Science Centre, University of Edinburgh, Edinburgh, UK

<http://www.edikt.org/eldas/>

## Abstract

The Grid Database Service Specification defines a standard for accessing data stored in systems such as relational databases, XML repositories and files. Edikt have developed Eldas, a commercial quality implementation based on these standards, J2EE and industry adopted Web Services. We discuss: the design issues and considerations in Eldas' development; the resultant architecture; architectural benefits in light of Web Services Resource Framework; the current functionality; future work.

## 1. Introduction

Data is a critical resource in any research or commercial endeavour. Reliable tools to access and integrate legacy, raw and derived data are in high demand. Addressing these needs, the Edikt team has developed Eldas, a commercial quality implementation recognising the core requirements and functionality addressed by the Grid Database Service Specifications (GDSS) [1] from the Data Access and Integration Services Working Group (DAIS-WG) [2] within the Global Grid Forum (GGF) [3]. Eldas services can presently be invoked using both a Grid Services and a “pure” Web Services [4] interface. To date Grid Services have been enabled by using an implementation of Open Grid Service Infrastructure (OGSI) [5] from Globus [6].

Recently OGSI has been superseded by the Web Services Resource Framework (WSRF) [7]. Accordingly future versions of Eldas will support WSRF along with a more lightweight “stateful Web Services” interface.

Typically the phrase “commercial quality software” means a software package is robust, well-documented, easy to use and designed to support evolution and the addition of new service features. This is not sufficient, however, to ensure broad adoption of the Eldas software. Edikt have identified several other issues, common to various scientific disciplines, that could constrain Eldas adoption.

Careful consideration of requirements, the aforementioned issues, and industry standards has resulted in an architecture based on Java 2 Platform, Enterprise Edition (J2EE) [8], Enterprise Java Beans (EJB) [9] Session Beans and established design patterns. It is this

architecture which allows Eldas to support different interfaces such as WSRF.

### 1.1 Edikt

The e-Science community consists of scientists using and developing software tools and applications to support collaborative research involving very large datasets in disciplines as diverse as astronomy, biological sciences, physics, and geosciences. Edikt (e-Science Data Information & Knowledge Transformation) [10] is a project based at the National e-Science Centre [11], University of Edinburgh, UK. Edikt's remit is to provide software solutions for the e-Science community using both existing and emerging technologies. This naturally leads to taking both proven industrial solutions and cutting-edge research ideas and applying rigorous software engineering standards to develop “commercial quality” products. Of particular importance in this area is the ever widening scope of service oriented computing and the increasing adoption of these standards in both academia and industry. Both Web and Grid Services are key areas in service oriented computing and have been areas of particular focus for Edikt.

Eldas [12] (Enterprise Level Data Access Services) is a core Edikt technology – its aim is to be a service oriented data access solution for the e-Science and industrial communities, and also a component in the bespoke applications Edikt builds for scientific and commercial enterprises.

### 1.2 Structure of Document

#### *Introduction*

This section sets the scene for Eldas and relevant technologies.

### ***Motivation***

The Motivation section discusses the reasons for developing Eldas including some specifics on the standards used.

### ***Design Issues and Considerations***

We investigate some of the requirements and issues influencing the Eldas design, and take a brief look at the software engineering disciplines used within Edikt.

### ***Architectural Overview***

The section includes details on the Eldas architecture, and discusses how this approach addresses the issues highlighted in the previous section.

### ***Functional Overview***

The current functionality is described, both in terms of the Eldas Server itself and its associated tooling. We also identify some of the functionality to be included in future releases.

### ***Application Areas***

Eldas application areas are identified and referenced.

### ***Conclusions, Acknowledgements & References***

The paper finishes with a summary of the conclusions, acknowledgements and a list of references.

## **2. Motivation**

Many of the motivations for Eldas specifically are coincident with the motivations for Edikt generally i.e. the provision of commercial quality software solutions for the requirements of the e-Science community. Eldas meets these needs by providing a service oriented approach to solving the problems of data access and integration - for academia initially, the commercial world thereafter.

### **2.1 Data Access in Service Oriented Architectures**

A very important feature of the Eldas architecture is the ease with which different interfaces can be supported – it is possible to invoke Eldas Data Services via a “pure” Web Services interface as well as through using a Grid Services interface. To date the Eldas Grid Services interface has been based upon Globus Toolkit 3 (GT3) [13] which is an implementation of the OGSi from Globus. Details on the Eldas OGSi Grid Service interface can be found in [14].

### **2.2 From OGSi to WSRF**

An important recent development has been the succession of OGSi by WSRF with the intention of assimilating some of the advances made by OGSi into the wider Web Services community. As a result GT3 will be replaced by a WSRF implementation, GT4, and Eldas will transition to this new platform once it is released, later in 2004. This migration is facilitated by Eldas’ multi-tiered architecture where the functionality is separated into layers: a presentation layer which contains the interfaces, a business layer which contains the functionality, and a data layer for accessing data resources. This Eldas Architecture is more fully explored in the Architectural Overview.

### **2.3 Web and Grid Services**

WSRF refactors some of the ideas explored in OGSi within the context of established Web services standards. This is an approach Eldas encourages given that we have always viewed the key elements of Grid Services as a logical extension to Web Services. Consequently future versions of Eldas will support WSRF. In the meantime, however, it proves useful to make Eldas data access services available via both Web Services and Grid Services. The decoupling of Eldas interfaces from its business logic facilitates supporting service invocations using a variety of interfaces. This allows Edikt to explore the possibility of developing the interfaces such that, in addition to supporting a WSRF interface, future versions of Eldas will allow service invocation using a more lightweight alternative in the form of “stateful Web Services”. This idea is best summarised by Edikt’s simplified view of Grid Services - that is “Web Services plus identity, state and lifetime”.

### **2.4 Enterprise Level Software**

Core to the ethics of the team that produced Eldas is the belief in producing commercial quality software using industrial software engineering practices. This involves employing design techniques that facilitate robustness, scalability and performance; leveraging proven, world-class platforms such as Sun's Java 2 Enterprise Edition (J2EE) framework and the JBoss Application Server [15]; and following best practices in software engineering.

## **3. Design Issues and Considerations**

Core to Edikt’s remit is that software solutions must be “commercial quality”

products, and Eldas is no exception. Prior to considering what this means, however, we focus on discussing some major issues which could potentially constrain the adoption and wide use of service oriented, data access middleware.

### **3.1 Issues to Broad Adoption**

#### ***Ease of Use***

Edikt believe that unless software is easy to install, deploy and use, then people will look elsewhere. This is a direct reflection of our own experiences when evaluating software. Hence, if data access middleware is to be commonly and widely adopted it must be straightforward in all aspects of its use. Consequently Edikt have invested a great deal of time and effort in ensuring that Eldas is easy to deploy and use.

#### ***Robustness***

If ease of use is necessary to ensure adoption of a software product, then robustness is absolutely critical in ensuring that the software continues to be used and extends its user base. Hence Edikt have been developing robust software as a result of the software engineering standards highlighted below.

#### ***Extensibility***

It is also critical that Eldas is designed to an extensible architecture. A modular design enables the addition of, and modification to, functionality with minimal disruption to the code base, facilitating development of robust code in a controlled, manageable fashion, and in achievable timescales.

#### ***Data Resource Independence***

Eldas is designed to provide access to heterogeneous, distributed data such as Relational Database Management Systems (RDBMSs), XML databases, ASCII and binary flat files. This also requires a modular architecture such that the code for accessing the data can be extended with the minimum of disruption and effort.

#### ***Machine Independence***

Distributed data means that the data sources are inevitably hosted on a variety of machine architectures. Indeed service oriented and Grid computing make provision for this by providing interfaces which are not specific to any operating system. While, in theory, these service oriented interfaces allow Eldas data access services to be invoked from any machine, it is clearly paramount that instances of the Eldas servers can be deployed on various machine architectures. Consequently this was a

consideration when deciding how to develop Eldas.

#### ***Interface Independence***

While Eldas is supplying data access solutions for a service oriented world, invoking these services naturally depends on which particular interfaces are being used. With this in mind it is unnecessarily restrictive to adhere to one single set of standards, and conversely hugely advantageous to support as many interface standards as possible. Hence why, in addition to Web and OGSi Grid Services, future versions of Eldas will support WSRF Grid Services and a lighter-weight version described as "stateful Web Services".

### **3.2 Software Engineering**

Well defined processes have been, and will continue to be used, throughout the software development cycle of Eldas. Example processes include: use case analysis for requirements capture; UML class, object and sequence diagrams for design; test driven unit development reflecting required specifications and agreed interfaces. Our test infrastructure includes nightly builds, and daily automated testing covers both unit and system tests for regression and to match predefined results. Our designs typically reflect industry standards and widely accepted patterns.

In support of the software development lifecycle, Edikt have a range of infrastructure tools to support configuration control and update repositories. Infrastructure maintenance is aided by highly qualified systems staff at the National e-Science Centre.

## **4. Architectural Overview**

The previous section discussed some of the Design Issues and Considerations pertinent to making Eldas a successful product, notably: Ease of Use; Robustness and Performance; Extensibility and Data Resource Independence; Machine Independence; Interface Independence. The Eldas architecture, while summarised here, is discussed in more detail in last year's meeting [16]. This section consequently focuses more attention on how the architecture addresses the above issues.

### **4.1 Addressing the Design Issues**

#### ***Ease of Use***

Eldas has a number of tools and clients presently available which make it easy to use, discussed in more detail in the Functional Overview. The architecture has aided this

development through having a thin presentation layer. This presentation layer can be easily tailored to support development of the tools and clients built on top of the core server functionality.

### **Robustness and Performance**

Careful development and consideration of a modular design helps ensure robust and performant software. Clearly defined, stable modular interfaces allow improvement and enhancement to existing code without affecting the remainder of the software. Without such a modular design, any changes to the code tend to be hugely disruptive.

### **Extensibility and Data Resource Independence**

Extensibility and Data Resource Independence also greatly benefit from the adoption of a modular design as such an architecture facilitates extension of components in a non-disruptive fashion. This is particularly true of Eldas' Data Access Component where different data sources can be supported simply by adding specialised units which communicate with the generalised core. There are parallels with the aforementioned presentation layer which allows Eldas to support any number of interfaces. In effect we also have a distinct, dedicated data layer, decoupled from the business logic, which means there are no restrictions on the number of types of data resource which Eldas can access.

### **Machine Independence**

The use of Java, J2EE and related technologies ensures Eldas is machine independent. This is reinforced by building Eldas' deployment tools using Apache Ant [17] scripts as this permits the building of automated tools which are not machine specific.

### **Interface Independence**

The ability to invoke services independently of the interface used is a direct result of a multi-tiered architecture which groups the interfaces in a presentation layer distinct from the underlying business logic. Use of a delegate pattern makes it relatively simple to include different interfaces such as Web Services, Grid Services or "stateful Web Services". Similarly migration from OGSi to WSRF is a straightforward task, reinforcing the advantages of Eldas' multi-tiered architecture.

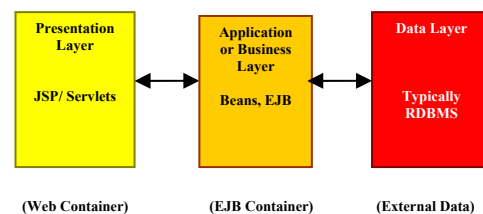
## **4.2 J2EE and EJBs**

J2EE, EJBs and their relevance to service oriented architectures is very briefly

summarised here and explored in more detail in [16].

### **J2EE Framework**

J2EE is designed to support applications that need to be scalable, available, reliable, secure, transactional and distributed, recommending it as an ideal framework on which to build data access services. J2EE has become the *de facto* standard for developing multi-tier enterprise applications, by having standardised, modular components and services. This allows many details of application behaviour to be created easily, without complex programming. A typical server-side J2EE multi-tiered architecture is illustrated in Figure 1. Layering in the J2EE Framework.



**Figure 1. Layering in the J2EE Framework**

### **Enterprise Java Beans**

Enterprise Java Beans are used as server-side components in Eldas as their architecture enables the factory and service features of Grid technology to be implemented easily.

All EJBs have similar behaviour when invoked. Bean lifetime is managed by an Object Factory, which instantiates a specific instance, or EJB Object, which in turn supplies a service to the user.

## **4.3 The Eldas Architecture**

The GDSS extends the concepts of the more general Grid Service Specification [18] to allow services to access data resources.

As detailed in [16], Grid Services can be implemented using a Web Services framework, making use of all the features and functionality of J2EE, i.e. web containers and application servers. There are also many similarities between the way EJBs and Grid Services behave. An implementation of the GDSS interfaces in an EJB framework gives all the advantages of enterprise level applications to service oriented data access.

Eldas implements the GDSS of WSDL port types using EJB technology. An Eldas Data Service Factory (EDSF) has the same behaviour attributes as an EJB Object Factory and an Eldas Data Service (EDS) has the same behaviour attributes as an EJB Object.

The amount of functionality in the presentation layer is thus minimised through this use of EJBs. This architecture follows a delegation pattern and is a major factor in why Eldas can easily support and migrate to different interfaces.

The EDS is implemented as a stateful session EJB. This particular implementation has been used so that only one bean is available for each client session, thus maintaining the conversational state with the client; in other words, they can have state or instance fields that can be initialised or changed by the client with each method invocation. A service invoking a stateful session bean will thus have “identity, state and lifetime”.

Unlike many EJB database applications, we do not use entity beans to access the database, since for each query produced by the client the overhead of invoking an entity bean each time would be very large on large datasets. Instead, we connect the EDS session bean to a data resource using our Data Access Component (DAC).

The DAC has been designed to allow access to various data resources depending on the driver used. The DAC has been implemented as a pluggable component, allowing changes to be made easily to the underlying implementation without affecting the development of the bean classes. Connecting to different data resources is hence simply a case of including additional components.

The architecture is illustrated in Figure 2. The Eldas Architecture.

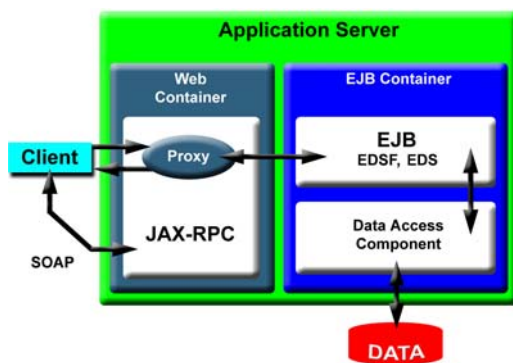


Figure 2. The Eldas Architecture

## 5. Functional Overview

This section details some of the functionality available in Eldas. First we look at what is currently available, and then what functionality is to be included in future releases. At the time of writing, the current release is Eldas 1.0.

### 5.1 Current Eldas Server Functionality

#### Interfaces to Eldas Data Service Invocation

Eldas currently supports the following interfaces for invoking its data services.

- Grid Services - an OSGI based Grid Services interface for running queries against data resources. This interface supports streaming of results as well as results being returned in bulk.
- Web Services - a simple Web Services interface for running a query against an identified data source is available.

#### Supported Data Resources

The current Eldas release supports the MySQL database.

#### Supported Operating Systems

Eldas is supported on Linux, Solaris and Windows. Supported means that Eldas has been exhaustively tested on these systems.

#### Supported Application Servers

Eldas can be deployed on the application server JBoss 3.2.2.

#### Security

The current security model uses GSI Message Level Security [19] and HTTPS.

#### Stored Database Procedures

Eldas allows the invocation of MySQL database stored procedures.

### 5.2 Eldas Clients

Eldas has a range of clients, both command line and graphical, facilitating invocation of data access services.

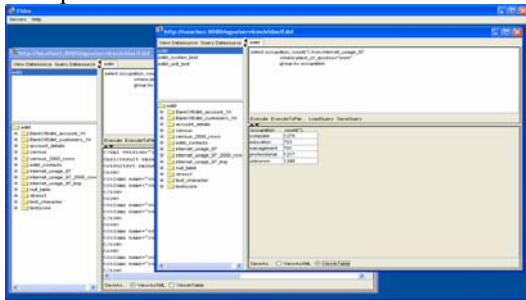
#### Command Line

Each of the available interfaces comes with a simple command line client allowing the creation of services and hence queries on data resources. These command line clients are principally included as examples illustrating the ease with which Eldas clients can be developed and extended. Hence the distribution also includes client source code and documentation.

#### Eldas Query Tool

An example of an alternative to the command line clients is the Eldas Query Tool [20]. Building as it does on the OSGI Grid Services interface, this stands as a very useful tool in its own right as well as an exemplar

piece on the graphical clients which can be built on top of Eldas' interfaces.



**Figure 3. The Eldas Query Tool**

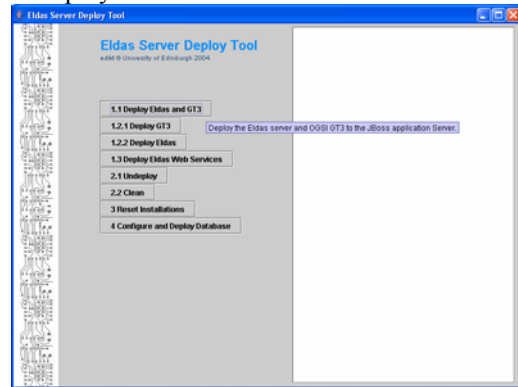
Figure 3. The Eldas Query Tool is a screenshot of the query tool. The different panes reflect connection from a single graphical client to distinct instances of the Eldas Server. On each pane, the left hand frame can be used to inspect the different data sources available, revealing the structure of the data. The right hand panes allow queries to be entered where the results are returned either in a tabulated fashion, as XML, or saved to file in the case of streaming. Results can subsequently be saved to other files in a variety of data formats.

### 5.3 Eldas Tools

We have stressed the importance of ease of use from the beginning of the paper. One factor in a product being easy to use is that it quick and straightforward to install and/or deploy. Consequently, Edikt have developed tools which greatly simplify the deployment and installation of the Eldas Server and Clients respectively. This section will illustrate the Eldas Server tooling, the comments therein also being applicable to the Clients.

Figure 4. The Eldas Server Deploy Tool [21], is downloaded as an executable, self extracting Java Archive (JAR). Running this jar allows the user to graphically select a folder for unpacking the distribution. The top level of this distribution similarly contains an executable jar which launches a graphical interface to the deploy tool. This tool renders deploying Eldas, and configuring data resources, as simple as a few mouse clicks. The user simply has to identify the location of the application server and, if wanting a Grid Services as well as a Web Services interface, the location of the appropriate Globus Toolkit. Similarly data resources can be configured quickly using the interface. The whole process takes a matter of minutes as opposed to the hours required for other middleware solutions. The use of executable jars and Apache Ant scripts for deployment control make the tool completely

independent of the machine on which Eldas is to be deployed.



**Figure 4. The Eldas Server Deploy Tool**

### 5.4 Future Eldas Functionality

This section identifies some of the improvements and additional functionality likely to be included in future releases of Eldas.

#### *Interfaces to Eldas Data Service Invocation*

Future releases will additionally support the following interfaces.

- Grid Services - a WSRF interface will be supported in future releases. Whether or not Eldas continues to support an OGSI interface depends on the demands of our users.
- “Stateful” Web Services – a Web Services interface offering a lightweight alternative to Grid Services. This is lighter weight as there is no dependence on an underlying Grid framework such as Globus Toolkit, and will be a realisation of “Web Services plus identity, state and lifetime”.

#### *Supported Data Resources*

In addition to MySQL it is anticipated that future releases of Eldas will support IBM DB2, Postgresql, Oracle, SQL Server, MS Access, CSV Files and – through integration of Edikt's product BinX [22] – it will be possible to run queries on binary files. Invocation of stored procedures will also be possible where RDBMS vendors offer this functionality.

#### *Supported Application Servers*

Future release will, at the very least, also be deployable within IBM WebSphere.

#### *Clients and Tools*

Clients and tooling will continue to be developed and improved as the available Eldas Server functionality progresses.

### Data Integration and Data Transport

Through Edikt's application projects (see Application Areas) Eldas is developing some preliminary support for data integration, principally client/application level support for coordinating the results from distributed data sources. This has also led to some basic functionality for supporting data transport via FTTP rather than returning results directly to the client.

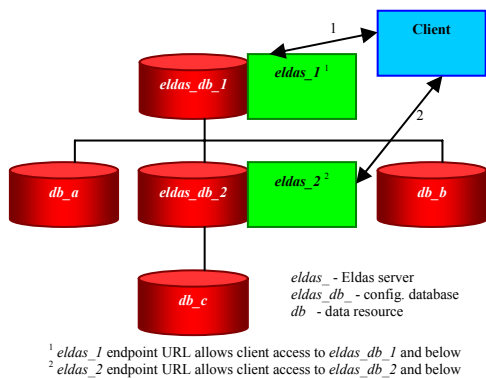
### Documentation, Performance, Scalability and Security

These areas will keep step with additional Eldas functionality and the needs of our application projects and users.

### Configuration

Eldas data resources are presently configured by reading text files at server start-up. Eldas will in future store resource configuration data in a database. Such databases will also contain access information for the associated Eldas Server. This has the following advantages.

- Data resources can be added dynamically.
- The configuration database can contain data other than configuration data e.g. metadata and registry type data.
- Eldas Servers can be represented as data resources themselves.



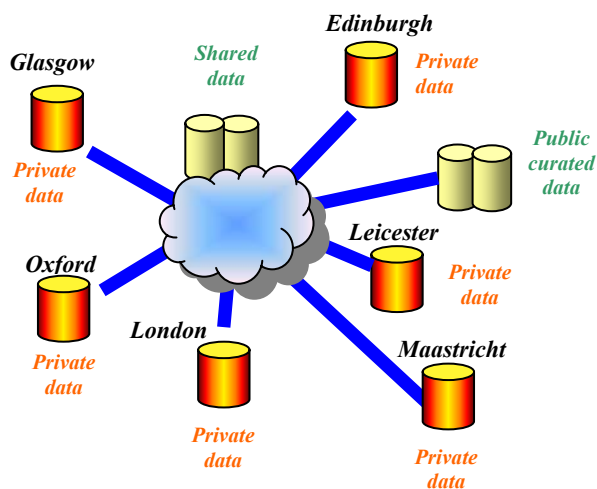
**Figure 5. Multiple Eldas Servers and Distributed Data Resources**

The last point is interesting as it effectively means that Eldas Servers can view other Eldas Servers as simple data resources like any other. Hence clients can access multiple servers, each one an endpoint to different data resources. See Figure 5. Multiple Eldas Servers and Distributed Data Resources for example, where *eldas\_db\_1* - which is the configuration

database for *eldas\_1* - contains information about *db\_a*, *db\_b* and *eldas\_db\_2*, while *eldas\_db\_2* contains data about *db\_c*.

## 6. Application Areas

Eldas is available as a product for download [23], but is also a key component in Edikt's application projects including BRIDGES and EdSkyQuery-G [24]. These projects place particular emphasis on data integration as well as data access, see Figure 6. Location of Data Sources used in the BRIDGES Project.



**Figure 6. Location of Data Sources used in the BRIDGES Project**

## 7. Conclusions

The J2EE framework, with emphasis on the use of EJB Session Beans, has been shown to be an excellent platform upon which to build robust and scalable service oriented data access middleware.

We have discussed the advantages this architecture offers and how it allows Eldas to meet certain essential requirements such as: machine and interface independence; access to heterogeneous data resources; robustness and performance. The recent movement in the Web and Grid Services communities from OGSi to WSRF makes the issue of interface independence particularly important and timely. The migration between these two sets of specifications will be straightforward for Eldas.

Ease of deployment and use has also been identified as a fundamental requirement. Hence Edikt have made considerable investment in tooling, ensuring Eldas is quick and easy to deploy, install and use. This distinguishes Eldas from other Grid middleware solutions.

## 8. Acknowledgements

Edikt is supported by a research development grant from the Scottish Higher Education Funding Council. It is based at the National e-Science Centre in co-operation with EPCC, both at the University of Edinburgh.

## 9. References

- [1] M.A. Antonioletti et al, *Grid Database Service Specification*, Sept 19, 2003, <http://www.cs.man.ac.uk/grid-db/papers/draft-ggf-dais-spec-ggf9.pdf>
- [2] DAIS Working Group, <http://www.cs.man.ac.uk/grid-db>
- [3] Global Grid Forum, <http://www.gridforum.org/>
- [4] Web Services, W3C, <http://www.w3.org/2002/ws/>
- [5] S. Tuecke et al, *Open Grid Services Infrastructure (OGSI) Version 1.0*, June 27, 2003, [http://www.globus.org/research/papers/Final\\_OGSI\\_Specification\\_V1.0.pdf](http://www.globus.org/research/papers/Final_OGSI_Specification_V1.0.pdf)
- [6] The Globus Alliance, <http://www.globus.org/>
- [7] Karl Czajkowski et al, *The WS-Resource Framework Version 1.0*, May 3, 2004, <http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf>
- [8] Sun Microsystems, Inc., Java 2 Platform, Enterprise Edition (J2EE), 2004, <http://java.sun.com/j2ee/>
- [9] Sun Microsystems, Inc., Enterprise Java Beans Technology Downloads and Specifications, 2003, <http://java.sun.com/products/ejb/docs.html>
- [10] The Edikt Project, <http://www.edikt.org/>
- [11] The National e-Science Centre, <http://www.nesc.ac.uk/>
- [12] Eldas, The Edikt Project, <http://www.edikt.org/eldas/>
- [13] Globus Toolkit 3, The Globus Alliance, <http://www-unix.globus.org/toolkit/downloads/3.2/>
- [14] Aileen Campbell, Alan Gray, Brian Hills, Stephen Rutherford, Davy Virdee, Rob Baxter, *Eldas (Enterprise Level Data Access Services)*, GGF11, June 2003, <http://www.doc.ic.ac.uk/~sjn5/GGF/GGF11/BGBS-Eldas.pdf>
- [15] JBoss Inc., The JBoss Application Server, <http://www.jboss.org/products/jbossas>
- [16] Rob Baxter, Denise Ecklund, Aileen Fleming, Alan Gray, Brian Hills, Stephen Rutherford and Davy Virdee, *Designing for Broadly Available Grid Data Access Services*, AHM 2003, Sept 2003, <http://www.nesc.ac.uk/events/ahm2003/AHMCD/pdf/023.pdf>
- [17] The Apache Ant project, <http://ant.apache.org/>
- [18] Steve Tuecke et al, *Grid Service Specification*, July 17, 2002 <http://www.globus.org/research/papers/gsspec.pdf>
- [19] Globus Toolkit 3, *Message Level Security*, [http://www-unix.globus.org/toolkit/3.0/ogsa/docs/message\\_security.html](http://www-unix.globus.org/toolkit/3.0/ogsa/docs/message_security.html)
- [20] The Eldas Query Tool, The Edikt Project, <http://www.edikt.org/eldas/eldasqt.htm>
- [21] The Eldas Server Deploy Tool, The Edikt Project, <http://www.edikt.org/eldas/eldasdt.htm>
- [22] Edikt, BinX, <http://www.edikt.org/binx/>
- [23] Eldas 1.0, The Edikt Project, <http://www.edikt.org/eldas/download/index.htm>
- [24] Aileen Campbell, Brian Hills, Rob Baxter, Alan Gray, Denise Ecklund, Stephen Rutherford, Davy Virdee, Tom Sugden, Bob Mann, Richard Sinnott, *A Federated Architecture for Data Access and Integration Services in e-Science*, AHM 2004, Sept 2004.