

**UK Role in Open Grid Services Architecture
UK e-Science Core Programme
Draft: Please do not Circulate**

Version 0.7
12th March 2002

Prepared by
The UK e-Science Architecture Task Force
for
The UK e-Science Technical Advisory Group

Malcolm Atkinson
Vijay Dialani
Ken Moody

Jon Crowcroft
Andrew Herbert
Steven Newhouse

David De Roure
Ian Leslie
Tony Storey¹

¹ Contact Information may be found in Section 25.

Table of Contents

| | | |
|-----------|---|-----------|
| 1 | Executive Summary | 4 |
| | Introduction..... | 5 |
| 2 | Document History | 5 |
| 2.1 | <i>The e-Science Context.....</i> | 6 |
| 2.2 | <i>The Industrial Context</i> | 6 |
| 2.3 | <i>Structure of this Document</i> | 8 |
| | Background, Motivation and Contribution..... | 10 |
| 3 | Web Services | 10 |
| 4 | Grid Services and OGSA | 12 |
| 4.1 | <i>Grid Service Openness.....</i> | 14 |
| 4.2 | <i>Grid Service Portability.....</i> | 14 |
| 4.3 | <i>Features of an Open Grid Service Architecture</i> | 15 |
| 4.4 | <i>Grid Service Lifetimes.....</i> | 16 |
| 4.4.1 | <i>Challenges for Lifetime Management.....</i> | 17 |
| 4.5 | <i>Delimiting the Scope of OGSA.....</i> | 18 |
| 4.6 | <i>The Open Grid Services Infrastructure.....</i> | 18 |
| 4.7 | <i>OGSA's Expected Development Path</i> | 19 |
| 5 | Case for OGSA..... | 19 |
| 6 | The Challenge of OGSA | 21 |
| 7 | UK's OGSA Contributions | 22 |
| | Phase I: Goals and Activities | 24 |
| 8 | Understanding, Validating and Refining OGSA..... | 24 |
| 9 | Establishing Common Infrastructure | 25 |
| 9.1 | <i>Grid Services Hosting Environment</i> | 25 |
| 9.2 | <i>Standard Types</i> | 26 |
| 9.3 | <i>Grid Service Primitives.....</i> | 27 |
| 10 | Pilot Projects — Education and Consultation..... | 27 |
| 11 | Baseline Database Integration | 28 |
| 12 | Validation and Testing of Grid Services and Platforms..... | 29 |
| 13 | Baseline Logging Infrastructure..... | 30 |
| | Phase II: Goals and Activities..... | 31 |

| | | |
|-----------|--|-----------|
| 14 | Refining and Exploiting G-WSDL | 31 |
| 15 | Higher-level Data Integration | 32 |
| 16 | Advanced Database Integration | 33 |
| 17 | Scheduling for Workflow Optimisation..... | 34 |
| 18 | Computation Economies | 35 |
| | Phases I and II: Trustable Services | 36 |
| 19 | Performance Engineering | 36 |
| 20 | Dependable Engineering | 36 |
| 21 | Engineering for Change | 37 |
| 22 | Manageability and Operations Support | 37 |
| 23 | Privacy, Ethics and Legal Issues..... | 38 |
| | Summary and Supporting Information | 39 |
| 24 | A Programme of OGSA Developments..... | 39 |
| 25 | Contact Information | 39 |
| | Glossary | 40 |
| | Bibliography | 41 |
| | Appendix 1: Requirements for an Architecture..... | 44 |

1 Executive Summary

The UK e-Science Core Programme will focus on architecture and middleware development in order to contribute significantly to the emerging Open Grid Services Architecture (OGSA) [TCF+02, FKNT02]. This architecture views grid technology as a generic integration mechanism assembled from Grid Services (GS), which are an extension of Web Services (WS) to comply with additional grid requirements. The principal extensions from web services to grid services are the management of state, identification, sessions and lifecycles, and the introduction of a notification mechanism in conjunction with grid service data elements.

By engaging vigorously in the development and exploration of these architectures the UK will boost the productivity and success of its e-Science teams and develop substantial components that fit within that framework that will be valued by industry as well as by researchers.

The UK e-Science Programme has many pilot projects that require integration technology and has an opportunity through its Core Programme to lead these projects towards adopting OGSA as a common framework. That framework must be suitable, for example it must support adequate grid service interoperability and portability. To enable the e-Science pilot projects to benefit from the advent of OGSA, the Core Programme must:

1. Explain and document the OGSA for pilot projects and, by working closely with pilot projects, refine and record their requirements from OGSA.
2. Encourage the adoption by pilot projects of software engineering processes that can intercept the emerging technologies.
3. Recommend interim strategies that allow pilot projects to develop while OGSA itself is developed.
4. Contribute substantially to the definition of OGSA and to the pioneering of related implementations.
5. Lead the e-Science community in making good use of emerging middleware technology and standards.
6. Adopt a family of grid services for which the UK is primarily responsible and deliver their reference implementations.

The UK e-Science and computing science community is well placed to contribute substantially to data integration services [Wat02, Pea02, PAD+02]. Richer information models should be introduced at the earliest opportunity to progressively approach the goal of a semantic grid [DRJS01].

We therefore recommend that the UK e-Science Core Programme choose this focus and promptly allocates resources to its development. It requires urgent action because pilot projects will make most strategic decisions in the next six months and because during the early stages of the definition and prototyping of OGSA we will have the best opportunity to have influence and to establish a major UK role.

This proposed strategy is based on initial discussions with representatives of IBM and with the Globus design team. The design of OGSA is developing rapidly, and the roles of individuals and organisations are still emerging. It is therefore essential to maintain close liaison with other participants and to be prepared for substantial revisions to the plan outlined here.

These recommendations pertain to the next three to five years of effort. A longer-term view is taken in the research agenda, "Realising a broad view of e-Science" [Rod02]. These two recommendations share a common vision and are closely articulated. This "architectural road map" prepares for the research agenda, and will benefit greatly from input from that research in future years.

Introduction

This section of the report presents background information and indicates the basis for the decisions conveyed in the rest of the report. The history of discussion leading to the report is followed by a view of e-Science trends. We then discuss the current industrial activity relevant to UK e-Science Core Programme planning, and its implications for our plans. This section concludes by presenting the structure of the remainder of the report.

2 Document History

The UK e-Science Architectural Task Force (ATF) was established by the UK e-Science Core Programme in October 2001 with the remit of developing an “architectural road map”. This document represents the first version of our development of that goal.

This document derives from meetings with the Globus team (Dr. Ian Foster & Dr. Steven Tuecke) at Argonne National Laboratory (ANL) 3rd December 2001, from a subsequent meeting with IBM (Dr. Jeffrey Nick, Dr. Jeffrey Frey & Dr. Jean-Paul Prost) with Foster and Tuecke at O’Hare 4th December 2001, a meeting of “Grid Luminaries” called by Irving Wladawsky-Berger in London, 5th December 2001 and a meeting at the UK e-Science Institute (eSI) 12-14th December 2001. The latter meeting included the UK e-Science Database Task Force (DBTF) and the UK e-Science Architecture Task Force (ATF), Ian Foster, Tony Hey and Ken Moody². The full list of participants is given in the table below. It developed the essence of the proposals presented here and the discussion of architectural requirements given in Appendix 1.

| Name | Role | Affiliation |
|---------------------|---|---------------------------------|
| Dr. R. Allan | Member of the Grid Support Centre & ETF | CLRC Darsbury |
| Prof. M.P. Atkinson | Chair ATF & Member DBTF | NeSC |
| Dr. D. Boyd | Chair ETF & member Grid Support Centre | CLRC RAL |
| Dr. J. Crowcroft | Member ATF, Member GNT, Member TAG | Univ. of Cambridge |
| Dr. D. De Roure | Member ATF | Univ. of Southampton |
| Prof. I. Foster | OGSA expert advisor | Univ. of Chicago & ANL |
| Dr. A. Herbert | Member ATF | Microsoft Research Cambridge |
| Prof. A.J.G. Hey | Director of UK e-Science Core Programme | EPSRC & Univ. of Southampton |
| Dr. D. Hutchison | Chair of Grid Network Team (GNT) | Univ. of Lancaster |
| Dr. K. Moody | Member of ATF | Univ. of Cambridge |
| Dr. S. Newhouse | Member of ATF, Member of ETF | Imperial College of STM |
| Prof. N.W. Paton | Chair of DBTF | Univ. of Manchester |
| Dr. D. Pearson | Member of DBTF | Oracle UK |
| Dr. A.R. Storey | Member of ATF & of DBTF, e-Science SC | IBM United Kingdom Laboratories |
| Prof. P. Watson | Member DBTF | Univ. of Newcastle |

Contributors to the meeting at the e-Science Institute 12-14th December 2001

A subsequent meeting of the ATF (at ICSTM 30th January 2002) reviewed a version of this document, endorsed its general recommendations and suggested several improvements.

Version 0.4 followed the presentations of OGSA at a panel session in GGF4 (Toronto) 18th February 2002, and a BoFs to establish an *Open Grid Services Infrastructure Working Group* (OGSI WG) and a *Databases and the Grid Working Group* (DAG WG), that same day. It also takes advantage of discussions at ANL with Ian Foster and Steve Tuecke 21st February and at ISI 25th to 27th February with Carl Kesselman and his team.

The current version follows detailed review by the ATF at ICSTM on 5th March 2002.

² Others attended for part of the time: David Boyd and Rob Allan, representing the Grid Support Centre, and David Hutchison, representing the Grid Network Team (GNT).

2.1 The e-Science Context

The fundamental goal of the e-Science programme is to enable scientists to perform their science more effectively. The methods and principles of e-Science should become so pervasive that scientists use them naturally whenever they are appropriate just as they use mathematics today. The goal is to arrive at the state where we just say “science”. Just as there are branches of mathematics that support different scientific domains, so will there be different branches of computation that support different domains. We are in a pioneering phase, where the methods and principle must be elucidated and made accessible, and where the differentiation of domain requirements must be explored. We are confident that, as with mathematics, these results will have far wider application than the scientific testing ground where we are developing them.

The transition that we are catalysing is driven by technology and is largely manifest in the tsunami of data. Detectors and instruments benefit from Moore’s law, so that in astronomy for instance, the available data is doubling every year [Gray01, SKP+00]. Robotics and nano-engineering accelerates and multiplies the output from laboratories. For example, the available genetic sequence data is doubling every 9 months [Vent01]. The volume of data we can store at a given cost doubles each year. The rate at which we can move data is doubling every 9 months. Mobile sensors, satellites, ocean-exploring robots, clouds of disposable micro-sensors, personal-health sensors, combined with digital radio communication are rapidly extending the sources of data.

These changes warrant a change in scientific behaviour. The norm should be to collect, curate and share data. This is already a trend in subjects like large-scale physics, astronomy, functional genomics and earth sciences. But perhaps it is not yet as prevalent as it should be. For example, the output of many confocal microscopes, the raw data from many micro-arrays and the streams of data from automated pathology labs and digital medical scanners, do not yet appear as a matter of course for scientific use and analysis. It is reasonable to assume that if the benefits of data mining and correlating data from multiple sources becomes widely recognised, more data will be available in shared, often public, repositories.

This wealth of data has enormous potential. Frequently data contains information relevant to many more topics than the specific science, engineering or medicine that motivated its original collection. If we are able to study these large collections of data for correlations and anomalies they may yield an era of rapid scientific, technological and medical progress. But discovering the valuable knowledge from the mountains of data is well beyond unaided human capacity. Sophisticated computational approaches are needed carefully guided by the skills of scientists, technologists, computer scientists, statisticians and many other experts. Our challenge is to enable both the development of the sophisticated computation and the collaboration of all of those who should steer it. At the same time, we must make the whole process attainable by the majority of scientists, sustainable within a typical economy and trustable by scientists, politicians and the general public.

The challenge of making good use of growing volumes of diverse data is not exclusive to science and medicine. In government, business, administration, health care, the arts and humanities we may expect to see similar challenges and similar advantages in mastering those challenges. Basing decisions, judgements and understanding on reliable tests against trustworthy data must benefit industrial, commercial, scientific and social goals. It requires an infrastructure to support the sharing, integration, federation and analysis of data.

2.2 The Industrial Context

We consider only two factors here from the extensive forces of industry and commerce that bear on our plans.

1. Recent industrial response to the grid.

2. The industrial production of software platforms and middleware.

At the Supercomputing Conference in November 2001, eleven companies made a joint press announcement describing their commitment to develop grid computing. At the same time, IBM announced a major bioinformatics grid collaboration in North Carolina. Since then, they announced their partnership in the digital mammography grid, led by the University of Pennsylvania. Platform Computing have announced that they will “Red Hat” Globus. IBM announced their commitment to converging web services and the grid at GGF4. These samples, from a much larger population of companies making statements about supporting grid computing, are evidence that industry, including major providers of software, is taking grid computing seriously and that, in the medium term it will probably be a major provider of relevant software and services. It is already the only provider of relevant equipment.

In this document we propose that the UK e-Science Core Programme invests heavily in OGSA, which is based on a confluence of web service middleware developments and on grid middleware development. Both development paths will continue, but it is important to recognise the high volume and pervasive nature of the industrial investment in web services development. To illustrate this, the table below indicates some of the languages that have been, or are being developed to describe web service computation by major companies and which are being presented as proposals for W3C recommendations.

| Language | Comments |
|-----------|---|
| WSDL | Web Services Description Language [CCMW01], describes operations and bindings |
| WSIL | Web Services Inspection Language [Bri01], collects references to descriptions |
| SOAP | Simple Object Access Protocol 1.2 [Mit01], baseline invocation model |
| WSC | Web Services Choreography, used to describe combinations of web services [XXX] |
| WSEL | Web Services Endpoint Language [XXX] describes properties of WS implementations |
| SAML | Security Assertion Markup Language, see http://www.oasis-open.org/committees/security/ . |
| WScontext | Describes the context of a computation, such as its history and transaction id. [XXX] |
| WStrans | Describes transaction behaviour [XXX] |
| WScoord | Describes the coordination between web services [XXX] |
| UDDI | Capability based registration and discovery system [XXX] |

Some of the languages which groups of companies have or are developing for web services

An important aspect of web services is that the high-level descriptions of web services, of web service invocation, of registration and discovery can be dynamic. Whether dynamic or static, bindings to web services are often implemented by tools that optimise away the generality and exploit particular properties of invocations. There are many company supplied generic web service tools and components, e.g. Apache axis, Web Sphere, Visual .net, etc. There are well-written reference books, for example [GSB+02]. Already there is substantial investment and products. This will lead to a wealth of well-supported middleware that supports integration, distribution and sophisticated computation within three years.

Combining this with industrial interest in grids leads us to expect industrially supplied and supported scalable, high-performance distributed-integration middleware within five years. As this will be adequate to support *most* of the e-Science requirements, it raises the question, “Why spend research council and DTI funds on work that industry will do better within five years?” In answering that question we draw some lessons to guide the choice of e-Science Core Programme investment.

1. Where industrial software exists that meets an e-Science need it should be used by e-Scientists. The reasons are: that this is far less expensive than building bespoke generic software and then supporting and maintaining it; that it will result in far more rapid pilot project development, and that it will assist with the exchange of ideas and partnership between researchers and industry.
2. There may be work, which should be undertaken now, because it has not yet been done by industry to avoid e-Scientists waiting. In many cases, that will be overtaken in five years by industrial products. So it should be conducted to ease the intercept

with the emerging industrial middleware and to influence industry to adopt appropriate requirements for these products.

3. There is work that should be undertaken because it will not be done by industry. In this case, care must be taken to produce work that articulates well with the contemporary industrial progress.
4. It is essential to avoid unnecessary divisions and diversity. For example, the set of languages being commercially developed (see table above) is already difficult to assimilate and coordinate. Adding variations unnecessarily through divergent development will result in high costs.

There are examples of all these issues in the proposed development path. They deserve serious consideration. For example, many academic projects dismiss the complexity of some of the middleware technologies as unnecessary. But repeated experience has shown that as application sophistication, heterogeneity and scale increase, all of the mechanisms are necessary. Retrofitting such mechanisms is usually prohibitively labour intensive if it can be achieved at all.

Another reason why research projects shun existing technology is a failure to realise the economic importance of general and flexible solutions. By generating bespoke solutions to handle their current task they use skilled research labour to generate a product that only has one use. They contribute little to solving other e-Science tasks and little to solving their own future problems as their science evolves.

Finally science communities build their own software because the real cost of that task is hidden and the costs being store up for the future are unrecognised. Industry recognises the need to amortise these costs over many applications that are generic. But to the research group these have two costs that seem prohibitive: software licence costs and the cost of learning to use another technology.

We argue strongly that through education about the potential of available software and through improved commercial agreements for licensing the relevant software e-Scientists should be strongly encouraged to work with high-quality, high-level industrially generated standards and tools. The products that are developed by the e-Science programme should be relative to that high-level supported platform.

To summarise, industry is actively developing web services hosting products as commercial endeavours. It is important that UK e-Science activities leverage these rather than duplicate them for several reasons. First, the effort required to catch up and keep up in a fast moving area is immense. Second, most vendors are offering academic licenses to enable research use of their products and have programmes (e.g., IBM's Alphaworks) to give the academic community access to technology on the same path but ahead of their products. Thirdly, when research Grid projects spin out commercial ventures these can move seamlessly to full commercial license platforms with the levels of support required for business applications. The challenges for the ATF are to communicate the benefits of OGSA in particular and the web services philosophy in general to the e-Science community

2.3 Structure of this Document

The remainder of the document is structured in four parts. The next part, *Background, Motivation and Contributions*, comprises sections 3 to 7. It looks at the technical context of web services and the proposal for the Open Grid Services Architecture (OGSA). It then reviews some challenges that need to be addressed and presents an overview of the contributions that may be made by the UK to grid middleware, with a particular concern for data integration.

The part, *Phase I — Goals and Activities*, comprising sections 8 to 13, presents a number of activities that are already underway, or are proposed for an immediate start. These are

concerned with assessing the details of the OGSA and the infrastructure to support it (OGSI). They are also concerned with interaction with pilot projects to establish requirements and to develop understanding of the potential of OGSA-based data access and integration middleware. Finally, they identify two immediate implementation goals, to support database connection and to develop a grid economy.

The part, *Phase II — Goals and Activities*, comprises sections 14 to 18 and covers activities that should overlap with those of phase I, but be started later. They are concerned with: refining and exploiting the description languages used to describe grid services, developing higher-level (semantic) models of data and grid components, providing advanced data integration systems, and developing schedulers that take account of data access and integration requirements. These activities build on those of Phase I, which will continue where they do not merge with those of Phase II.

The part, *Phases I & II — Quality of Service*, comprises sections 19 to 23. It presents pervasive concerns that must be considered and incrementally addressed during phases I and II. They cover such issues as: performance, dependability, support for change and privacy.

The remainder of the document provides a summary (§24), contact information (§24), a Glossary, References and Appendices.

Background, Motivation and Contribution

Two lines of technological advance: web services and the grid, and their initial confluence in the Open Grid Services Architecture (OGSA) are introduced. The potential value of OGSA is recognised. As it is a design that is undergoing change, some of the challenges yet to be fully resolved are enumerated. Both grid research and the multi-company efforts to develop web service standards will continue to make substantial progress. The current status of OGSA, summarised here, should therefore be read as a snapshot, as it will certainly evolve as implementations address the challenges and as both streams of development feed it with new ideas and technologies. The proposal that the UK e-Science Core Programme engages vigorously with this dynamic process with a view to delivering a powerful family of data integration grid services, concludes this part of the document.

3 Web Services

This section reviews some aspects of web services from the viewpoint of e-Science and in preparation for the description of grid services.

Web services (WS) are an emerging integration architecture designed to allow independently operated information systems to intercommunicate. Their definition is the subject of W3C standards processes in which major companies, e.g. IBM, Oracle, Microsoft and SUN are participating actively. Web services are described well in [GSB+02], which offers the following definition:

“A Web service is a *platform and implementation independent* software component that can be:

- *Described* using a service description language
- *Published* to a registry of services
- *Discovered* through a standard mechanism (at runtime or design time)
- *Invoked* through a declared API, usually over a network
- *Composed* with other services”

Web services are of interest to the UK e-Science community on three counts:

1. Their function of interconnecting information systems is similar to the grid's intended function. Such interconnection is a common requirement as scientific systems are often composed using many existing components and systems.
2. The support of companies for web services standards will deliver software development tools to enable rapid development of grid services and applications, a standard framework for describing and composing web services and grid services and a potential route to commercialisation.
3. Several of the UK e-Science groups already have experience of web services and several pilot projects are already committed to using web services.

The principal features of web services are introduced for use in the remainder of this document. A standard web service is similar to an interface to an object, in that it is defined as being capable of supporting a number of method calls and is virtual, in the sense that it permits multiple implementations. It differs from an interface in an object-oriented language in several important ways, e.g.

- There is a specific *set* of languages (WSDL, WSIL, ...) for describing a web service (see also §2.2), which may then be implemented in any language. In contrast, in an object-orientated programming language the same language is used both to describe the interfaces supported by an object and their implementation. This simplifies things for the programmer, but makes interoperability between languages difficult. By contrast, web service interface definitions describe interfaces using a high level semantic model, which can then be mapped to specific interface description syntaxes

and protocols onto the primitives of your model. This should give enough power to be able to automatically build adaptors, interceptors or what have you to interoperate across heterogeneous protocols and tool chains. RM-ODP is an example of this model-based approach [ISO-10746, Put00].

- Web services separate their API from their invocation mechanism so that operations may be invoked by a variety of protocols and mechanisms.
- The nomenclature is different.
- There is currently no explicit specification of mechanisms for managing web service instances and their lifetimes.
- The functionality of a web service is a composition of a set of *port types*. Each port type is a set of operations, each of which has a signature. Different ports may be invoked using different protocol bindings. Where as, in most object-oriented languages an object has a flat set of methods and a uniform invocation mechanism.
- Web services support a capability-based web service discovery mechanism.
- Web services are dynamically bound. This remains logically the case even after optimisation to exploit common contexts and platforms.

The main features of a web service are:

1. A web service is described using the Web Service Description Language (WSDL) [CCMW01].
2. A web service receives and sends messages containing document-oriented or procedure-oriented information, i.e. it can operate in message mode (a response is asynchronous) or in RPC mode (a response is synchronous and the structure of the message body corresponds to the parameters of the called procedure).
3. Each web service, port and message is described abstractly and then made concrete by a *binding*.
4. *Types* are used to define the information carried by a message. They are normally defined using an *XML Schema* defined in an XSD [W3Ca] and normally take types from a schema *namespace* [W3Ca] which allows the same types to be used in many places.
5. A *message* consists of a sequence of one or more *parts*. Each part is of a specified type. This defines an abstract message, which can be named and then multiply used, e.g. it may carry the parameters for an operation.
6. An *operation* takes a message as input and may deliver a message as output.
7. A *port type* is a set of operations.
8. A *binding* is a concrete protocol and data format for a particular port type.
9. A *port* is the composition of a binding with a network address.
10. Each web service is a collection of related *ports*.

This is a summary of the present status of web services, but they will of necessity develop further to serve commercial requirements, e.g. to provide transactional coordination, workflow-based integration and other forms of orchestration. They will also need to be developed to meet e-Science requirements, e.g. to use namespaces with types suitable for scientific data.

The present bindings associate a format (representation of the type's values) and protocol with a port. This binding may also determine the security mechanism protecting the communication as operations are invoked. A popular example is Simple Object Access Protocol (SOAP) [Mit01] using HTTP with an XML data encoding. This focus suits current

commercial needs as the initial applications use relatively small volumes of data and have to operate between organisations protected by firewalls. Efficient mechanisms have been developed for use within organisations and on single platforms, including connection by shared memory within one operating system regime. Such optimised bindings are essential for high-performance, high-volume applications in e-Science and in commerce. Bindings are determined independently for each connection, and may be constructed dynamically. WSIF [Muk01] is suggested as a mechanism that provides dynamic invocation.

In summary, a binding determines three independently varying concrete properties: *the encoding, the transport protocol and the security*, for each connection between a client and a port.

It is well known that failure handling is a necessary part of interconnection architectures in distributed systems. This is not yet systematically addressed for web services. SOAP provides a means of returning failure information and conventions for that information. But the definitions in WSDL do not yet describe the potential failures nor what they imply for the computation. Such additional information is necessary to support graceful degradation and recovery.

A web service offers a set of operations via its ports as defined above. There may be several implementations of that web service. A *provider* arranges that an implementation is executing on a particular platform and accessible via particular ports. The *client invokes* the operations it requires. There may be several instances of a particular web service operating at the same time, e.g. to handle load or supplied by alternative service providers. The client is unaware of replication and of mechanisms used by the provider to activate the service. Typically, in today's web services, providers manage the lifetime of the web services they provide, without specifying how this will be done, e.g. when the service will terminate, and without using a standard mechanism.

Distributed systems depend on protocols to provide interoperation between heterogeneous systems and on standard APIs for portability, i.e. to permit code or designs developed for one hosting environment to be re-used in another (See Appendix 1). Web services have addressed the protocol issue, but have yet to define standard APIs, such as those defined in Globus to permit process management, security, etc. Such standard APIs would clearly benefit web or grid service developers.

In summary, there is very substantial commercial impetus behind web services that is sufficiently strong that it is reasonable to suppose they will be an important and well-supported integration technology. Their definition is still evolving (§2.2).

4 Grid Services and OGSA

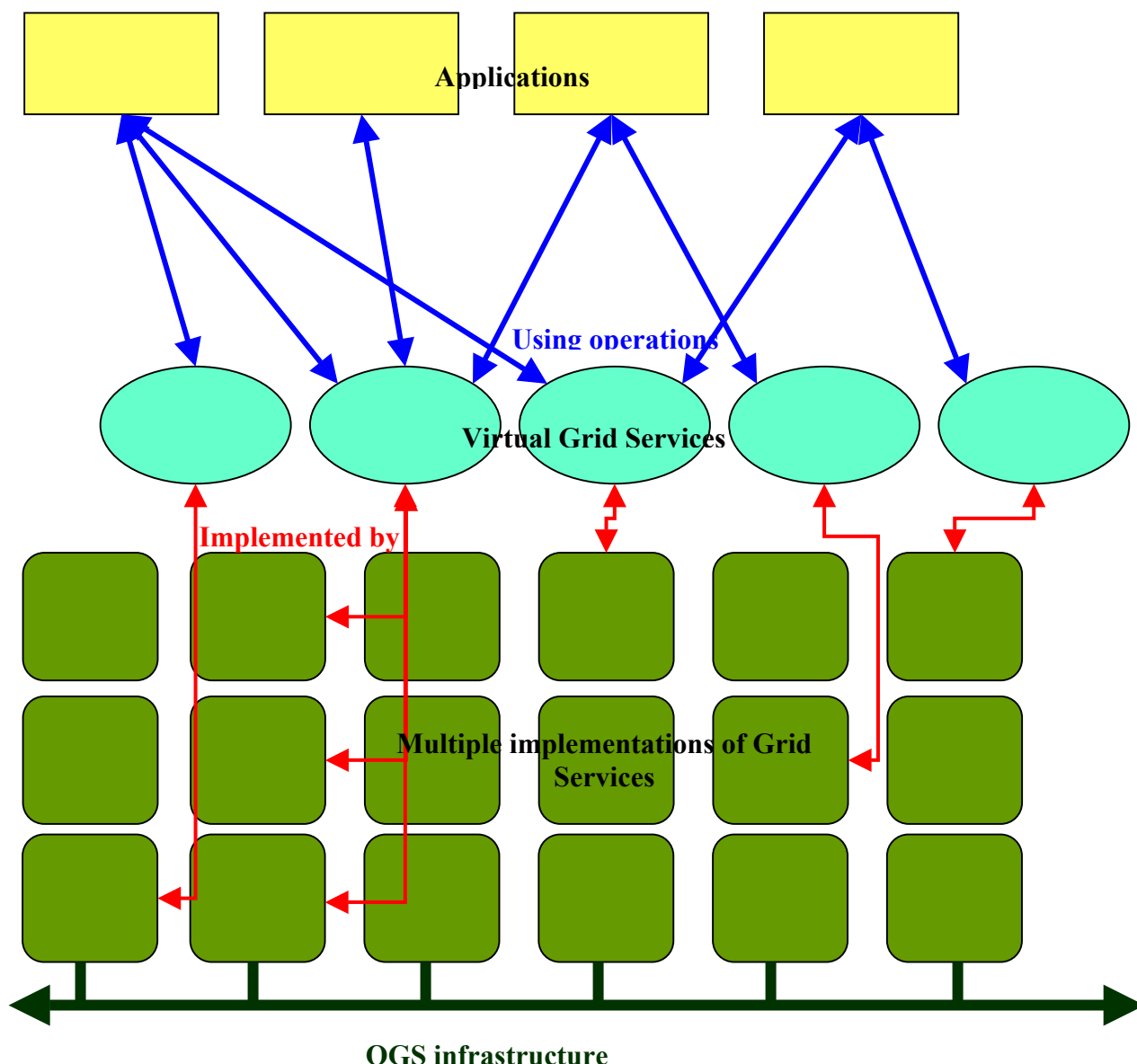
This section introduces the principal concepts of the open grid services architecture, names the parts needed for further discussion and delimits the scope of the open grid services infrastructure.

The Open Grid Services Architecture (OGSA) combines the emerging strengths of Grid technology [6] and the developing open standards of web services (§2.2 & 3). The concepts and technologies of *both* will be *developed* to deliver a coherent architecture. This architecture will be defined in terms of *Grid Services* (GS), which are extensions of web services, i.e. a grid service is also a web service. Advances in grid technology will flow from web service description, integration and engineering techniques. Advances in web services will derive from technologies pioneered for grids. That is, benefits are expected to flow in each direction. It is an aspiration of the designers of OGSA that it will continue to benefit from developments in both fields.

Grid services will combine to produce a common virtual infrastructure to support higher-level grid services. (Everything is viewed as a service in this model, including applications, which become web-services at the top level, directly delivering some user requirement, supported by

sufficient layers of composed grid services.) They will comply with a standard set of additional requirements for grid services. An initial definition was presented at GGF4 [TCF+02, FKNT02], but this definition is expected to evolve as the result of multi-party discussions and experiments. Sets of grid services will deliver functions that combine to meet the requirements of each application. Thus a grid service is defined as a virtual grid service and then bound to a concrete implementation, c.f. web services. The concrete implementations will draw on grid technology to deliver facilities required by e-Science projects.

An example of the resulting structure can be seen in the following diagram. In this example, several applications are supported on the set of provided grid services and use these according to their virtual operations defined in Grid-Web Service Definition Language (G-WSDL) an extension of WSDL³.



³ Papers [TCF+02, FKNT02] use WSDL throughout except in xmlns: definitions where gsdl is used. WSDL admits extensions, and standard extensions for grid aspects of GS will inevitably emerge. Therefore, we coin the term “G-WSDL” to capture the notion of these particular extensions of WSDL. G-WSDL has yet to be defined. It may be synonymous with gsdl. It *must not* cease to be WSDL.

4.1 Grid Service Openness

The grid services also inter-work via these virtual operations, so that the architecture is *open* in the sense that new grid services may be defined to inter-work with (or to compose) existing grid services using these published interfaces. That is, the client invoking a service does not need to know on what platform the service it is invoking is running, nor does it need to know in what language that service is implemented. This essential interoperability between heterogeneous components is intrinsic to both web service protocols and grid protocols. With dynamic invocation mechanisms the client is automatically provided with code that maps to and from the particular binding that is being used to support the invocation. In this case, the invocation of an operation is mapped to protocols that take advantage of commonalities (same address space, same operating system instance, same LAN, etc.).

Grid services are also open, in that many alternative implementations may be available for each virtual grid service definition and a new provider may offer a new implementation that delivers the specified operations, perhaps with different performance and reliability characteristics.

Finally, the grid service architecture is open, in that its infrastructure may be multiply implemented and these alternative implementations should interwork.

4.2 Grid Service Portability

To amortise development costs and increase the speed with which services become available to all e-Science projects, most implementations should be portable. That is, they should be implemented using technologies that run with exactly the same semantics on a wide range of platforms. We are interested here primarily in the portability of grid services rather than the portability of platforms and hosting environments set up to support them.

Two forms of portability are of interest. Where a grid service is implemented for a particular hosting environment, e.g. plain Java, J2EE or C in a Unix process, that environment can be replicated on top of various platforms and in various organisations. Provided it delivers the same APIs to support grid service implementation then each grid service implemented for that hosting environment can be replicated in each occurrence of that hosting environment. (We can assume that all of the hosting environment's standard operations are already replicated consistently.) To achieve this *code* portability within one hosting environment simply requires that the APIs provided to support grid service implementation are consistently defined and consistently represented *in that hosting environment*. This is clearly desirable, but, for reasons described below, it is not part of the OGSA itself. *The UK grid service implementation community will need to join with others to develop the definitions of these consistent APIs for the hosting environments with which they are concerned.*

The second form of portability we will call *design* portability. That is, the designs of grid services delivering the same operations via the same construction should move between different hosting environments. This has two advantages. As design is perhaps the hardest part of creating a good quality service, its re-use can be a substantial saving. Similarity between the functional behaviour of the different hosting environments, will reduce learning costs, increase sharing of ideas and community formation, and make it easier to deliver equivalent grid services in the different hosting environments. To achieve this *design* portability, then the APIs that support grid service implementation should have *consistent functionality across different hosting environments*, though they will inevitably have different representations. This is not yet explicitly part of the OGSA, though it should be. Their incorporation into the OGSA definition, or into the definitions of a basic set of grid services

commonly used with OGSA, is just a matter of time, as the definitions will, in part, derive from current Globus APIs, and these will evolve as the existing Globus services are redesigned and restructured for OGSA. *The UK grid service implementation community should play an active role in developing and defining these consistent APIs.*

4.3 Features of an Open Grid Service Architecture

The OGSA is an architecture that addresses wide and significant goals. It seeks to support the integration of heterogeneous software components and systems that may be distributed across globally dispersed diverse platforms operated by autonomous organisations. It plans to support requirements, such as security, performance and dependability across these platforms for a wide variety of dynamically determined virtual organisations and a wide variety of services. It seeks to retain the description, capability-based discovery and dynamic binding emerging in web services.

The following extract from section 6.1 of [FKNT02] provides more detail.

A basic premise of OGSA is that everything is represented by a *service*: a network enabled entity that provides some capability through the exchange of messages. Computational resources, storage resources, networks, programs, databases, and so forth are all services. This adoption of a uniform service-oriented model means that all components of the environment are virtual. More specifically, OGSA represents everything as a *Grid service*: a Web service that conforms to a set of conventions and supports standard interfaces for such purposes as lifetime management. This core set of consistent interfaces, from which all Grid services are implemented, facilitates the construction of hierarchal, higher-order services that can be treated in a uniform way across layers of abstraction.

Grid services are characterized (*typed*) by the capabilities that they offer. A Grid service implements one or more *interfaces*, where each interface defines a set of operations that are invoked by exchanging a defined sequence of messages. Grid service interfaces correspond to *portTypes* in WSDL. The set of *portTypes* supported by a Grid service, along with some additional information relating to versioning, are specified in the Grid service's *serviceType*, a WSDL extensibility element defined by OGSA.

Grid services can maintain internal state for the lifetime of the service. The existence of state distinguishes one *instance* of a service from another that provides the same interface. We use the term *Grid service instance* to refer to a particular instantiation of a Grid service.

The protocol binding associated with a service interface can define a delivery semantics that addresses, for example, reliability. Services interact with one another by the exchange of messages. In distributed systems prone to component failure, however, one can never guarantee that a message that is sent has been delivered. The existence of internal state can make it important to be able to guarantee that a service has received a message once or not at all, even if failure recovery mechanisms such as retry are in use. In such situations, we may wish to use a protocol that guarantees exactly-once delivery or some similar semantics. Another frequently desirable protocol binding behavior is mutual authentication during communication.

OGSA services can be created and destroyed dynamically. Services may be destroyed explicitly, or may be destroyed or become inaccessible as a result of some system failure such as operating system crash or network partition. Interfaces are defined for managing service lifetime.

Because Grid services are dynamic and stateful, we need a way to distinguish one dynamically created service instance from another. Thus, every Grid service instance is assigned a globally unique name, the *Grid service handle (GSH)*, that distinguishes a specific Grid service instance from all other Grid service instances that have existed, exist now, or will exist in the future. (If a Grid service fails and is restarted in such a way as to preserve its state, then it is essentially the same instance, and the same GSH can be used.)

Grid services may be upgraded during their lifetime, for example to support new protocol versions or to add alternative protocols. Thus, the GSH carries no protocol- or instance-specific information such as network address and supported protocol bindings. Instead, this information is encapsulated, along with all other instance-specific information required to interact with a specific service instance, into a single abstraction called a *Grid service reference (GSR)*. Unlike a GSH, which is invariant, the GSR(s) for a Grid service instance can change over that service's lifetime. Each GSR has an explicit expiration time, and OGSA defines mapping mechanisms, described below, for obtaining an updated GSR.

The result of using a GSR whose lifetime has expired is undefined. Note that holding a valid GSR does not guarantee access to a Grid service instance: local policy or access control constraints (for example maximum number of current requests) may prohibit servicing a request. In addition, the referenced Grid service instance may have failed, preventing the use of the GSR.

As everything in OGSA is a Grid service, there must be Grid services that manipulate the Grid service, handle, and reference abstractions that define the OGSA model. Defining a specific set of services would result in a specific rendering of the OGSA service model. We therefore take a more flexible approach and define a set of

basic OGSA interfaces (i.e., WSDL portTypes) for manipulating service model abstractions. These interfaces can then be combined in different ways to produce a rich range of Grid services.

From this it is evident that OGSA is extending the web architecture to include such topics as life cycle management, state management and session management which are particularly important for distributed scientific data processing. It can be assumed the web services community will find similar needs in other domains and therefore OGSA must participate in the discussion to avoid divergence. By exploring models such as the one proposed above, OGSA can lead these efforts.

This is likely to lead to refinement of OGSA as lessons are learned from the discussion. The UK with its strong experience in middleware architecture and implementation should play a significant role. For examples members of the UK community who have worked on distributed computing and persistent programming have insights into the abstraction of instantiation and invocation mechanisms, middleware architecture that could be introduced into OGSA to develop the architecture further and enhance it's capabilities.

There are two other interrelated properties of OGSA that are emerging in recent versions of the OGSA Specification [TCF+02]. These are: Grid Service Data Elements (GSDE) and notifications. A GSDE is an element of data that has a defined period of validity, and which represents some aspect of the operational state of the grid service that returns it. They have an XML definition that allows extension and nesting. There is a minimal name-based query language, and an option to extend this with more powerful languages.

The notification system allows a grid service to register interest in events based on the GSDE structures. These registrations have a time-out, so that there is not an indefinite obligation in the event of partial failures. These notification commitments have no affect on the lifetime of the service with which they are registered. When an event is detected, the notification is transmitted to all of the registered listeners.

This provides an example of an issue raised under portability (§4.2). The grid service detecting the event must transmit the notification to its listeners, but the API it should use to do this is not yet defined.

4.4 Grid Service Lifetimes

The introduction of GSH generates a reference structure, which at first sight might either require global distributed garbage collections (almost completely intractable in large-scale distributed persistent systems) or indefinite commitments. However, there is no commitment that the referenced grid service will still exist when a client tries to use it. A grid service is created for a specified period, determined by the factory that creates it, or by parameters supplied with the request that it be created. The former case is precisely analogous to the tradition of providers controlling lifetimes for web services.

There may be an additional contract for existence negotiated via a protocol for extending lifetimes, but the provider of the grid service decides precisely how much to extend the life of the grid service. The client of a grid service may also ask it to terminate itself. This is not mandatory as there may be other dependent clients.

In principle, at least, this means that services should not fail in a way that causes isolated resources to be permanently allocated. In practice it may not be that simple. The grid service that is terminating must have an opportunity to clean up, as it may have many resources, e.g. disk space, allocated on its behalf. But failures may occur during this clean up. There is also a live-lock that can preserve the allocation of unreachable resources. A cycle of requests between otherwise unreachable grid services can keep them alive. Several approaches can be taken to handling this, based on registering contracts with higher-level services.

4.4.1 Challenges for Lifetime Management

Preliminary discussions among the UK e-Science community have raised some concerns about the current lifetime management model. These are:

1. How does a client developed to use web services transport to use the same service delivered by a grid service?
2. What is the notion of instance and state?

There are likely to be many clients already developed, e.g. to visualise genome-proteome relationships, that will use existing web services, e.g. those provided by EBI. For efficient transition to using grid services such clients should be able to operate with both services. It is certainly the case that the cost of changing clients or maintaining two versions of clients is best avoided.

Existing web service clients do not deal with factories and GSHs. They invoke a service and any consequent creation of processes and state representations is hidden from them. Any state that requires long-term preservation on their behalf is stored in databases and is referenced by parameters in operations. It would therefore seem sensible to require this as the default behaviour. That is, the use of the factory and GSHs would be implicit. Introspection operations should allow the explicit use of factories and GSH, when this is necessary.

The exact definition of instance and state has always generated many challenges for large-scale, distributed and long-running computations [AJ00, JA00, Atk00]. The challenge is exacerbated when we consider evolution of long-running systems [ADHP00]. The EJB definitions try to delimit this by restricting *session beans* from exporting any references. The present definitions of OGSA appear to have several potential difficulties.

- Typically a service is used concurrently by several clients. It is not clear what state is shared by these and what state is replicated on a per-client basis. The mechanisms for avoiding conflicting use of that shared state are not defined. Are these all issues to be defined on a per service basis, or are there to be general standards?
- A grid service may pass a GSH to its clients, and they may pass that GSH to other services, as they compose services to implement their operations. If these third party services now act as clients of the referenced service, how does that client know these operation requests are all part of the same larger operation, which may determine whether they are allowed to operate on the same state? Again, this could be determined on a per service basis, or there might be general standards.
- The grid service, as it implements an operation for a client, will typically use operations delivered by other grid services. Hence the state changes in the transitive closure of the services used in delivering an operation are all *logically* part of one state change from the viewpoint of the client. The suggestion that most concurrency is avoided by generating new grid service instances exacerbates this problem. It is not possible to replicate the tree of invocations because there are intrinsically shared resources, such as processors, disks and data. The operations and definitions that concern state have to be clear about the extent of this transitive closure that they reference.

It may be useful to define the shared *internal* state that a grid service uses to deliver its operations and to discriminate this from an *activation* state that is created to support the evaluation of one operation on behalf of one client. *UK e-Science should contribute to the development of improved models of state and lifetime management via position papers, prototypes and engagement in the OGSI WG discussions.*

4.5 Delimiting the Scope of OGSA

It is helpful to clarify what OGSA is not, i.e., what it does not define and why these definitions are left open. Three definitions are useful.

A *hosting environment* is a language context from which the OGSA APIs may be used (§4.2). This may be C (which includes C++, PERL and Fortran in practice), J2EE, plain Java, or C#.

A *platform* is an operating system context supporting a hosting environment. Current examples are: Linux, Solaris, AIX and .NET.

An *administrative regime* runs one or more instances of platforms. Different administrative regimes operate different policies. For example a campus, such as the University of Michigan may issue 40,000 Kerberos certificates to its students to establish security and usage policies. The DOE e-Scientists or the UK e-Scientists may all use a single Certificate Authority policy for their security but may vary their authorisation and charging mechanisms according to more local administrative regimes.

In order that OGSA can span variations in hosting environments, variations in platforms and variations in administrative regimes, it is necessary to leave many aspects of the architecture unspecified. For example, authentication and authorisation mechanisms cannot easily be defined to cover that range. In some cases, a common standard specifying minimal functionality may eventually emerge, in others, higher-level services will need to accommodate and perhaps virtualise the variation.

It is open to any community, for example the UK e-Science community and a number of international collaborators, to choose to introduce additional structure and regularity. This might be through adopting regimes, platforms and hosts with particular properties, or by introducing a layer above OGSA.

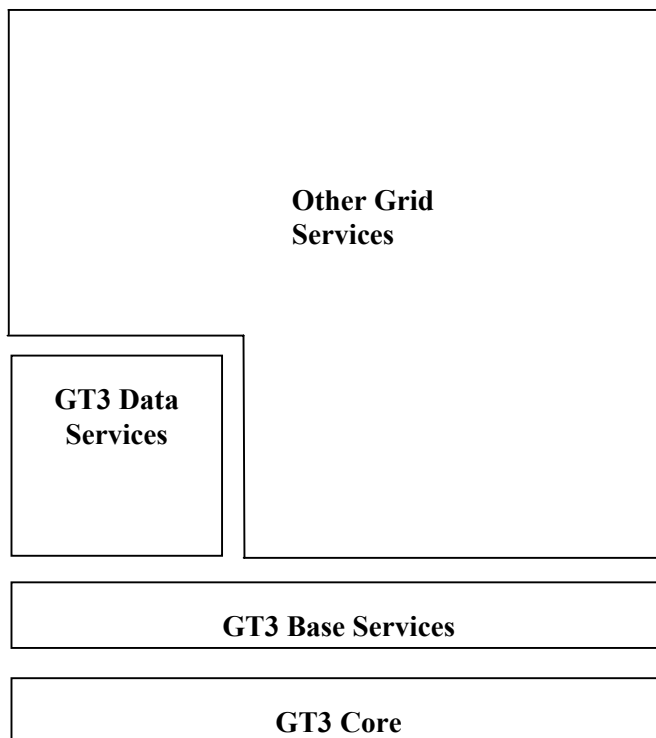
It should also be clear that not every use of web services within projects that develop grid applications is of concern to OGSA. Many projects, e.g. those developing problem solving environments, portals, etc., were already using web services as part of their integration technology. That use will continue. It is independent from OGSA, though it may be facilitated by OGSA making grid facilities more easily accessed by those expert in web services.

4.6 The Open Grid Services Infrastructure

The Open Grid Services Infrastructure is a minimal layer that will underlie all grid services and enable them to invoke each other via the grid and to invoke baseline grid operations. This corresponds to the GT3 Core and GT3 Base Services of the proposed Globus Toolkit Version 3 (GT3) described in a short paper at GGF4 [OGSA02]. The proposed reference implementation is shown in the diagram below. The GT3 Core will implement the interfaces and behaviours described in the *Grid Service Specification* [TCF+02]. The GT3 Base Services will exploit the GT3 Core to implement existing Globus Toolkit capabilities (e.g., resource management, data transfer & information services) and to add new capabilities (e.g., reservation & monitoring). A prototype of the GT3 Core in Java using Apache Axis was demonstrated at GGF4. It used a SOAP/HTTP/TLS+GSI protocol binding for transport.

The UK e-Science middleware developments proposed here will:

1. Provide early testing and feedback of OGSI prototypes.
2. Elucidate application requirements and feed these into the OGSI design.
3. Build prototype components of OGSI, particularly those supplying data services.



4.7 OGSA's Expected Development Path

Intensive discussions can be expected to continue for the next year developing the definitions and deciding the appropriate boundaries between that which must be defined to achieve a common virtual infrastructure that is useful, and that which must not be defined if the OGSA is to be sufficiently tolerant of the variations described above. These discussions will probably occur mainly in the context of the Open Grid Services Infrastructure Working Group of GGF (OGSI-WG). *It is important that the UK develops experience and contributions sufficiently rapidly that it can have an effective role in OGSI-WG.*

A summary of the Globus Project's *provisional* timetable for developing GT3 is given below, taken from [OGSA02].

Alpha release of GT3 Core: Estimate April 2002.

Alpha release of early GT3-Based Data Grid Services: Estimate June 2002.

Alpha release of GT3 Base Services: Estimate August 2002.

Public release of GT3 (supported beta): Estimated late 2002.

5 Case for OGSA

The authors of OGSA [TCF+02, FKNT02] expect the first implementation, Globus 3, to faithfully reproduce the semantics and APIs of the current Globus 2, in order to minimise the perturbation of current projects. However, the influence of the thinking and industrial momentum behind web services, and the need to achieve regularities that can be exploited by tools, will surely provoke profound changes in grid implementations of the future. Indeed, OGSA is perceived as an opportunity to restructure and re-engineer the Globus foundation technology. This will almost certainly be beneficial, but it will also surely engender semantically significant changes.

Therefore, because of the investment in existing grid technology (e.g. Globus 2) by many application projects the case for a major change, as is envisaged with OGSA, has to be compelling. The arguments for adopting OGSA as the direction in which to focus the development of future grid technology concern three factors: politics, commerce and technology.

The *political case* for OGSA is that it brings together the efforts of the e-Science pioneers and major software companies. This is essential for achieving widely accepted standards and the investment to build and sustain high-quality, dependable grid infrastructure. Only with the backing of major companies will we meet the challenges of:

- Installing widespread support in the network and operating system infrastructures,
- Developing acceptance of *general* mechanisms for interconnection across boundaries between different authorities, and
- Obtain interworking agreements between nations permitting exchange of significant data via the grid.

The companies will expect from the e-Science community a contribution to the political effort particularly through compelling demonstrations.

The *commercial case* is the route to a sustainable grid infrastructure and adequate grid programming tools, both of which are missing for the grid at present because the e-Science community's resources are puny compared with the demands of building comprehensive infrastructure and tool sets. If convergence can be achieved between the technology used in commercial applications for distributed software integration and that used for scientific applications, then a common integration platform can be jointly constructed and jointly maintained. As commerce is ineluctably much larger than the science base alone, this amortises those costs over a much larger community. Commerce depends on rapid deployment and efficient use of many application developers who are rarely experts in distributed systems. It therefore has strong incentives to build tool sets and encapsulated services that would also benefit scientists if we share infrastructure⁴.

A further commercial advantage emerges from the proposed convergence. It will be easier to rapidly transfer e-Science techniques to commerce and industry. Using a common platform, companies will have less novel technology to learn about, and therefore less assimilation costs and risks.

The *technological case* for OGSA is largely concerned with software engineering issues. The present set of components provided by the grid has little structure to guide the application developers. The discipline of defining grid services in terms of a language (G-WSDL) and of imposing a set of common requirements on each grid service should significantly improve the ease and accuracy with which components can be composed. Those same disciplines will help grid-service developers to think about relevant issues and to deliver dependable components. We expect significant families of grid services that adopt additional constraints on their definition and address a particular domain. Such families will have improved compositional properties and tools, which exploit these, will be a natural adjunct. Dynamic binding and rebinding with soft state are necessary for large-scale, long-running systems that are also flexible and evolveable. The common infrastructure and disciplines will be an appropriate foundation from which to develop:

- Tools, subsystems and portals to facilitate e-Science application development, taking advantage of the richer information available from the metadata describing grid services,

⁴ As we do today for operating systems, compilers and network IP stacks. The contrast between IP support and multicast support, currently of less interest to commerce, illustrates these effects well.

- Advances in the precision and detail of the infrastructure and disciplines to yield dependable, predictable and trustworthy services, and
- A richer set of well-documented functions, including the database services [Wat02, Pea02, PAD+02] and higher-level semantic grid services [DRJS01, NMF+02] of particular interest to UK e-Science.

6 The Challenge of OGSA

As section 4 explained, OGSA is still at an early stage of design and development. This provides opportunities. It also means that there are issues that have not yet been addressed in this context or that require further exploration. These are illustrated in this section.

To deliver the potential of OGSA described in the preceding section, many challenges have to be met. Sustaining the effort to achieve widely adopted standards that deliver the convergence of web services and the grid and rallying the resources to build high-quality implementations are obvious international challenges. Here we focus on more technical issues, particularly those where UK expertise may be of benefit, or where solutions are prerequisites for UK e-Science's plans to contribute to OGSA. These should be seen as illustrative examples. Experience through engagement in OGSA engineering will be necessary to refine this list.

1. The types needed for e-Science applications and for database integration need to be defined as XML Schema namespaces. If this is not done different e-Science application groups will develop their own standards and a Babel of types will result.
2. The precision required in G-WSDL definitions will need to be specified so that it is sufficient to support the planned activities. A succession of improved standards should be investigated, so that progressively more of the assembly of grid services can be automated or verified.
3. The context supplied for applications using grid services and the context for a grid service should be defined, they are expected to be similar. Again a succession of approximations may lead to a viable standard without requiring premature design or unrealistic demands on early implementers.
4. The infrastructure for problem diagnosis and response (e.g. detecting, reporting, localising and recovering from partial failures) has to be defined. This will include further definitions of failure notifications and the incremental development of diagnosis and management infrastructure.
5. The infrastructure for monitoring an assembly of grid services. This requires similar development of management and monitoring infrastructure.
6. The infrastructure for accounting within an assembly of grid services.
7. The infrastructure for management and evolution. This would deliver facilities for limiting and controlling the behaviour of grid services and facilities for dynamically replacing grid service instances in an operational system.
8. Coordination services that some grid services can participate in. Grid services capable of being coordinated would offer variants of a coordination port type. Coordination grid services would then use these to deliver various forms of synchronisation, concurrency and transactions.
9. Definitions and compliance testing mechanisms so that it is possible to establish and monitor quality and completeness standards for grid services. Without such gate keeping, time will be wasted attempting to use deficient components and collections of components with inadequate properties. Preferably, this will be offered as a positive endorsement service.

10. Programming models that establish and support good programming practice for this scale of integration.
11. The identification of groups of grid service instances to such functions as load sharing, availability and reliability.
12. Support for grid service instance migration to permit operational, organisational and administrative changes.
13. Support for intermittently connected mobile grid services to enable the use of mobile computing resources by e-Scientists.

These issues already exist as challenges for the “classical” grid architecture. In some cases, GGF working groups are already considering them. OGSA provides a framework in which they may be addressed effectively. For example, interfaces can be defined using G-WSDL to more precisely delimit working groups’ areas of activity.

The full set of relevant technical challenges will be exposed only when the OGSA is tested by use in a variety of applications. Below we recommend a series of experiments, mapping successively more advanced e-Science applications onto OGSA, in order to provoke early refinement of the technical challenges list. A major aspect of this will be working with selected pilot projects to elucidate requirements and priorities (§10).

7 UK’s OGSA Contributions

The UK’s e-Science Core Programme should coordinate middleware development to align with, influence and develop the OGSA. This will inevitably be a dynamic process; i.e. the initial plan will need to be continuously monitored and modified in response to contributions by other countries (primarily the USA) and by major companies. To circumvent the danger that the UK efforts are diminished by frequent changes of course, it is important that the UK e-Science community stake out a claim by declaring its intentions promptly and clearly. If it is to do this convincingly, then it must commit sufficient resources to deliver usable grid services. This, in turn, means that it must focus on a limited set of objectives compatible with available resources. Ineluctably this implies that some of the enthusiasms of some grid researchers will not be as well supported as others.

The UK grid middleware community must work closely with pilot projects to explore the potential of OGSA, to conduct evaluations and to share implementation effort. This requires coordination, which will probably include mutually agreed selection. Some projects will wish to be early adopters of OGSA to gain from the web service aspects as rapidly as possible. It will also need to work with other groups, such as the UK Engineering Task Force and the GGF Grid Web Services WG.

The primary UK focus is to develop data integration. This is a wide-ranging focus that can include many aspects of grid technology and which can develop from basic data management, via database integration to high-level semantically driven integration. It is highly pertinent to both science and commerce, since both rely on integration of growing and diverse collections of data. We also suggest that the UK take a substantial role in developing grid accounting mechanisms and the corresponding grid economies.

To pursue this focus it is necessary to work on a number of sub-themes set out below, and amplified later in this document. These themes may each progress through a number of phases. In addition it will be necessary to address the challenges set out in the previous section where they are not being addressed appropriately for our purposes elsewhere. The themes and supporting work will proceed concurrently, though as some depend on the initial results of others, there will be a staggered start, approximately reflected in the order below. Eventually each strand will have milestones and the integrated programme will have deliverables, but it is too early to present detail. A later section (24) suggests a schedule to start that discussion.

Phase I: Current and immediate actions to position the UK e-Science Core Programme Middleware effort. The activities in this phase can already be well defined. In many cases, there is already substantial engagement and additional effort is already being proposed (e.g. through Grid Core Projects).

1. Understanding, validating and refining OGSA concepts and technical design (Section 8).
2. Establishing the common context, common types and a baseline set of primitive grid services (Section 9).
3. Discussing OGSA with pilot projects and developing technology convergence strategies (Section 10).
4. Defining and prototyping baseline database integration technology (Section 11).
5. Initiating a grid service validation and testing process (Section 12)
6. Establishing baseline logging to underpin accounting functions (Section 13).

Phase II: Advanced development and research. Although some aspects of the following can proceed immediately, many further details will need to be resolved through the work in phase I and through extensive discussions with the OGSA team, GGF working groups and pilot projects. The following are therefore presented in less detail and with the anticipation that activities may be both added and removed as the UK activity develops.

7. Refining G-WSDL, e.g. specifying semantics, and developing tools that exploit it (Section 14).
8. Defining and pioneering higher-level data integration services en route to the semantic grid (Section 15). It is important that this is started as soon as possible as it should have a major influence.
9. Defining and pioneering more advanced database integration technology (Section 16).
10. Defining and pioneering integrated co-scheduling and data management to support integrated database processing (Section 17).
11. Developing advanced forms of Grid Economies (Section 18).

The work in phases I and II must take into account a variety of engineering and design issues that are necessary to achieve affordable, viable, maintainable and trustworthy services. These are introduced in sections 19 to 23, covering respectively: performance engineering, dependable engineering, engineering for change, manageability and operations support and privacy, ethical and legal issues.

Phase I: Goals and Activities

These activities are either already well underway, or they are well defined so that work can commence immediately. They are required now to meet immediate e-Science project needs, to engage in OGSA-related work sufficiently rapidly that we may have an influence and to inform the planning and provide a foundation for the more major activities described in Phase II (§14 to 18). It is important in each of these activities to leverage vendor activities, both by assimilating the emerging proposals that extend web services and by using the best available platforms for experiment and development. Each activity should be targeted at technology intercept, i.e. it should seek to deliver methods and components appropriate to the anticipated industrial and grid supporting environments anticipated by the time it is a product.

8 Understanding, Validating and Refining OGSA

The first phase of this task (8.1)⁵, already underway, is to read, review and discuss documents from the OGSA team, including [1,2]. The review will include evaluation against the architectural requirements set out in Appendix 1.

This should be followed by a series of paper application designs and existing system analyses (8.2) that will develop understanding, establish tutorial case studies and either validate the existing design of OGSA or indicate required refinements. Initial studies will include characterising existing Globus services as grid services, examining how parts of the basic database integration technology (§11) can be defined as grid services and constructed using grid services. Parts of some pilot projects will deliver more extensive case studies (§10). The UK ATF and DBTF will combine forces in some of these studies⁶.

It is expected that by engaging in this study of OGSA as rapidly as possible, in conjunction with other OGSA architects and companies the UK e-Scientists will make substantial contributions. It is important to do this early, to engage actively in the OGSI working group established at GGF4, and to write papers defining these improvements (8.4) backed up by validating prototypes. In order that such validating prototypes can be built, it is necessary to quickly establish a “breadboard” prototype of OGSA operational in the UK (8.3)⁷.

Selected pilot projects should test the OGSA via these early prototypes, since paper exercises often fail to notice critical issues that are easily exposed by real tasks (8.5). The products and understanding that is obtained should be shared as quickly as possible.

As work proceeds in pilot projects and in the other lines of development undertaken by UK teams, further understanding of requirements and potential solutions will emerge. These will be collated by the UK ATF and used to develop the plan for Phase II and beyond. It should be noted that the conceptual model underpinning OGSA admits several forms of partitioning effort. There is a pervasive baseline infrastructure referred to as OGSI. Once that is established, we can of course refine it. At the same time, we and others can explore its mapping down to different hosting environments and platforms. Above the OGSI will be a number of standard grid services, some of which will be equivalent to existing Globus services, such as GIS and MDS, and others will be new, e.g. accounting (§13 & 18). We can expect to contribute to the definition and implementation of a specific selection of these standard grid services, as well as providing new standard services, such as those for database access and integration (§11).

⁵ These numeric identifiers of phases are used in the Gantt chart in Section 17.

⁶ This work has already started and has resulted in feedback to the OGSA team.

⁷ Steve Tuecke has agreed to pass a copy of his OGSA test bed to the GMT.

9 Establishing Common Infrastructure

There are three parts to this: the common grid services: computational context, the standard set of e-Science and grid service types, and the minimal set of grid service primitives. These are interrelated and will benefit from closely coordinated concurrent development. They will build heavily on the services already in use in Globus and in e-Science applications, and it is again essential that effort is continuously invested in coordinating with the rest of the OGSA community.

For developers who are building new grid services or applications (in OGSA every application is also a grid service), they require to know precisely which operations are always supported by a hosting environment (§4.2). As code portability can only pertain within a single hosting environment, these operations may be specific to that hosting environment. That is the operations for developers working within a J2EE hosting environment will not be the same as that for developers using plain Java, or developers using C. The set of functions available for developers may therefore vary between these hosting environments. However, there will be a pervasive baseline functionality, which will have various syntactic forms. The physiology paper [FKNT02] describes it thus:

However, as the above discussion highlights, successful implementation of Grid services can be facilitated by specifying baseline characteristics that all hosting environments must possess, defining the “internal” interface from the service implementation to the global Grid environment. These characteristics would then be rendered into different implementation technologies (e.g., J2EE or shared libraries).

A detailed discussion of hosting environment characteristics is beyond the scope of this article. However, we can expect a hosting environment to address mapping of Grid-wide names (i.e., Grid service handles) into implementation-specific entities (C pointers, Java object references, etc.); dispatch of Grid invocations and notification events into implementation-specific actions (events, procedure calls); protocol processing and the formatting of data for network transmission; lifetime management of Grid service instances; and inter-service authentication.

Whilst traditional grid implementations have mapped such requirements directly to the native operating system or to library code, grid services are likely to be significantly influenced by the richer platforms used in web service hosting.

9.1 Grid Services Hosting Environment

The computational context in which a grid service computes should be defined for several reasons:

1. It establishes what the developer of the grid services can assume (§4.2).
2. It provides a foundation and guarantees needed for management, control and evolution (§6 & 19-23).
3. It enables grid services to be written so that they are portable and re-usable (§4.2).
4. It is a necessary delimiter of state for error recovery and persistence.

We are familiar with the power of well-defined computational contexts, e.g. that defined by the Java Virtual Machine [LY96] and that defined for Enterprise Java Beans [EJB]. Designing such a context for grid services (or e-Science) requires a judicious balance between the following forces:

- Parsimony of facilities so that existing components can be transferred with minimal perturbation.
- Complete functionality that ensures all of the standard web service, grid service properties and the infrastructure identified as necessary in Section 6 are fully supported.

The appropriate way to address this design task is through a rapid cycle of putative designs evaluated through prototypes (9.1.1, 9.1.2, ...) until a consensus is reached that can be supported as a part of the OGSA standard. The UK may wish to take a lead on this for

specific hosting environments that are heavily used by projects or that are considered appropriate platforms for improving developer efficiency in e-Science and e-Commerce.

A common issue that must be addressed is the adoption of a standard representation of a computation's history, sometime called its *context*. This context contains information that must be passed from one service to the next as invocations occur. Examples are the subject on behalf of whom the computation is being conducted (needed for authorisation) the transaction identifier if this computation is part of a transaction, etc. Operations for updating this context should be defined so that its interpretation is unambiguous. An example of such a context is `MessageContext` in Apache Axis.

9.2 Standard Types

The SOAP definition [Mit01] defines a set of types including primitive types, and the recursive composition of these as structures and arrays, based on name spaces and notations of the XML Schema definitions [W3Ca, W3Cb]. The applications and libraries of e-Science use many standard types, such as complex numbers, diagonalised matrices, triangular matrices, etc. There will be a significant gain if widely used types are defined and named early, so that the same e-Science oriented name spaces can be used in many applications and in many libraries, components and services. The advantages include:

1. Simplification of interworking between components that adopt these standards.
2. Better amortisation of the cost of type design.
3. Early validation of the use of WSDL for these aspects of e-Science.
4. Simplification of the task of providing efficient mappings for serialising and deserialising these structures by avoiding multiple versions.

It is likely that we would benefit by addressing some widely used standard types initially, such as those required for MPI [MPI], and those used by the most popular numerical packages (9.2.1). At the same time, those needed for the database integration work in which the UK will specialise should be explored (9.2.2). These would include standard types for transmitting a generic query, and for returning generic result sets or error reports. They would extend to more specific types, such as those needed to return a relational table, those needed to deliver relational updates, and those needed to transmit extracted slices of a multi-dimensional array. The exact choices will depend to some extent on the UK pilot project requirements (§10) and on the database integration requirements (§11 & §15) but we should also aim to have the UK undertaking formative work in this area to influence the international standards. There are, for example, already proposals under discussion at W3C on complex number representations [Compl1, Compl2] and arrays [Array] and proposals in SOAP regarding sparse matrices.

Many communities, such as astronomers, protein crystallographers, bioinformaticians, etc., are developing standards for their own domains of communication and for curated data collections. The e-Science core programme can build on and facilitate this process by developing component types that can be re-used, by developing a pool of expertise and by developing good relationships with the relevant standards communities. As higher-level types are standardised and re-used it becomes easier to move activities towards the information and knowledge layers to which the semantic grid aspires. *Therefore, the UK e-Science programme should invest in encouraging, supporting and coordinating such efforts to develop standards.*

Standard types naturally lead to work to develop more efficient serialisation and deserialisation mappings for these identified standard e-Science types (9.2.3).

9.3 Grid Service Primitives

Concomitant with defining the context in which a grid service executes (§9.1), it is also necessary to define an initial set of grid services that are ubiquitously available, so that more complex grid services can be built assuming these primitives. There is already a commitment from the OGSA team to develop replacements for existing Globus services, such as GRAM, GIS and MDS, as grid services (§4.6 & 4.7). Design issues here include:

1. Carrying forward what has been learnt about application requirements in grid computing.
2. Balancing a simplicity and parsimony that makes comprehension, provision and composition easy with integration that delivers efficiencies.

The programme should engage in prototyping and testing putative foundation sets of Grid Service Primitives (GSPs) (9.3.1, 9.3.2, ...). By way of illustration the following GSPs *might* form a suitable foundation. Some of these are a matter of wrapping existing Globus functions and are already being implemented at ANL.

1. Authenticated connection service.
2. Name resolution service that supports global naming of any item that may be used by a grid service.
3. Data transfer service (with various properties, such as, *reliably exactly once, at most once*, etc.).
4. Synchronisation and time-stamping service, based on an efficient mechanism, such as [LCB99]. OGSI has already chosen a specific solution based on universal time.
5. Secure authentication service for services and resources. Much of this requirement is already met by GSI, which is carried forward from Globus 2 to GT-2.
6. Secure user and role information service, which can be used as a foundation for authorisation mechanisms.
7. Capability based computational and storage resource discovery service.
8. Resource allocation and scheduling services.

The preceding list is illustrative. Some of these components are presumed in OGSI. The exact list must be based on analysis of requirements, on experiments and on dialogue with other parties developing OGSA and its applications. This will be best conducted as a requirements study (§10). It should also take maximum advantage of the APIs being defined in WSDL by the commercial developers of web services (§2.2).

10 Pilot Projects — Education and Consultation

Section 2.2 identified the wealth of innovation underway in the commercial community, the advantages of building on commercially supported platforms, and the risk that many e-Science projects would not take advantage of that opportunity. This section therefore advises the UK e-Science Core Programme to engage in three strands of interaction with e-Scientists engaged in pilot projects:

1. *Education*: using courses, tutorial examples and case studies.
2. *Requirements capture*: using visits and systematic collection and collation (as undertaken by the DBTF) elucidate the immediate and medium-term requirements and priorities of those intending to use web services and grid services.
3. *Consultation and Advice*: It is likely that many projects will wish to plan a technology interception path. Other projects will wish to obtain and deploy established

commercial tool sets and platforms. They should be provided with good quality advice and help in such matters.

These three activities should be integrated, so that the time of those involved is efficiently used. The *education* should comprise three activities:

1. *Visits* to pilot projects at which an introductory overview is presented. On the same visits, the education and technical requirements are solicited and recorded.
2. *Development of sample solutions and tutorial examples*. Based on the actual tasks identified from pilot projects, a team would develop sample solutions showing how a toolkit, technology or method can be used to address the task. These would be hosted at the NeSC & GSC web sites for free download, they would also be used in courses.
3. *Courses* would be provided to stimulate the community and to meet the identified educational requirements. They would bring in experts as well as develop local resources. They would also provide a focus to share good practice and to build supportive communities.

The educational requirement is continuous. The rate at which commerce and the grid communities develop technology and the steady discovery by e-Scientists of new problems, requires a continuous programme. It is straightforward to combine requirements capture and consultation with education, provided sufficient resources are allocated to the total task.

It is likely that there will be significant variety in the response of pilot projects to new technology, including grid services. Some may grasp it with enthusiasm. We may expect to work very closely with these to avoid duplication of effort. Others may wish to avoid disrupting their momentum, and this should be respected. There may be many who take an intermediate position and wish to adopt aspects of a new technology for parts of their work. It is therefore necessary to support incremental adoption. To achieve this, the technology must be capable of working in composition with other integration and e-Science technologies.

There should be meetings to initiate this process soon after GGF4 (10.1). There should also be a series of workshops initiating the educational process.

Representatives of the ATF should visit the regional centres and pilot projects. The distilled OGSA requirements (10.2) and the technology interception plans (10.3) should be completed by July 2002 (GGF5). Initial requirements should be compared with the expected properties of GT-3 and with the capabilities of emerging web service technology. *The strategy for development should be guided by this analysis as well as by the plans of other major grid middleware development groups.*

11 Baseline Database Integration

This is primarily the responsibility of the UK Database Task Force [PAD+02]. Outline information is presented here as *Database Access and Integration* (DAI) is a major component of the UK plans.

A BoF intended to establish a GGF WG on Databases and the Grid (DAG WG) was held at GGF4 (11.1). Four components were presented:

1. A charter for a working group on databases and the grid [DBTF02].
2. A paper outlining the relationship between databases and the grid [Wat02].
3. A paper summarising our initial understanding of DB requirements [Pea02].
4. A paper setting out an initial draft of the UK e-Science database integration road map [PAD+02].

This proposal was well received and it is clear that while the UK could easily play a lead role in this area, there will be plenty of international collaboration and users.

A *basic database interconnection architecture for Globus 2* will be designed and implemented (11.2), for three reasons:

1. To deliver an early component to pilot projects.
2. To gain experience and validate the design and implementation strategy (it will use a web services plus GSI model).
3. To provide a platform for more advanced forms of DB integration (§15 & 16).

It would be basic, in that it would allow a Java application to use an independent database via Globus 2 in terms of the model supported by that database. (This is part of a multi-site plus multi-company GCP proposal, which should be consulted for details [NIE+02].)

The path forward for database connection technology is a *family of grid data services* [Atk02, PAD+02]⁸. These will be designed to have a consistent set of operations and management functions, so that they compose well. They will include various forms of data transport between grid data services, processing elements, data management elements, database connection elements and data storage elements. The goal is a set of building blocks well suited to support the advanced database integration (§16) and higher-level models, such as the semantic grid (§15). These will bring into one framework, data management facilities such as replication (e.g. wrapping Giggle [FIR+02]), file storage accessed via application metadata, and database access. At the same time, the OGSA model will be thoroughly tested. It is very likely that new requirements on its common infrastructure will be identified and solutions prototyped in collaboration with the teams at ANL and ISI.

A *content discovery service* is needed to support content-based discovery of data collections (analogous to capability-based discovery of services). This requires the development of metadata standards for describing the contents, availability and access policies of data collections. It also requires a registry that progressively supports more sophisticated forms of searching and matching. This data discovery service will need to become an integral part of the generic discovery services for the grid. This leads to a continuation of this work (§15 & 16).

12 Validation and Testing of Grid Services and Platforms

It is essential that grid services, grid infrastructure and grid-service hosting environments are developed to meet their specifications. Progressive refinement of these specifications and care to ensure they are mutually consistent should simplify the construction of applications. Experience has shown that if there is no investment in compliance testing, components fail to deliver the required composability. *It is therefore recommended that a compliance testing service be initiated.*

This will develop harnesses and test loads that progressively measure more aspects of behaviour. As higher-level grid service families are introduced (that comply with additional rules to enable rapid and partially automated integration (§14 & 15)) they would warrant specialised testing.

Two outcomes of this would be:

1. Infrastructure, resources and expertise to support compliance testing, which would also assist developers in debugging new services, and
2. Improved quality of grid services which should improve developer efficiency and user experience.

It would form a basis for developing other forms of capability and performance testing. This is necessary if trustworthy e-Science applications are to be achieved (§19 to 23). We may

⁸ An architecture similar to Apache Axis is being considered as a mechanism for composing and orchestrating processing steps, with a structure similar to MessageContext to organise the process.

expect, in the long-term, third party accreditation of components to be a prerequisite for use in mission critical, secure and trusted applications.

13 Baseline Logging Infrastructure

The UK community of e-Scientists uses a large variety of facilities, such as compute resources, storage resources, high-performance networks, and curated data collections. Our goal is to establish a culture in UK science where the e-Science techniques, which depend on these resources, are widely used.

To support the resource sharing within virtual organisations and between real organisations it is essential that resource usage be recorded. The units that need to be recorded will differ between services but could include:

- Number of bytes crossing various boundaries, such as a gateway on a network, the main memory of a computer, the fibres from disk, etc.
- Number of byte seconds of storage use for various stores.
- Number of CPU-hours.
- Number of records examined in a database.

There are clearly many potential units that could be the basis of charging, but for a logging infrastructure to be scalable and usable a relative small number must be agreed and understood. These units may be accumulated, or the critical recorded unit may be based on peak rates delivered. If resource owners are to contribute their resources, these units also have to approximately reflect the origins of their costs within their organisation.

The OGSA needs to intercept service usage as part of its core architecture and record this information through a logging service. This service provides mechanisms to distribute this data to other organisations. It is an area that has been overlooked in the current Globus toolkit and needs to be addressed for the widespread deployment of Grid infrastructures.

It is unlikely that a single charging model or a single basis for charging will emerge in a diverse community. There therefore needs to be a mechanism where different charging policies and different clients can meet. We refer to it as a *market* (§18).

Phase II: Goals and Activities

The goals and activities that follow are more advanced and therefore require additional preparation and planning. Though some individuals may already be engaging in these areas, it is almost certain that coordinated UK action will benefit from the additional understanding and experience gained from phase I activities. Therefore, the activities in Phase II will typically start in the second half of 2002. In every case, there are immediate goals that we can be confident of obtaining with reasonable effort. These will provide useful grid services for both e-Science and e-Commerce. But there are also many worthwhile longer term goals that can be achieved with sustained development and input from research. It is important, therefore, to use these activities to build strong UK teams that expect to continue the work in the longer term, thereby taking advantage of the knowledge and skills built in phases I and II.

14 Refining and Exploiting G-WSDL

There are already aspirations in the web service community to develop WSDL to denote more of the behaviour of a service (§2.2). In addition properties such as performance, reliability, problem diagnosis, cost, etc. are of considerable interest in e-Science. The development of standards for describing such properties and the development of techniques and tools that support or exploit those standards will be discussed in this section. This is an example of the kind of work it is particularly appropriate for the UK to sponsor for two reasons:

1. There are considerable strengths in the UK computing science community: semantics, language design, and the application of formal reasoning and models.
2. It is the kind of work, which is unlikely to be undertaken by industry (§2.2), and so it will be complementary to their efforts and should be applicable in web as well as grid services.

To address these issues it will be necessary to extend WSDL/WSC⁹ and other emerging description languages (§2.2) (using their extension facilities), so that they can specify additional properties. We call the resulting language G-WSDL, although this names an extensible and evolving language rather than fixed one. An incremental approach is appropriate where particular aspects of a language receive attention, developing a precise semantics and techniques for reasoning about compositions of this semantics. *It is of paramount importance not to diverge from or needlessly duplicate the work being invested in web description languages.* Apart from being wasteful, that would also increase the difficulty of deploying the results in industry.

Automated inference can support the composition of services. The goal is to enable e-scientists to specify their required application and for service discovery mechanisms and automated inference to construct dynamically an application meeting this specification. Early versions would be limited to helping with detailed composition over carefully specified composable families of grid services in restricted domains. An example of what can be achieved by current techniques is the Amphion system [LPPU94], which synthesises software for space probe, flyby missions from a library of Fortran sub-routines.

A challenge is to provide a friendly user interface for capturing grid service and application specifications. An initial focus on assisting with problem diagnosis and repair would circumvent some of these issues, as the existing composition can be analysed.

General solutions to the feature interaction problems intrinsic in predicting the quality of service behaviour of compositions of incompletely described grid services are intractable. However, partial solutions are known to be obtainable and applicable in similar contexts, such as telecommunications [CM00]. Since a long-term general goal must be to enable trustworthy e-Science systems [Rod+02] the current e-Science programme should pave the way by

⁹ A proposal to W3C based on a reconciliation of Microsoft's Xlang and IBM's WSFL.

incorporating exploration of the applicability of current inference and applied formal methods research to e-Science applications.

15 Higher-level Data Integration

This section proposes the development of a path towards semantic grids [DRJS01] that will fit within the general programme. It is important to engage in this work early, as it will have a profound influence on requirements and capabilities. Initially it will build on the basic database facility (§11). It is also highly appropriate as a UK activity for the same reasons as those given in section 14, namely:

1. There are considerable strengths in the UK computing science community: ontology, schema and type systems, agents and services, planning and automated synthesis, archiving systems, and evolutionary systems.
2. It is the kind of work, which is unlikely to be undertaken by industry (§2.2), and so it will be complementary to their efforts and should be applicable in web as well as grid services.

The deluge of data emerging in e-Science demands substantial advances in our methods of organising its use (§2.1). If the specification of all data handling remains at its current level, based on physical constructs such as files and on direct manipulation of the detailed data, it is unlikely that the full breadth of science will benefit. Low-level operations are both too labour intensive and too error prone. As the volume of data grows, and the diversity of sources that may be composed expands, then these deleterious effects increase.

Similarly, composition of computational components, particularly software is an extremely powerful mechanism for combining understanding and methods. A grid service, program, workflow, script, library, schema, or portal, is in each case an encapsulation of developed understanding as to how some process is best accomplished or represented. More sophisticated and advanced tasks are achieved by composing such components. Again, if we continue to operate entirely at the current level of discourse, progress will be inhibited for the same reasons as before.

The goal of this branch of e-Science development is to progressively raise the conceptual level of discourse. Practical achievements are accessible through improved metadata descriptions of data and software, incrementally moving the discourse towards the information layer [ABS99]. Much has been done in developing improved understanding of how to support the capture and input of metadata, how to sustain its quality via validation and consistency checks, and how to integrate data from multiple sources (§16). Incremental progress should be obtained in areas such as data curation, data integration, provenance tracking [BKT00, BKT01, BKT02], fully accessible historical archives [BKTT02], ontology management and agent-based discovery of relevant information. Pioneering will often be accelerated by focussing on well-defined domains of e-Science applications, but general methods should be extracted from such focused work.

This is an area in which the UK has many strengths, e.g. the various teams engaged in the Advanced Knowledge Technologies (AKT) IRC (www.aktors.org), and the information and knowledge intensive components of a number of e-Science test beds such as myGrid (<http://mygrid.man.ac.uk/>) and GEODISE (<http://www.geodise.org/>). In particular, a major concern of these projects is to research the problem of providing data with meaning and generate a surrounding semantic context in which data can be meaningfully interpreted. Data equipped with such a context can properly be called information, information applied in an appropriate fashion to effect a decision or achieve a goal can be regarded as knowledge.

It is important to build on and supplement these efforts quickly. This requires that the infrastructure developed as the baseline database integration facilities (section 11) provides an appropriate initial platform [NMF+02]. Strong synergy would be expected between this line of development and the more advanced data integration described in the next section.

It is important to recognise that fundamental research on systems and services is needed to allow us to move from the current data centric view supporting the grid to a set of knowledge services supporting the semantic grid. Some of these need to be tackled sooner rather than later if we are to progress. In particular:

1. Techniques to support the traceability and provenance of content throughout its lifecycle
2. Support for the design, development and deployment of large-scale ontologies for content repositories and the associated annotation of content with these ontologies
3. Support for semantically directed knowledge discovery to complement data mining methods
4. Support for sharing and collaborating across different content repositories at varying scales

16 Advanced Database Integration

The foundations for advanced database integration will be laid in the baseline database access work (§11). It is well understood that there is a pervasive requirement to integrate data from many sources, which are usually geographically distributed, are often organised and stored using different schemas and models, and are invariably managed independently under differing access control and charging policies. Much work has been done by the database industry and by researchers to address this challenge. The multidisciplinary collaborations and the growth in available data (§2.1) increase the demand that this challenge be met. At the same time, it increases the frequency with which it is met and the diversity to be handled, whilst the dynamic nature of virtual organisations and scientific enquiry preclude heavy weight static solutions, such as agreed global schemas.

The challenge may be tackled through four lines of development, as long as they are properly coordinated:

1. Dealing with the geographic distribution using distributed query, distributed transaction and mobile computation techniques.
2. Dealing with data, schema and organisational diversity using schema integration and data translation techniques.
3. Generalising and integrating database constraints and triggers to support both quality assurance mechanisms and active dynamic systems.
4. Negotiating with different charging and access policy regimes using generic authorisation management schemes.

The UK Database Task Force (DBTF) has proposed an approach to this challenge for e-Science [PAD+02, Nle+02]. This focuses initially on developing distributed query [ST00] and transactional processing for science applications. It will depend on the baseline facilities delivered by (§11) and on evaluation, planning and execution strategies already well developed in the database industry. Its demands from OGSA will include access to sufficient data to permit comparison of the execution costs of different evaluation plans (§13). The development may sensibly take advantage of the efforts in modelling and implementing replication for caching and performance [FIR+02] but it may need to extend these concepts to handle subsets of data. It will also require sympathetic scheduling (§17).

Schema integration will draw heavily on the high-level models and ontologies developed under (§15). Research into semi-structured data access [BDH+95, SGP+99, LAW99, ACM97] has shown the value of both semi-automated integration based on formal foundations and on the use of partial integration to support a particular activity. Judicious selection from this general, and rapidly advancing area of database research can yield

workable tools that will benefit scientists. Once again an incremental approach with early products available for field trials will allow refinement of priorities and functions.

Constraints and triggers are used extensively in databases [CCW00] to initiate computation in response to assertions about data or changes to data being violated. This is often used to assist with the task of maintaining data quality, but it is also used to code logic that deals with the consequences of some condition without expensive polling and repeated querying. As databases and other services are integrated via the grid, it is useful to extend such mechanisms to cover a whole distributed system. For example, without them, a locally indexed cache of protein and gene function data might have to repeatedly poll multiple sources to test whether the collection is up to date. The present OGSA has a limited system for notification (§4.3), focused on informing subscribers of grid service data element changes [TCF+02].

A more general notification system would be required, that delivered notifications from any grid service, to any set of subscribing grid services on any topic. The subscription and notification delivery systems of OGSA can almost certainly be re-used, though the topic description notation requires extension. A standard transmission system would need to be developed. A general purpose, scalable distributed event handling system is known to pose research problems. The UK is well placed to address these as it has several strong groups in industry and academia working in an active international context [MB98, BHM+02, Tibco, Graticule, RK, OMG00, CRW01].

The challenge of coping with various administrative regimes depends on developing reliable and flexible authorisation. The present and proposed versions of Globus have effective mechanisms for authentication and we expect further generalisation and improvement of these mechanisms. This delivers a reliable identity that can, in principle be mapped to the identities expected by the various authorities. However, data access is often based on a wide variety of attributes such as: organisation, employer, current role, etc., many of which are time varying. This requires adequate dependable sources of that information and mechanisms, such as X509 attribute certificates, for transmitting it to the database gate-keeping functions. As ability to pay, may often be a criterion, this requires integration with the economic model support (§18).

The support for integration will need to incorporate several management functions, such as fault determination (§20), performance monitoring (§19), charging (§13) and system evolution (§21).

Scientific data has properties that make it amenable to *specialised* indexing, and its increasing scale makes effective indexing ever more important [SKP+00, HAW01, CSF+01]. It is not possible to mandate effort on *specialised* index development *per se*, as it depends on established index design skills and deep knowledge of the data and workloads. However, where indexing development is underway it should be considered in the context of database integration. The stability and scale of scientific data make it worthwhile storing indexes over the data in multiple collections. These indexes involve all of the issues of advanced integration, both during their construction and during their use.

17 Scheduling for Workflow Optimisation

Most applied e-Science jobs involve the exploitation of distributed resources (computational, data, etc.) to meet the expressed needs of the user but within any constraints specified by the service provider. The jobs are generally comprised of multiple tasks, which may be executed sequentially or concurrently. A Web Services Choreography Language (WSC), or a derived markup language, will define the dependency between these tasks. The requirements of such a language and the scheduling infrastructure necessary to exploit it, is an area where the UK can make a contribution. This work is closely related to any activity utilising computational economics (§18) to guide resource selection.

One common job within the currently emerging Data Grids is the processing of a data set. There is a significant difference in cost of data extraction if the most selective filtering or the most compressing summarisation operations can be shipped close to the data. However, when data is owned or hosted by an organisation that does not offer compute cycles then the extraction may involve a bulk copy operation. The movement of large data sets across shared networks is a lengthy and unpredictable operation that needs to be scheduled in advance. Effective scheduling within such a Grid environment must include the ability to reserve resources and specify the quality of service provided by those resources.

Within a Data Grid, a data set may be permanently replicated at a number of sites [FIR02], and this needs to be known to the scheduler through the Grid information system. Data sets may also be cached in other locations as a result of previous scheduling decisions providing temporary data replication. As data operations are rarely isolated, through judicious caching and its exploitation by the scheduler, the costs of data movement, the costs of building indexes and the costs of partial evaluations can be shared over many queries. The scheduler, through the batching of queries, can often enhance this effect. Conversely both data and queries can be partitioned to take advantage of opportunities for parallel evaluation, and it is common to stream stages of evaluations across a tree of processors.

By expressing the work flow and interaction between tasks, and by having the mechanisms to reserve resources and define quality of service levels, a scheduler is able to optimise task distribution across distributed computational and data services through advanced reservation and co-allocation. Factors such as these suggest that close cooperation between the development of high-level and integrated data access (§15 & 16) and scheduler development will be worthwhile. At the very least the cost models used by current execution and scheduling systems will need extension.

18 Computation Economies

This develops further the mechanisms for sharing and accounting initiated as *Baseline Logging Infrastructure* (§13). The complex funding mechanisms that prevail in Europe are stimulating work in this area.

To encourage sustained use of e-Science computational and grid services between organisations and the provision of these resources by independent ‘service providers’ there have to be mechanisms for service charging. The creation of a sustainable long-term economic model that will attract independent service providers requires:

1. A logging infrastructure that records resource usage (§13).
2. A commonly understood means of describing charges.
3. A mechanism to negotiate charges between the consumer and service provider.
4. A secure payment mechanism.

The UK has both the necessity and the expertise in all of these areas to advance the development of such a model in collaboration with international partners.

Charging policies may be static for a particular mode of service invocation (off-peak, background with potential pre-emption, priority, etc.) or dynamic where service cost is determined through negotiation between the consumer and provider (e.g. English auction, Dutch auction, etc.) Users and their computational agents have to understand the charging model. Hence a common description mechanism must be agreed.

Phases I and II: Trustable Services

Throughout the design and implementation of the grid service middleware introduced in Phase I and Phase II, it is necessary to consider and incrementally achieve various goals that are prerequisites of trustworthy scalable and maintainable distributed computation. These are introduced below. *A generic change in the computing science underpinning these issues is required.* Previously, the analyses, models and algorithms have dealt with *closed* systems. There is now a pervasive requirement, dramatically exposed by e-Science, to deal with *open* systems. This applies to each of the issues introduced below (§19 to 23).

19 Performance Engineering

Very large volumes of data from a multiplicity of sites combined with sophisticated data mining, analyses and simulations generate a combinatorial explosion of opportunities for execution performance pathologies. Rapidly deployable and easily adapted monitoring systems, both for regulation and for diagnosis will be essential. However, they will not be sufficient. The complexity of the envisaged systems suggests that typical developers and all users will struggle to understand the causes of costs and performance problems.

A skill base and tool set should be developed to address these issues, building on the contemporaneous work on computation rich performance engineering, to add those dimensions triggered by high data volumes and by complex data sources.

Frequently, performance from individual grid service components will be achieved by mapping their functions onto established and well-developed middleware, such as database systems, hosting environments and operating systems. Ultimately, direct provision by those middleware of grid service interfaces will be needed to achieve adequate performance, but we must expect to prove and refine the functional requirements first. Inter-working between components that have been and adapted and those that haven't will be necessary to allow incremental adoption.

The data-intensive workloads envisaged in e-Science will pose new workload patterns for which existing middleware is probably maladapted. In this case, re-engineering will be necessary, but again, the requirement has first to be demonstrated. The basis for such demonstration must be collected. An appropriate strategy is to establish widespread monitoring of e-Science workloads, performance and usage. The collected data should be in a curated public annotated repository, so that engineers and computing scientists can use it as a resource to establish such cases and to steer designs.

The emerging set of collections of observations of multiple grid systems, running a wide range of applications and with different behavioural aspects observed is comparable with an astronomical data collection, such as that used by SDSS. It has similar properties. Only a subset of workloads and grids are observed. The techniques for observing will evolve and will always be varied. There are benefits from sharing such collections of annotated measurements of grid behaviour. New analyses are stimulated, which in turn stimulate new recognition of phenomena and engineering responses to them.

A necessary component of trustworthy computation is predictable performance. If the task of computing the probable advance of a bush fire over the next 60 minutes takes 90 minutes to compute, the answer does not help preserve fire crew's lives. Conversely, if a scheduler (§17) takes an overly pessimistic view and requisitions excessive resources for this task, the Sydney fire service may receive a crippling bill, and other vital work may be interrupted.

20 Dependable Engineering

As humans are unable to verify the envisaged computations by inspection great care must be taken to ensure that the computations are sufficiently trustworthy that decisions based on their

output can be made with reasonable safety. There are many uncertain factors in the sources of data, the processes handling the data, the software and the platforms supporting the computation. No panacea can be expected but incrementally monitoring systems, assertion checking systems and validation systems must be assembled to progress towards dependability. Database practice makes extensive use of assertions and triggers to detect anomalies [CCW00], software engineering has similar extensive practice [Hoare99], and the OGSi proposes a notification infrastructure, which could raise alarms regarding anomalies detected by the infrastructure. The challenge of bringing these together into a coherent framework is worthy of attention, and essential for data intensive e-Science.

Dependability is often achieved through replication and redundancy [FIR+02]. Where software is concerned, independent implementations may sometimes be used to increase confidence. However, as implementations call on other services, they may use common infrastructure that contains an error. Traceability of computations, closely linked with work on provenance (§15), may be a step towards this issue.

As the research agenda identifies [Rod+02], the challenge is to dependability in the context of large open systems in which uncertainty is prevalent. There is uncertainty about the behaviour of each component and about the infrastructure that connects them. The emerging architectures must ultimately address this issue, as we face a future where every commercial system is directly or indirectly coupled to all others. As that is not a distant future, and as web services will accelerate its arrival, the mechanisms to achieve stable rather than chaotic behaviour should be explored. E-Science provides a context where the risks of such exploration should be considered acceptable. An example of such an engineering experiment is the Farsite project [Farsite]. Systems research that may be relevant includes [ZCD02, DC, YHSD, KE-S+, Parr].

21 Engineering for Change

A fundamental issue faced by all computing systems is how to accommodate change throughout the lifetime of a system and often while delivering a continuous service. Many data centred sciences handle streams of data, e.g. from satellites, from ground sensors, from mobile personal health care sensors, from very long running analyses and simulations. The challenge is to sustain these workloads while dynamic component replacement is supported at all levels. This has to be integrated into the design of platforms, OGSi, hosting environments, grid services and applications. Work is underway at present for fabric management [XX] for the European Data Grid and we have experience of dynamic replacement mechanisms [ED99, EDA01]. The challenge is to develop an integrated and accessible system with reasonable safety and performance characteristics that meets both developers' needs and operational objectives. As always, an incremental approach is the only viable strategy.

22 Manageability and Operations Support

The systems we strive to build will be ever more extensive and ever more complex. The driving force of Moore's law combined with our ambition to build systems to support almost every branch of human enquiry and activity means that there will be many more systems to manage and operate.

To make it possible to manage even a single system as its scale and complexity increases, substantial investment in technology to support management is needed. This will be built on monitoring and logging systems, but the complexity of that data means that tools will be needed to derive information, spot anomalies and visualise operations.

To manage a growing number of systems, increasing investment in higher-level management technology is required, that progressively automates detailed operations. The long-term goal is autonomic systems that carry out their detailed management functions completely automatically [KHC98]. Such an ambition is defined in IBM's project Eliza [XX].

23 Privacy, Ethics and Legal Issues

The interchange of large volumes of data, and the development of sophisticated and accessible data mining and analysis techniques raises issues regarding privacy, ethics and legality. Although the techniques of security implementation are an essential foundation [SAML, And01] they remains a deep question as to how they should be used [FIPR].

The use of data concerning people immediately raises concern about privacy. For example, many e-Science applications are concerned with medical research and improved health care. Here the issues are well known and there is work underway to develop standards [XX]. Grid-based computation may encourage epidemiological and trait analyses, and there are clear advantages to combining data for this across international boundaries. A grid should make that easy. A challenge is to develop and implement support for international agreements that would apply in this case.

Commercially sensitive data, such as IPR that may be exploitable, poses another set of privacy and legal issues. These may differ between disciplines, in different contexts and certainly between countries. International business consortia and multinational corporations already function so there is a legal and technical understanding. However, it may be challenging to consider this in the context of the grid, particularly when data integration or meta-scheduling uses resources that are not known to the subject initiating the investigation.

The principles of science seek openness. The technology that is built must support protection of resources and information. It will be necessary to develop clear ideas about how these interact, how people's rights and efforts are protected and how this can be expressed as policies so that automated discovery and integration doesn't violate local, national or international policy.

Summary and Supporting Information

24 A Programme of OGSA Developments

This will develop a rough overview of how the components of the work are scheduled and their interdependencies. It will be developed after TAG has provided initial feedback.

25 Contact Information

Prof. Malcolm Atkinson
National e-Science Centre
e-Science Institute
15 South College Street
Edinburgh EH8 9AA
United Kingdom
Email: mpa@nesc.ac.uk

Dr Andrew Herbert
Microsoft Research
7 J J Thomson Avenue
Cambridge, CB3 0FB
United Kingdom
Email: aherbert@microsoft.com

Prof. Jon Crowcroft
University of Cambridge
Computer Laboratory
William Gates Building
J J Thomson Avenue
Cambridge CB3 0FD
United Kingdom
Email: Jon.Crowcroft@cl.cam.ac.uk

Prof. Ian Leslie
University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
Email: Ian.Leslie@cl.cam.ac.uk

Prof. David De Roure
Department of Electronics and Computer
Science
University of Southampton
Highfield, Southampton
SO17 1BJ
United Kingdom
Email: dder@ecs.soton.ac.uk

Dr. Ken Moody
University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
Email: Ken.Moody@cl.cam.ac.uk

Vijay Dialani
Department of Electronics and Computer
Science
University of Southampton
Highfield, Southampton
SO17 1BJ
United Kingdom
Email: vkd00r@ecs.soton.ac.uk

Dr. Steven Newhouse
Research Director
Imperial College Parallel Computing Centre
180 Queen's Gate
South Kensington
LONDON SW7 2BZ
Email: s.newhouse@doc.ic.ac.uk

Dr. Tony Storey
IBM UK Laboratories
Hursley Park
WINCHESTER
SO21 2JN
Email: tony_storey@uk.ibm.com

Glossary

| | |
|--------|---|
| ANL | Argonne National Laboratory |
| API | Application Programming Interface |
| ATF | UK e-Science Architecture Task Force |
| B2B | Business-to-business |
| C2B | Customer-to-business |
| DBTF | UK e-Science Database Task Force |
| DCE | Distributed Computing Environment |
| EAI | Enterprise Application Integration |
| EBI | European Biological Institute, Hinxton, Cambridge |
| EDG | European Data Grid Project |
| EJB | Enterprise Java Bean |
| GS | Grid Service |
| GSDE | Grid service data elements, units of data about the state of a grid service |
| GSDL | Grid Service Description Language (used in Globus Team schema examples) |
| GSH | Grid Service Handle, a reference to a grid service instance that is unique for all time |
| G-WSDL | Grid-oriented extensions to WSDL |
| GSI | Grid Security Infrastructure, the interface and mechanisms for maintaining security |
| GMT | Grid Middleware Team @ NeSC |
| HTTP | Hypertext Transport Protocol, provides the basis for web transmissions |
| ISI | Information Sciences Institute, University of Southern California, Marina del Rey, Los Angeles, CA |
| J2EE | Java 2 Enterprise Edition |
| OGSA | Open Grid Services Architecture |
| OGSI | Open Grid Services Infrastructure, a pervasive layer in the OGSA that spans platforms and hosts, providing a uniform virtual distributed computation platform |
| OSF | Open Systems Foundation |
| P2P | Peer-to-peer |
| RPC | Remote Procedure Call, a synchronous mechanism enabling a program in one process to call a procedure in some other process, often running on a different computer |
| SDSS | Sloan Digital Sky Survey |
| SOAP | Simple Object Access Protocol |
| W3C | The World Wide Web Consortium, standards body encouraging development of the world wide web |
| WS | Web Service |
| WSC | Web Services Container |
| WSDL | Web Services Definition Language |
| WSEL | Web Services Endpoint Language |
| WSFL | Web Services Flow Language |
| WSIF | Web Services Invocation Facility [Muk01] |
| WSIL | Web Services Inspection Language [Bri01] |
| XML | Extensible Mark-up Language |
| XMLNS | XML Name Space |
| XSD | XML Schema Definition |

Bibliography

- ABS99 Abiteboul, S., Buneman, P. and Suciu, *Data on the Web: from Relations to Semistructured data and XML*, Morgan Kaufmann, 1999.
- ACM97 Abiteboul, S., Cluet, S. and Milo, T., Correspondence and Translation for Heterogeneous Data, in Proceedings of the 6th International Conference on Database Theory, 1997.
- ADHP00 Atkinson, M.P., Dmitriev, M., Hamilton, C. and Printezis, T., Scalable and Recoverable Implementation of Object Evolution for the PJama Platform, in *Proc. of the Ninth International Workshop on Persistent Object Systems*, 255-268.
- AJ00 Atkinson, M.P. and Jordan, M.J., *A Review of the Rationale and Architectures of PJama: a Durable, Flexible, Evolvable and Scalable Orthogonally Persistent Programming Platform*, Technical Report TR-2000-90, Sun Microsystems Laboratories, CA, USA, 103 pages.
- And01 Anderson, R.J., *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, January 2001, ISBN: 0471389226
- Atk00 Atkinson, M.P., Persistence and Java — a Balancing Act, in *Proceedings of the ECOOP Symposium on Objects and Databases*, LNCS number 1944 (invited paper), 1-32.
- Atk02 Atkinson, M.P., The Open Grid Services Architecture and a Model for Database Access and Integration Services on the Grid, contribution to UK DBTF discussions, 26 January 2002.
- Array <http://www.w3.org/2001/03/XMLSchema/TypeLibrary-nn-array.xsd>
- Axis02 Apache axis project, xml.apache.org/axis/index.html.
- BDH+95 Buneman, P., Davidson, S., Hart, K., Overton, C. and Wong L., A Data Transformation System for Biological Data Sources, in *Proceedings of the Twenty-first International Conference on Very Large Databases*, 1995.
- BHM+02 Bacon, J., Hombrecher, A., Ma, C., Moody, K. and Pietzuch, P., *Building Event Services on Standard Middleware*, work in progress, 2002.
- BKT00 Buneman, P., Khanna, S. and Tan, W-C., Data Provenance: Some Basic Issues, in Foundations of Software Technology and Theoretical Computer Science, 2000.
- BKT01 Buneman, P., Khanna, S. and Tan, W-C., Why and Where – A Characterization of Data Provenance, in Proceedings of the International Conference on Database Theory, 2001.
- BKT02 Buneman, P., Khanna, S. and Tan, W-C., On Propagation of Deletions and Annotations Through Views, to appear in the Proceedings of the Conference on the Principles of Database Systems, May 2002.
- BKTT02 Buneman, P., Khanna, S., Tajima, K. and Tan, W-C., *Archiving Scientific Data*, to appear in Proceedings of ACM SIGMOD 2002.
- Box+00 Box, D. *et al.*, *Simple Object Access Protocol (SOAP) 1.1*, W3C Note 8 May 2000, www.w3.org/TR/SOAP.
- Bri01 Brittenham, P. An Overview of the Web Services Inspection Language. 2001, <http://www.ibm.com/developerworks/webservices/library/ws-wslover>.
- CCMW01 Christensen, E., Curbera, F., Meredith, G. and Weerawarana, S., *Web Services Description Language (WSDL) 1.1*, W3C Note 15 March 2001, www.w3.org/TR/wsdl.
- CCW00 Ceri, S., Cochrane, R. J. and Widom, J., Practical Applications of Constraints and Triggers: Successes and Lingering Issues, in *Proceedings of the 27th VLDB conference*, Cairo Egypt, Sept. 2000.
- CM00 Editors M. Calder and E. Magill, *Feature Interactions in Telecommunications and Software Systems VI*, IOS Press 2000. ISBN 1 58603 065 5.
- Comp11 <http://www.w3.org/2001/03/XMLSchema/TypeLibrary-nn-math.xsd>
- Comp2 http://www.posc.org/ebiz/pefxml/bdoc/dt_complex.html
- CRW01 Carzaniga A., Rosenblum D. S. and Wolf A. L., Design and evaluation of a wide-area event notification service, *ACM Transactions on Computer Systems*, Volume 19, no. 3, pp. 332-383, 2001.
- CSF+01 Cooper, B.F., Sample, N., Franklin, M.J., Hjaltason, G.R. and Shadmon, M., A fast index for semistructured data, in *Proc. of the 27th International Conf. on Very Large Databases*, Morgan Kaufmann, 2001.
- DC Doyle and Carlson, *Power laws, Highly Optimized Tolerance and generalized source coding*, PRL.
- DBTF02 The UK Database Task Force, *Proposed Databases and the Grid, working group charter*, <http://www.cs.man.ac.uk/grid-db/>.

- DRJS01 De Roure, D., Jennings, N. and Shadbolt, N., *Research Agenda for the Semantic Grid: a Future e-Science Infrastructure*, Report for the UK e-Science Core Programme Directorate, UK eSC TR: 2002-1, December 2001, www.semanticgrid.org.
- ED99 Huw Evans and Peter Dickman Zones, Contracts and Absorbing Change: An Approach to Software Evolution, in *Proceedings of the Conference on Object-Oriented Programming, Systems, Languages and Applications* (OOPSLA '99), Denver, Colorado, USA, 1999.
- EDA01 Huw Evans, Peter Dickman and Malcolm Atkinson The Grumps Architecture: Run-time Evolution in a Large Scale Distributed System, in *Workshop on Engineering Complex Object-Oriented Solutions for Evolution, OOPSLA ECOOSE Workshop*, Tampa, Florida, USA, October 2001.
- EJB **EJB context ref to be filled in.**
- Farsite <http://research.microsoft.com/sn/Farsite/publications.htm>
- FIPR The Foundation for Information Policy Research, <http://www.fipr.org/>
- FIR+02 Foster, I., Iamnitchi, A., Ripeanu, M., Chervenak, A., Deelman, E., Kesselman, C., Hoschek, W., Kunszt, P., Stockinger, H., Stockinger, K. and Tierney, B., Giggie: A Framework for Constructing Scalable Replica Location Services, submitted to HPDC11, 2002.
- FKNT02 Foster, I., Kesselman, C., Nick, J. and Tuecke, S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, to be presented at GGF4, Feb. 2002. <http://www.globus.org/research/papers.html>
- FKT01 Foster, I., Kesselman, C. and Tuecke, S., The Anatomy of the Grid: Enabling Virtual Organisations, *Intl. J. Supercomputer Applications*, 15(3), 2001.
- Graticule <http://www.graticule.com/products/MapGPS.html>
- Gray01 Keynote at Super-Computing Conference, Denver 2001.
- GSB+02 Graham, S., Simeonov, S., Boubez, T., Davis, D., Daniels, G., Nakamura, Y. and Nayama, R., *Building Web Services with Java: Making Sense of XML, SOAP, WSDL and UDDI*, Sams Publishing, Indianapolis, Indiana, USA, 2002.
- HAW01 Hunt, E., Atkinson, M.P. and Irving, R.W., A database index to large biological sequences, in *Proc. of the 27th International Conf. on Very Large Databases*, 139-148. Morgan Kaufmann, 2001.
- Hoare99 Hoare, A., *Assertions a Personal Perspective*, Talk on the use of assertions in Microsoft software development, TOOLS workshop, Zurich, 1999.
- ISO-10746 *Reference Model for Open Distributed Processing, Part 2: Foundations and Part 3 Architecture*, http://www.dstc.edu.au/Research/Projects/ODP/ref_model.html.
- JA00 Jordan, M.J. and Atkinson, M.P., *Orthogonal Persistence for the Java Platform: Specification and Rationale*, Technical Report TR-2000-94, Sun Microsystems Laboratories, CA, USA, 46 pages.
- KE-S+ Kurata, El-Samad, Yi, Khammash, and Doyle, *Feedback regulation of the heat shock response in E. Coli*.
- KHC98 Isidor Kouvelas, Vicky Hardman and Jon Crowcroft, "Network Adaptive Continuous-Media Applications Through Self Organised Transcoding," in *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, (Cambridge, England), pp. 241–255, Jul. 1998.
- LAW99 Lahiri, T., Abiteboul, S., and Widom, J., Ozone: Integrating Structured and Semistructured Data, in *Proceedings of the Workshop on Databases and Programming Languages*, 1999.
- LCB99 Liebig, C., Cilia, M. and Buchmann, A.P., Event Composition in Time-dependent Distributed Systems, In *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems*, Edinburgh, Scotland, IEEE, 1999.
- LPPU94 Lowry, M., Philpot, A., Pressburger, T. and Underwood, I., Amphion: Automatic Programming for Scientific Subroutine Libraries, in *Proc. 8th Intl. Symp. on Methodologies for Intelligent Systems*, 1994.
- LY96 Linholm, T. and Yellin, F., *The Java Virtual Machine*, Addison and Wesley, 1996.
- MB98 Ma, C. and Bacon, J., COBEA: A CORBA-Based Event Architecture, in *Proceedings of 4th Usenix Conference on Object-Oriented Technologies and Systems*, 1998.
- Mit01 Mitra, N., SOAP Version 1.2 Part 0: Primer, W3C Working Draft, 17 December 2001, www.w3.org/TR/2001/WD-soap12-part0-20011217/, supersedes Box+00.
- MPI **MPI reference to be filled in.**
- Muk01 Mukhi, N. Web Service Invocation Sans SOAP. 2001, <http://www.ibm.com/developerworks/library/ws-wsif.html>.
- Nie+02 NeSC, IBM, eSNWC, NEReSC, Oracle, *OGSA – Data Access and Integration Services*, OGSA-GCP-PR-V0.11.doc, NeSC 13 South College Street, Edinburgh, February, 2002.

- NMF+02 Newhouse, S., Mayer, A., Furmento, N., McGough, S., Stanton, J. and Darlington, J., *Laying the Foundations for the Semantic Grid*, London e-Science Centre, December 2001, [www-icpc.doc.ic.ac.uk/components/](http://www.icpc.doc.ic.ac.uk/components/).
- OGSA02 Grid Services in Action: A Prototype of the GT3 Core – An Early Globus Toolkit Implementation of the Open Grid Services Architecture, <http://www.globus.org/ogsa>
- OMG00 Object Management Group, Notification Service Specification, June 2000, <ftp://ftp.omg.org/pub/docs/formal/00-06-20.pdf>
- PAD+02 Paton, N.W., Atkinson, M.P., Dialani, V., Pearson, D., Storey, T. and Watson, P., *Database Access and Integration Services on the Grid*, UK DBTF working paper, January 2002 (presented at GGF4). <http://www.cs.man.ac.uk/grid-db/>
- Parr Parrilo, P., *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimisation*, PhD Thesis, Control and Dynamical Systems, Caltech.
- Pea02 Pearson, D., Data Requirements for The Grid: Scoping Study Report, UK DBTF working paper, February 2002 (presented at GGF4). <http://www.cs.man.ac.uk/grid-db/>.
- Put00 Putman, J.R., *Architecting with RM-ODP 1/e*, Prentice Hall, 2000.
- RFH+01 Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S., A Scalable Content-Addressable Network, ACM SIGCOMM, 2001.
- RK Rifkin, A. and Khare, R., A Survey of Event Systems, <http://www.cs.caltech.edu/~adam/isen/event-systems.html>
- Rod+02 Rodden, T. et al., *Realising a Broad Vision of e-Science – e-Science Research Agenda*, prepared for the UK e-Science Steering Committee, February 2002.
- SAML OASIS, *Security Assertion Markup Language*, <http://www.oasis-open.org/committees/security/>.
- SGP+99 Stevens, R., Goble, C., Paton, N.W., Bechhofer, S., Ng, G., Baker, P. and Brass, A., Complex Query Formulation Over Diverse Information Sources in TAMBIS, in *The Journal of Database Management*, 1999.
- SKP+00 Szalay, A.S., Kuntz, P., Thacker, A., Gray, J., Slutz, D. and Brunner, R.J., *Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey*, Technical Report MR-TR-99-30, Microsoft Research, Revised 2000.
- ST00 Sahuguet, A. and Tannen, V., ubQL: a language for programming distributed query systems. UPenn Tech Report, 2000, <http://db.cis.upenn.edu/DL/ubql2.pdf>.
- TCF+02 Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C. and Nick, J., *Grid Service Specification*, to be presented at GGF4, Feb. 2002. <http://www.globus.org/research/papers.html>
- Tibco TIBCO, Reuters system for events, <http://www.tibco.com>
- Vent01 Keynote at Super-Computing Conference, Denver, 2001.
- Wat02 Watson, P., *Databases and the Grid*, version 3, Technical Report CS-TR-755, Newcastle University, Department of Computer Science, January 2002. <http://www.cs.man.ac.uk/grid-db/>.
- W3Ca W3C Working Draft, *XML Schema Part 1: Structures*, work in progress.
- W3Cb W3C Working Draft, *XML Schema Part 2: Datatypes*, work in progress.
- YHSD Yi, Huang, Simon and Doyle, *Robust perfect adaptation in bacterial chemotaxis through integral feedback control*, PNAS.
- ZCD02 Zhou, Carlson, and Doyle, *Mutation, specialization, and hypersensitivity in highly optimised tolerance*, PNAS 2002 99: 2049-2054; published online before print February 12 2002, 10.1073/pnas.261714399

Appendix 1: Requirements for an Architecture

This appendix summarises the conclusions of discussions at the ATF, particularly at the meeting in December 2001, at the e-Science Institute. It is still in draft form and requires further development.

A1.1 Why a Programming Model?

Standards for distributed computing typically have two aspects: interoperability and portability.

Interoperability is achieved through protocol standards. At a minimum these are typically defined with respect to rules for sending signals, a set of abstract state machines and ways of encoding those signals on the interconnection media. In the networking community (IETF, OSI) etc., this has long been regarded as sufficient; to go any further would be seen as imposing unnecessary implementation constraints.

Many aspects of distributed system are hard to model in terms of signalling alone. For example, aspects of system management, security, availability often reference operating system state – for example, transaction protocols cannot easily be separated from the behaviour of transaction managers and local resource management. Consequently standards for distributed computing make architectural statements about system components as well as defining protocols.

It is often convenient for developers to have these system architectural statements represented by standards for programmatic interfaces (often termed “application programming interfaces” or APIs) so that software can be moved from one vendor’s implementation to another – in other words making the programming interfaces portability standards.

Portability standards for distributed computing cannot be defined separately from interoperability standards. The protocols used for interoperability set constraints on what can be communicated and the supported modes of communication (e.g., synchronous versus asynchronous, client/server, publish/subscribe, delivery guarantees and so forth). These naturally have an impact on the functions that need to be provided in the programmatic interfaces to create communication endpoints, interact successfully and handle failures.

A1.2 What is a Programming Model?

A Programming Model can exist at several different levels of abstraction. At the highest level it may be a purely conceptual model of system components and their interfaces. The classic example here is the ISO Reference Model for Open Distributed Processing [ISO-10746] which provides a set of basic concepts for describing distributed systems components and a simple prescriptive framework for assembling components into a consistent infrastructure. It works very much in terms of types of protocols, system components and interfaces, rather than requiring specific examples.

An example at a lower level is the X/Open Transaction Architecture (XA). This defines how to organize a distributed transaction processing system by describing abstract transaction monitors and resource managers and the interactions between them. In essence it defines the functionality that must be provided by specific implementations of those components by a database, transactional file system or message queue for example.

More concrete examples are given by the Java 2 Enterprise Edition environment and by Microsoft DCOM. Both defines an infrastructure for distributed application in terms of components and interfaces defined using specific programming languages and specific protocols included in the runtime environment (Java and the Java Runtime in the first case, C, C++ and Microsoft Windows in the second).

A1.3 Why a programming Model for the Grid?

The Grid is about providing distributed computation services for e-Science. Grid computing requires that distributed computations be set up in advance, torn down when complete and manage complex tasks spread over multiple nodes while in progress. At a minimum this requires an abstract model of the resources committed to such a computation so that protocols for discovery and management can be defined.

Extending this to a concrete model described in the form of APIs would enable Grid computations to be ported from one platform to another readily, masking the inevitable heterogeneity of the underlying environment.

A1.4 State of the Art in Distributed Computing Programming Models

Current distributed computing models are dominated by the needs of World Wide Web e-Commerce: a browser-based client interacts with a web server, which in turn interacts with backend databases and message queues mediated by an “application server”. (Often the application server is integrated with the web server.) The application server supports components written to a specific set of “container” APIs provided by the application server to deal with instantiation, security, transaction control, session and long term state management, load balancing and similar infrastructure functions.

Mention should also be made of distributed computing models for “Enterprise Application Integration” (EAI) of which CORBA and Microsoft DCOM are examples. Both pre-date e-Commerce and had their origins in corporate client/server applications. Increasingly they are used to connect e-Commerce shop fronts to back office systems and line of business applications such as inventory management.

Interestingly the development of these distributed computing environments has had a profound impact of the state of the art in programming language technology including for example:

- Blurring of design/compile/runtime boundaries – development environments allow components to be developed visually, bound dynamically to user interfaces, tested incrementally and deployed in an integrated and seamless way
- Automatic code generation – for example producing runtime “stubs” automatically from component interfaces (either statically when the component is required, or increasingly on-demand when one component is bound to another)
- Just-in-time compilation – producing runnable code only when required and generating code optimized for the environment in which it is to execute
- Runtime optimization – using heuristics and typing information to schedule computation, manage memory and other resources for best throughput, interactive response, etc.
- Extensive use of type checking to improve software safety – current research in type systems has moved on from data typing and now addresses such areas as concurrency control, information flow, memory management and protocol sequencing.

A1.5 Programming Models for Web Services

There is a move in the computer industry towards “web services”. This is an extension of the browser-based e-Commerce model. It is driven by two needs: first to allow for a richer, more intelligent client experience and second to allow for business-to-business integration. Fortunately a common architecture can meet both needs. It is the use of XML to format data sent between web server and browser and the introduction of an application oriented RPC system called SOAP.

The use of XML enables clients to easily extract, process and upload structured data. Presentation can be made an entirely local issue at the client allowing data to be presented in a form that best suits the user’s device form factor and be integrated more closely with local applications on the device, SOAP allows for a more interactive form of interface and for more complex dialogues to be supported.

Web services are at an early stage. The current focus is on inter-operability with standards for web services description (Web Services Description Language) and web services naming and discovery. There is at present no industry wide effort to define a portability model for web services – this is seen by vendors as a competitive opportunity for their development environments and associated operating system platforms.

A1.6 .net, J2EE and the GRID

The press enjoy portraying .net versus J2EE as an industry battle between two camps. Be that as it may, there is a difference in technical focus between the two that reflects their ancestry.

J2EE has a strong process model, but a very limited information model. It is focussed on interconnecting business processes and has a strong messaging and coordination flavour. Data is handled as a service connected at the edge to a J2EE application.

.net has a stronger data sharing model but much less of a process and coordination model. Data is central to the model (hence the emphasis on XML), although not pushed to the extent of a distributed shared memory found in some cluster computing models.

This is an overly simple characterisation. Supporters of each can point to evolutions of their technology that will remove the distinction over time.

The Grid has shades of both – information sharing and distributed coordination. This is an opportunity both to bridge between the dominant industry efforts (or to fall down the crack between them)!

A1.7 Introducing a Programming Model

While the technical case for providing a programming model for Grid services is strong, there are important real world constraints that have to be addressed.

The first is inertia. Most computer users learn one programming language / operating system and stick with it for life. There is no reason to believe the e-Science community is any different – indeed it has helped keep an ancient language (FORTRAN) alive for eons. Computer Science research and indeed companies have produced a plethora of new models and languages that have failed to make it out of the laboratory. That said, transitions do happen for example from assembler to C, C to Java; CORBA was a significant success compared to the OSF DCE, which preceded it.

The common factors seem to be that there must be an obvious big win to motivate users to move to the new model and an incremental strategy users can follow to adopt it. (For example, DCE required users to engage with a new operating system, system services and a host of other constraints before they could use the DCE. This was too high a price. CORBA initially offered much less functionality but had a compelling story about wrapping existing software and services that made it more approachable).

A1.8 What makes A Good Programming Model

From the development of RM-ODP, CORBA, J2EE, DCOM and .net it is possible to extract a list of strong design points for a good distributed processing architecture:

- Strongly typed – for software safety – typing allows more errors to be detected during the development, configuration phases of system construction. Types can provide useful information at runtime to enable optimised implementation. Typing and a strong concept of interface encourage modularity and re-use. They reinforce the distinction between a service and its implementation. Types are a key element of abstraction and a way to manage the evolution of a system.
- Anything in the architecture that can be created or manipulated should have name and names should be permitted as parameters to interactions. This allows dynamic structures to be set up and for applications to manage their resources explicitly should they want to do so. If the system has visible elements that can not be named, then there is no way in which an application can respond to a failure of the entity or control its use of it.
- It should be possible to defer binding as late as possible and to evaluate as lazily as possible – these provide huge flexibility to the infrastructure and its implementation. They provide the means to enable evolution of the infrastructure without having to rebuild applications.
- Transparency should be selective: while it is often useful to defer responsibility to the infrastructure there are often times when an application needs to take control of its destiny and manage resources, respond to failures directly. “Transparencies” should be seen as plug-in attributes to the individual interfaces and managed through the binding architecture.
- Everything should be self-describing – what interfaces are offered, what infrastructure assumptions are made. This makes it possible to build generic applications and dynamic systems with high levels of automated management and consistency checking. The ability to achieve ad hoc interoperability is enhanced.
- Good architectures are recursive. It should be possible to wrap any instantiation of the architecture as a component in a higher-level instantiation. This allows applications to scale up and out from cluster to wide area network in a uniform way. When coupled with self description and strong typing, tools can automatically configure appropriate infrastructure to support a growing application. Recursion allows for control – one system can be isolated from another by encapsulating it as a single entity, with reduced visibility at the higher level.
- Regular structure. Elegance, minimality and uniform composition rules are the foundations for consistent design and faster development.

A1.9 Grid Programming Issues

Since Grid computing by definition involves the Internet it has to be tolerant of the latency of wide area communications and the failure properties of networks not under a single point of control. This suggests an **asynchronous communication model**.

Grid computation will necessarily involve function shipping and data shipping where bodies of data are too large to transfer, or applications are tied to dedicated resources. This suggests a uniform model for **mobile components**.

The Grid enables computation in heterogeneous environments belonging to federated organizations. There is no single point of control or ownership. Often components will be used in ways not envisaged by their creators or owners. **Computations must be able to handle faults** – they can't rely on an overnight solution to the large-scale dependability problem that has been a topic of computer science research for the past 25 years.

A1.10 Towards a Grid Component Model

A Grid component is the virtualisation of a user of Grid resources – i.e., an application or a service.

A component should have an interface through which it can see a **reflection** of the supporting infrastructure – the resources it has been given, bindings between those resources and policies governing their use. Reflection is the foundation to allowing a component to take control of its destiny and modify the runtime in an acceptable way to optimise its performance through application-driven scheduling and resource management, give diagnostics, recover from infrastructure failures and so forth.

It should be possible to **introspect** over all of the attributes of a component – contextual information about how it was produced, who owns it, its infrastructure needs, published interfaces, versions, permissions, etc. This allows developers to search for and re-use components readily and to ensure they are fit for the intended purpose.

The Grid component model should support a wide variety of interaction facilities, including:

- Method invocation – asynchronous, one-to-one, one-to-many (multi-cast), many-to-many (publish/subscribe) – these all are important programmatic structures used in distributed computations which can be mapped onto specific protocols
- Handling flows of non-interpreted stuff – streaming media is the canonical example, but architecturally it should be possible for a computation to treat any communications binding as a black box and simply control connectivity without having to manipulate the transmitted data directly
- Explicit representation of bindings between communication end points (this is how we glue together complex interaction models and flows) into managed objects within a distributed computation
- Event recognition mechanisms to allow patterns of communication to be abstracted into higher level signals.

Components assume an infrastructure to support them. This is generally termed as being the **container** for the component. A Grid container is the virtualisation of Grid resource configuration and allocation functions. In terms of current Grid architecture Grid containment is grounded in the Grid Services Domain concept. It is a hosting service for Grid components and can itself be a component (e.g., the GRAM interface for Grid Service Domains).

Following from the previously given list of desirable architectural attributes we can infer that:

- Containers should permit nesting (encapsulation)
- Containers could be distributed (a consequence of general encapsulation)
- Containers are typed by the (parameterised) policies they embody with respect to quality of service, failover, security and so forth.

Containers have to be created, so we introduce **container factories** as a base concept. A container factory is a service that creates specific kinds of containers. The request to instantiate a container will necessarily indicate the range of policy (parameters) the container is required to support. Container factories are found by lookup in **Directories**.

A container may have the property of acting as a “sandbox”, granting privileges to the components it contains and mediating their access to components in other containers.

A container may be an entity that can migrate from one location to another. Mobile agents are an example. In order to do so the container has to be able to detach from the resources at one location and re-establish them at another. Reflection and introspection are important aids to this.

A1.11 Orchestration

Components and containers provide a suitable set of abstractions for making the elements of a distributed Grid computation into managed objects. This is just the start, as the activities of the components have to be coordinated in an organised way (called “orchestration” by some).

Orchestration requires a range of important facilities:

- Event registration, notification, filtering and action triggering
- Checkpoint and roll-back when components fail and restart
- Transaction integrity for multi-step interactions and consistent distributed state management
- Availability through replication, reconciliation of diverged copies of data
- Resource availability and congestion information to permit cooperative fair minded scheduling.

A1.12 Where to Start?

The definition of a general Grid Services programming model should be rooted in a list of named entities that comprise a virtual organisation using Grid computation. Having established the list, relationships between entities can be discussed and rules for detecting and resolving conflicts proposed to enable the modelling of sound Grid architectures.

An initial set includes:

1. Principal, role
2. Container, container properties
3. Resource types, resource instances
4. Service types, service instances
5. Service interfaces (operations, context, service levels).

