Experiences with GRIA – Industrial applications on a Web Services Grid

Mike Surridge and Steve Taylor
IT Innovation Centre, 2 Venture Road, Southampton, SO16 7NP, UK
{ms,sjt}@it-innovation.soton.ac.uk

David De Roure and Ed Zaluska
Electronics and Computer Science, University of Southampton, Southampton, SO17 1BJ, UK
{dder,ejz}@ecs.soton.ac.uk

Abstract

The GRIA project set out to make the Grid usable by industry. The GRIA middleware is based on Web Services, and designed to meet the needs of industry for security and business-to-business (B2B) service procurement and operation. It provides well-defined B2B models for accounting and QoS agreement, and proxy-free delegation to support account management and service federation. The GRIA v3 software is now being used by industry. By taking a business-oriented approach independent of the evolving Open Grid Services Architecture proposals from the Global Grid Forum, GRIA has demonstrated the need for a wider understanding of Virtual Organizations (VOs). Traditional academic VOs are persistent, resourceful and have logically centralized, membership-oriented management structures. In contrast, the GRIA experience has been that business VOs are likely to be project-focused and have distributed, process-oriented management structures.

1. Introduction

Grid computing is fundamentally about bringing a variety of computational resources together to provide new capabilities and is increasingly adopting a service-orientated approach. Ten years ago there was an emphasis on combining the resources of supercomputers with high-speed wide area networking to provide very-large-scale data processing. As Grid computing evolves it continues to focus on bringing resources and services together, but the emphasis has now shifted onto Virtual Organisations (VOs) as defined by Foster [1]:

"The real and specific problem that underlies the Grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations…This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a virtual organization."

The GRIA project set out to make the Grid usable by industry, adopting the VO approach. The GRIA middleware is based on Web Services because currently this provides the most pragmatic support for Grid applications in commerce and industry. GRIA is an acronym for 'Grid Resources for Industrial Applications' and the focus on business objectives has resulted in significant differences from other Grid research in several important areas.

In particular, we have successfully implemented a lightweight Grid infrastructure closely modelled on the trust and business practices typically encountered in commercial (rather than academic) activities. 'Lightweight' in this context refers to the ease of installation and use: only a minimal client footprint is required, together with a Java runtime environment.

This paper reports on our experiences in the GRIA project designing and implementing middleware using Web Services. The background and history of the GRIA project is reported in section 2, followed by a discussion of the architecture in section 3. A comparison and evaluation of GRIA is provided in section 4, followed by future work in section 5 and conclusions in section 6.

Proceedings of the First International Conference on e-Science and Grid Computing (e-Science'05)
0-7695-2448-6/05 $20.00 © 2005 IEEE

2. Background and related work

The GRIA project started in December 2001, with the clear but challenging aim of making the Grid usable for business and industry. This focus distinguished GRIA from the many academic Grid infrastructures then under development in the Grid community. The critical issues for business users were identified as security, service levels and interoperability.

Two of the GRIA partners provided detailed case studies to help determine the requirements and to evaluate the outputs of the GRIA project. In summary these are as follows.

- CESI develops and provides advanced numerical and experimental engineering solutions to support design, operation and environmental impact studies of hydropower plants. They have developed and continue to operate many models and are constantly seeking to improve accuracy. The models are run as-required to predict the structure behaviour. Because the models are being run intermittently on an ad-hoc, on-demand basis, CESI have conflicting needs: they require rapid results but they do not wish to procure high performance machines as these would be idle for most of the time.
- KINO TV and Movie Productions wished to use grid-based 3D rendering applications based on commercial third-party codes as part of an online collaboration between the user and their customer. In contrast to CESI, the Kino staff are animators rather than programmers, and their requirements emphasised ease of use. They requested a simple interface to GRIA via a portal. High-performance computational resources are again required on-demand to reflect customer requirements, and in addition the application had to integrate with the established business process.

GRIA originally proposed to incorporate business models and processes into the Globus GT2 platform [1]. At that time Globus was the most successful Grid middleware available, although still orientated towards the demands of the High Performance Computing community which had initiated the early Grid research. There were however many difficulties being experienced by the early Globus adopters – the software was too difficult to implement, to maintain and to use (e.g. the well-publicised difficulties with Globus access through firewalls).

The first phase of GRIA consisted of a requirements analysis into the business process support that would be necessary, but during this phase the Open Grid Services Architecture (OGSA) was launched [2]. It was immediately apparent that the original objective of using the Globus Toolkit 2 (GT2) was no longer viable as GT3/OGSA would soon replace GT2. It was equally clear that there would be a considerable delay before GT3 became sufficiently mature for use as the basis of an industrial-strength system. Other Grid middleware platforms (such as UNICORE [3]) were available or under development, but there was no consensus as to the long-term viability or suitability of any of these alternatives following the launch of OGSA.

GRIA at this point (2002) abandoned the original GT2 toolkit proposal and instead implemented a lightweight Grid infrastructure using Web Services to support file-compute processing in a commercial business-to-business (B2B) context. The use of Web Services was consistent with industry requirements and with the decision of the Grid community to develop Grid services as an enhancement of the Web Services model.

A pre-release of GRIA demonstrated the feasibility of the Web Services approach by supporting a business tender process modelled on real commercial practice ("invite-tender-contract"). The first full release of GRIA in 2003 introduced accounting of usage and invoicing. The financial model again reflected standard business practice, with a requirement to establish a supplier account in advance of any contractual relationship and invoicing typically at month-end. The tender process also supported quality-of-service negotiation and the additional feature of a 'pre-approved supplier list'.

Evaluation inside the project by end-users identified a wide spectrum of different requirements for the user interface. At one extreme were users that expected a command-line interface, while at the other extreme the users requested an easy-to-use graphical interface. GRIA v1 used a 'wizard'-type interface which could not accommodate these widely different requirements – the application workflow enforced proved to be much too restrictive. To provide maximum flexibility a client-side API was designed to allow user partners to write their own client-side programs for GRIA user applications.

The GRIA v2 release of early 2004 [4] provided this API to support the advanced users, with some limited graphical handlers provided for users who did not wish to use a command-line interface. The API was implemented in Java, and a complementary command-line interface was also supplied so that users could write scripts to enact the GRIA business processes.

The current version of GRIA is GRIA v3, architecturally similar to GRIA v2 but with numerous robustness and implementation improvements together with usability enhancements [5]. In particular GRIA v3 includes an 'Enterprise Client' portal, which is built using the Java API, so that the end user requires only a standard browser for full access to all GRIA functionality. This addressed a key KINO requirement for ease of use. The Open Middleware Infrastructure Institute (OMII) [6] has been set up by the UK e-Science program [7] to provide distributions of reliable, interoperable and open-source Grid middleware. OMII has adopted a Web Services approach [8], reflecting the adoption of Web Services solutions in the e-Science programme, which draws directly on GRIA.

During the course of the GRIA project there has been significant discussion inside the Grid and Web Services communities over the most appropriate standards to adopt to ensure convergence. At present, the WS-Resource Framework (WSRF) proposal [9] provides the best description of the current strategy and this is being implemented in the GT4 toolkit. There is still some debate as to whether WSRF will provide an acceptable solution and it is anticipated that interoperability with non-WSRF Web Services will continue to be necessary.

In parallel with this activity, projects such as DataGrid [10] continued existing development based on a modified Globus GT2 toolkit because there is a time-critical customer requirement for proven systems in 2007 (e.g. for the CERN LHC project [11]). This activity has now been taken over by the EGEE (Enabling Grids for e-Science in Europe) project which is proposing a new Web Services middleware (gLite) [12, 13]. Other related Grid research sponsored by the European Commission is summarised in [14].

3. Description of the GRIA architecture

This analysis is of the GRIA v3 system, which is based on Apache AXIS technology, and uses only Web Services features permitted in the WS-I Basic Profile 1.0 [15] and WS-I Basic Security Profile 1.0 [16]. The middleware has the following five main functions, each of which is described in the following sections:

- service providers and consumers
- security
- resource management and accounting
- data processing
- a client-side API

3.1 Service providers, consumers and process contexts

GRIA distinguishes a service provider from a service consumer. A service provider is a legal entity (e.g. a business) that provides a set of services for access to data storage and processing capabilities. A service consumer is a legal entity that consumes these capabilities, acting through a service client controlled by a representative person (or agent). Most consumers at commercial sites are behind firewalls, so there are no consumer-side services in GRIA.

All interaction between a service consumer and a service provider takes place as part of a well-defined business process. The well-defined business process specifies the individual steps required to enforce client compliance with a specific business workflow. The correct orchestration of each process step requires an understanding of the correct context, maintained in GRIA using a URI to provide a context identifier (or 'context ID').

Service providers assign a context ID to each process before they begin and convey these to clients as part of the previous process. Clients must include the relevant context ID in subsequent messages to the service to indicate which process the message relates to. This mechanism is similar to the everyday use of 'your reference' in business correspondence, which identifies the recipient's context. The GRIA context ID is analogous to a WSRF resource identifier, although because GRIA has been restricted to the WS-I specifications a context ID is handled as a service argument. There are no implicit methods provided to access any data held by the service associated with a process context.

3.2 Security components

GRIA supports the following security components:

- *Secure transport*: GRIA uses HTTPS as a transport protocol to provide message confidentiality, with client as well as service authentication activated.
- *Message authentication*: message signatures are provided by WS-Security headers, allowing messages to be checked for integrity and their senders reliably identified independently of the transport protocol.
- *Authorisation*: GRIA provides a Process-Based Access Control (PBAC) subsystem that services can access to check if a requested action is allowed in the specified process context for the calling user, or to update the policy for individual actions and users in each assigned context.

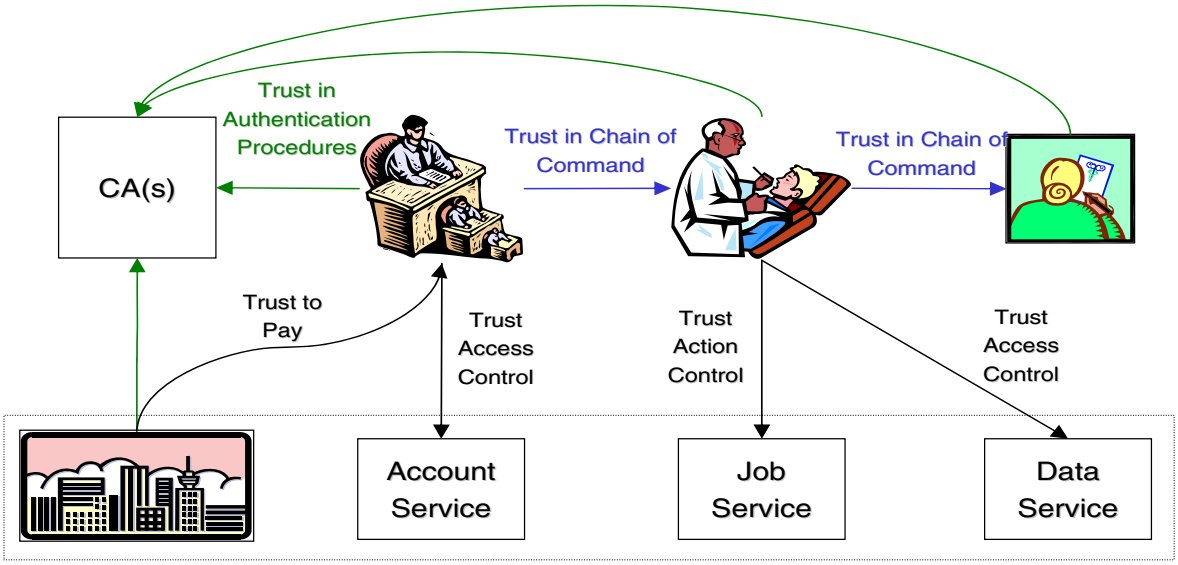

Figure 1. Trust and delegation in GRIA

The use of accepted commercial security mechanisms means that GRIA will operate with conventional HTTPS proxies and standard firewall configurations. GRIA implements security in depth by firstly enforcing mutual authentication for every HTTPS connection. Any attempt by an unauthenticated client to access a GRIA service will be dropped at the SSL handshake level which reduces the impact of any 'denial of service' attack. Requiring every message to be signed provides message-level security. Finally, fine-grained access control can be enforced as required for business purposes by the GRIA PBAC mechanism. Two types of control mechanism are provided, authorisations and restrictions. Authorisations define what is allowed (and by whom), while restrictions specify actions that are disallowed (and by whom).

Fine-grained access to service operations is thus constrained by the PBAC authorisation to enforce the required compliance with the well-defined business processes. Each process has a trusted principal, a consumer representative whose trustworthiness is based on actions in some previous process, or in attributes that have been verified "out-of-band" (e.g. creditworthiness). The principal can establish other actors (consumer representatives, or service providers) as their trusted delegates, allowing them to participate in part or all of the process.

Trust and delegation are therefore core properties of the GRIA architecture, expressed through business processes and client actions (see figure 1), but fully supported architecturally by the security components for authentication and PBAC. In any event, the mutual authentication mechanism provides for full traceability.

3.3 Accounting and resource management services

These services implement support for the business processes as explained below.

The commercial orientation requires that the account service is always the first point of access for a consumer representative. This service supports a top-level business process with a principal known as a budget holder and a context identified by account ID. The account service provides the following functions:

- applying for a credit account and monitoring the status of the application;
- fetching statements of account, allowing the budget holder to monitor usage;
- initiating resource allocation processes (to be charged to an approved account);
- specifying trusted delegates, who will be allowed to initiate resource allocation processes charged to an account;
- closing the account, terminating the associated top-level business process.

Once a new account has been opened, it will remain active for the lifetime of the business relationship (which could continue over a number of years). The financial status of the budget holder and the credit limit established must be supported by an

out-of-band mechanism, such as the provision of credit card details or trade references. The service provider controls the process using an internal management interface provided by the account service, providing for the approval (or rejection) of account applications, the notification of payments from the account holder, the setting (or modification) of credit limits and the recording of charges against the account.

The resource allocation (or allocation) service supports the establishment of service level agreements, whose principal is trusted to set these up under an account by its budget holder. The account service allocates the context ID for a resource allocation process (and any consequent agreement) on initiation of the process. The resource allocation service provides the following functions for the user:

- submitting a service request (for data transfer, storage and processing capacity) and receiving a service offer (which may be less than requested) from the service provider;
- confirming a service offer, thereby establishing a Service Level Agreement (SLA);
- extending an existing SLA (by submitting an extension request and confirming the returned extension offer)
- initiating data storage or job execution processes, drawing on the resources allocated in a service level agreement;
- terminating a resource allocation process and any associated SLA, along with any jobs or data storage processes initiated from it.

The resource allocation service is connected to a capacity model representing the physical network bandwidth, data storage and processing nodes made available to GRIA by the service provider. This model is used to determine the specific level of service to offer in response to a consumer request. The service provides an internal management interface to the service provider, allowing them to set up the capacity model, and register resource consumption by job execution and data storage processes. The unit of work is a standard CPU-second (normalized if necessary by a scaling factor). Client users request an allocation over a given time period and the service provider responds with an offer depending on their spare capacity in that period. Alternative offers are possible as well as user-defined constraints. Once the allocation has been agreed, the client may run jobs until the agreed work allocation has been reached.

The resource management service registers a charge with the service provider account service when an SLA is made or extended, based on the amount of computation and data transfer specified. It may optionally register a credit if the associated resource allocation process is terminated before the allocated resources have been consumed. Thus GRIA can support either usage-based or allocation-based charging models.

3.4 Data storage and processing services

Data storage and processing services support the application-level processes in GRIA, which conform to a file-compute application model orchestrated by consuming clients.

The data service supports data storage and transfer processes. The context ID for such a process refers to a file storage location and implicitly any data stored at that location. This context ID is assigned by the resource allocation service when the process is initiated. The main functions of the data service are:

- upload (write) and download (read) of stored data files;
- transfer of data files to (or from) another data service;
- enabling and disabling read or write access by trusted delegates (other consumer services);
- checking if someone else can read or write the data;
- write-protecting the data, to prevent accidental overwriting (even by those with access rights);
- closure of the data store and deletion of any stored data.

The data service interfaces with a logical data store accessible from the service provider compute cluster. The GRIA partner Dolphin Interconnect AS has developed a fast network file system for cluster systems suitable for a GRIA logical data store.

The processing service provides execution of (pre-installed) applications, as described below:

- submission of a job, including specification of the input and output data storage context IDs;
- monitoring the status of a job;
- killing a job after it has been submitted;
- terminating the job execution process, which also kills the job if still running.

The job service interfaces with the logical data store and with a compute resource or "execution platform" which may be the service host, or a cluster (e.g. PBS or Condor), or some other network-accessible platform. The job service acts as a client to data storage services, fetching input data prior to running a job on the execution platform and sending output data after it has finished.

Job and data services from different sites can be peered to support application workflows, provided the job service is granted the necessary access rights. In this scenario the job service must check that its client (as well as itself) has access to the data, to ensure one user cannot exploit rights assigned to the job service by another user of the same job service.

The interface to the job execution platform is a set of "platform scripts" that can be modified by the service provider to allow any computational resource to be accommodated. These can invoke application scripts on the execution platform to perform application-specific actions such as preparing a workspace, populating it with input data (e.g. by extracting ZIP archives), checking for malicious input, and packaging output for the job service for staging to output data stores.

When the associated job process is terminated, the job service registers the amount of processing time used and data transferred with the service provider resource allocation service. These are propagated through to the account service, where the charge to the client is computed and invoiced.

3.5 Client-side API and portal

On the client side, GRIA provides an object-oriented Java API on top of a basic Web Services invocation framework. The API makes it straightforward to set up data stores and jobs and create applications to orchestrate remote workflows across multiple Grid services. The main features of the GRIA API are:

- a local registry, where the context IDs provided by service providers are stored to keep track of all applications and resource allocations available to the user;
- object representations of contextualised services: accounts, resource allocations, data stores and jobs;
- helper classes to carry out common business processes, e.g. obtaining resource offers from several service providers and selecting between them.

These features enable straightforward programming of business and application processes. Without them, users would have to keep track of process contexts and pass them explicitly to a Web Services framework.

Finally, two client-side user interfaces are available: a command-line client for operating GRIA that end-users can access directly (or from a scripting language such as Perl) and a web portal allowing access via a browser.

4. Comparison and evaluation of GRIA

The GRIA end-user partners have evaluated GRIA v3 and successfully demonstrated operation using their commercial application codes. The feedback from these partners has been vital to development of the software and the success of the project.

The focus on business processes in GRIA provides the key distinguishing feature compared with other Grid middleware platforms, together with the business-orientated approach to trust and authentication. This business emphasis is the key defining feature: commercial business practices have evolved and been refined over many years and are widely understood. By adopting these practices as the starting point for the architectural design, it becomes inherently easier to deploy the GRIA software into existing businesses – both at the technical level and also at the even-more-important cultural and political level.

This can be illustrated by comparing the GRIA approach to the VO model. A traditional Grid (such as Globus) assumes that the VO is persistent, resourceful and managed (such as a large-scale academic research collaboration). The management focus is on maintaining centralised information about resources and VO members together with the definition and monitoring of rules for membership and operations. There may be explicit services operated by the VO for mapping this information onto local authentication and authorisation systems and accounting at VO level for the resources used by each member. This is illustrated in figure 2.

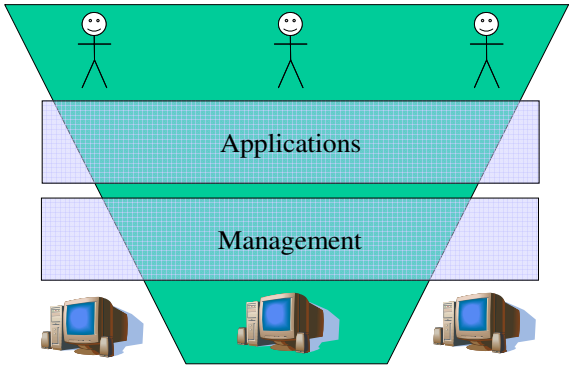

Figure 2. Traditional VO

In contrast, GRIA supports an entirely different style of VO, which encourages transient business-to-business federation for specific tasks. GRIA VOs may form and disband very rapidly, with little or no prior infrastructure and no requirement for a well-defined community. There is no VO control over resources, because these are managed by the

organisations that own them who will have a vested interest in optimising the use made of their assets. The management focus will be on fully-distributed process and trust relationships, rather than membership and role administration.

For example, each GRIA site is free to select which Certification Authority (CA) to trust, rather than being constrained to a single centralised CA provided by the VO management as often required by traditional Grids. The VO used and advocated by GRIA is what we call a “fast VO” (figure 3) as opposed to the traditional monolithic “large VO”.

Another example of the business orientation is the GRIA emphasis on commercial-grade security mechanisms. In particular, GRIA does not permit the use of proxy certificates for delegation, because this forces a user to trust any service holding a proxy-certified private key. Impersonation using proxies has some performance benefits, but the loss of control over dependencies is not acceptable for many commercial users.

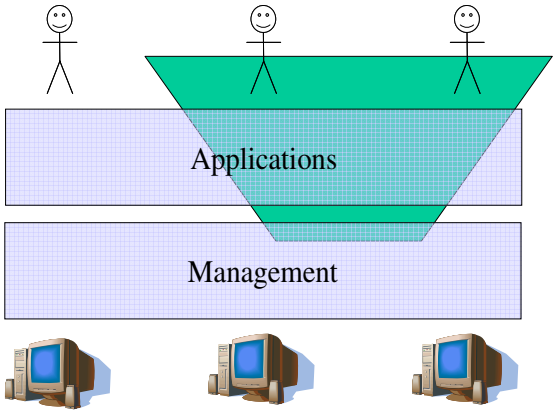

Figure 3. GRIA VO

The main strengths of GRIA are its architectural simplicity, adherence to well-established Web Services standards and its strong support for business processes. Although GRIA v3 is focused on a small number of core services, it is straightforward to build intermediaries, brokers, and other "high level" services.

The design decision to deal with context at the process level makes it easier to add services to handle rendezvous between multiple business processes. For example, a resource allocation service could be created to handle software rights (which may be acquired from another service provider) as well as computational capacity allocations. It is also straightforward to represent and manipulate process context on the client side through a simple API. To handle context entirely at the infrastructure level is likely to be far more complex and this remains a challenge for advocates of WSRF in its current form.

5. Future Work

The GRIA project has identified many challenges to the successful implementation of full interoperability between business entities. In the general case interoperability with other Grid infrastructures is required, but despite the adoption of agreed standards where possible this remains a future research goal. Many other research objectives have been clarified during the project, all of which will require significant further work.

By focusing on evaluation case studies, GRIA has worked with a limited number of services. With broader uptake of GRIA and other Web Services Grids, there will be a need to describe and discover services, and to negotiate service level agreements, in an automated manner to support the dynamic VO creation. As part of its interoperability strand, the GRIA project has tracked the development of ‘Semantic Web Services’ solutions such as OWL-S [17] and the Web Services Modelling Framework [18], which offer solutions to describing services and their composition.

These are emerging solutions and further studies are required to establish best practice in what we might term ‘Semantic Grid Services’. The approach to workflow adopted in GRIA is practical, sitting comfortably with existing business processes, but as workflow enactment solutions improve we expect it will become possible to represent these workflows using languages such as BPEL, which are aimed at the business context.

Even a simple negotiation between two different business processes has proved to be extremely hard to automate. Extending this to complex representations about required quality of service and an acceptable SLA raises even more fundamental questions about how these are best implemented and represented. Assuming that consistent representations can be agreed, it is then necessary to reason about which tender offer should be accepted for a given requirement.

With respect to negotiation, the WS-Agreement activity in the Global Grid Forum addresses part of the solution, supporting offers and agreements but without further scope for more flexible negotiation. Our work suggests that future solutions will draw on the body of work in the software agent community [19], as well as semantic web representations and workflow languages. GRIA use cases have been expressed as an interoperability challenge to the GGF Semantic Grid research group [20].

The GRIA project team plans to continue future development (as part of the EC IST NextGRID and SIMDAT projects) by mapping GRIA processes and

concepts onto emerging standards such as WSRF, WS-Trust and WS-Federation, and exploiting Semantic Grid ideas to facilitate dynamic federation. This work will be extended to address new types of application services (e.g. database access and workflow services), but without sacrificing the flexibility and simplicity of the basic GRIA architecture.

6. Conclusions

Starting with the apparently modest goal of adding business support to an existing Grid system, the GRIA project has produced a new middleware based entirely on Web Services and focused from the beginning on commercial business-to-business applications and business models. It includes off-the-shelf Web Services technologies, extensions for security, and a model of process based access control which underpins the business processes. It has also produced a stimulus for development and standardisation in the area of B2B negotiation and resource brokering methods.

GRIA highlights the need for a broader understanding of different VO models, including fast and agile B2B models as well as the large, persistent VO models more typical of large-scale academic research collaborations. GRIA also highlights the need for the Semantic Grid to support truly open markets and processes. These will be addressed in future work using GRIA middleware in the EC IST projects NextGRID and SIMDAT.

Acknowledgements

GRIA is an IST project (Project Number 33240) funded by the European Commission and the contribution of the GRIA partners is acknowledged. GRIA also uses software components developed in other projects: we would like to thank the EC GEMSS project, the UK e-Science Comb-e-Chem project (GR/R67729/01) and support from the OMII.

References

[1] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *Int. J. Supercomputer Applications*, 15(3), 2001, pp. 200-222.

[2] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", In *Open Grid Service Infrastructure WG, Global Grid Forum*, June 2002.

[3] The UNICORE project. http://www.unicore.org

[4] Steve Taylor, Mike Surridge and Darren Marvin, "Grid Resources for Industrial Applications", In *Proc. 2004 IEEE International Conference on Web Services (ICWS 04)*, San Diego, California, 2004, pp. 402-409.

[5] The GRIA project. http://www.gria.org

[6] The Open Middleware Infrastructure Institute. http://www.omii.ac.uk

[7] T. Hey, "e-Science, e-Business and the Grid", In *Int. Conf. on Computational Science (ICCS2002),* Amsterdam, The Netherlands, 2002.

[8] Atkinson, M, De Roure, D. et al "Web Services Grids : An Evolutionary Approach", Technical Report UkeS-2004-05, UK National e-Science Centre, 2004.

[9] WS-Resource Framework 1.0 reference. http://www.globus.org/wsrf/specs/ws-wsrf.pdf

[10] The DataGrid project. http://www.eu-datagrid.org

[11] A. Unterkircher, S. Jarp and A. Hirstius, "The CERN openlab for DataGrid Applications", http://www.ercim.org/publication/Ercim_News/enw59/unterkircher.html

[12] The EGEE middleware architecture. https://edms.cern.ch/file/476451/1.0/architecture.pdf.

[13] The EGEE gLite middleware external interfaces. https://edms.cern.ch/file/487871/1.0/EGEE-DJRA1.2-487871-v1.0.pdf.

[14] P. Graham, M. Heikkurinen, J. Nabrzyski, et al., "EU Funded Grid Development in Europe", In *AcrossGrids 2004 Conference,* Nicosia, Cyprus, 2004.

[15] WS-I Basic Profile 1.0 reference. http://www.ws-i.org/Profiles/BasicProfile-1.0-2004-04-16.html

[16] WS-I Basic Security Profile 1.0 reference. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2004-05-12.html

[17] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara, "Bringing Semantics to Web Services: The OWL-S Approach," In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, California, USA, 2004.

[18] D. Fensel and C. Bussler, "The web service modeling framework WSMF", Electronic Commerce: Research and Applications, 1(2), 2002. pp. 113–137.

[19] I. Foster, N.R. Jennings and C. Kesselman , "Brain Meets Brawn: Why Grid and Agents Need Each Other", in Proceedings of 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), New York, USA, 2004.

[20] D. De Roure and M. Surridge, "Interoperability Challenges in Grid for Industrial Applications", In *Proc. Semantic Grid Workshop, GGF9*, Chicago, Il., 2003.