Grid Resources for Industrial Applications

Steve Taylor
IT Innovation Centre
2 Venture Road, Chilworth
Southampton, SO16 7NP, UK
+44 23 8076 0834
sjt@it-innovation.soton.ac.uk

Mike Surridge
IT Innovation Centre
2 Venture Road, Chilworth
Southampton, SO16 7NP, UK
+44 23 8076 0834
ms@it-innovation.soton.ac.uk

Darren Marvin
IT Innovation Centre
2 Venture Road, Chilworth
Southampton, SO16 7NP, UK
+44 23 8076 0834
djm@it-innovation.soton.ac.uk

Abstract

We introduce Grid Resources for Industrial Applications (GRIA), a project that aims to enable commercial use of the Grid. GRIA enables service providers to rent out spare CPU cycles, and clients to hire those CPU cycles. Web services play a key role in the architecture of GRIA, chiefly through their interoperability and security features - they provide a well-defined means of limiting clients' access to discrete operations, thus allowing clients to do "work" on a remote application server without giving them full shell access to the server. In this paper, we focus on requirements, business processes and security, and interoperability and standardisation issues raised by this work. We also describe some of the lessons learned through experience of designing, implementing and deploying a prototype system, GRIA v1.

1. Introduction

GRIA is an acronym for Grid Resources for Industrial Applications and is a system that enables:

- service providers to rent out spare CPU cycles, thus attaining better utilization; and
- clients to hire those CPU cycles, thus providing them with access to HPC when they need it.

In short, it aims to permit commercial use of the Grid in a secure, interoperable and flexible manner.

Figure 1 shows the basic premise of GRIA for an example structural analysis problem using the finite element method (FEM). An engineer wants to outsource a simulation because they do not have enough internal resources to get it done in time. The horizontal path is the actual engineering simulation, and the vertical path is the "business" side of the outsourcing – discovery of a service provider, negotiation of quality-of-service (QoS) terms (perhaps including price, runtime, machine spec, software spec), agreement, running the job and settlement. This type of problem is typically very demanding of computational resources, and may not be run frequently - hence it is an ideal candidate for the computation-on-demand nature of GRIA.

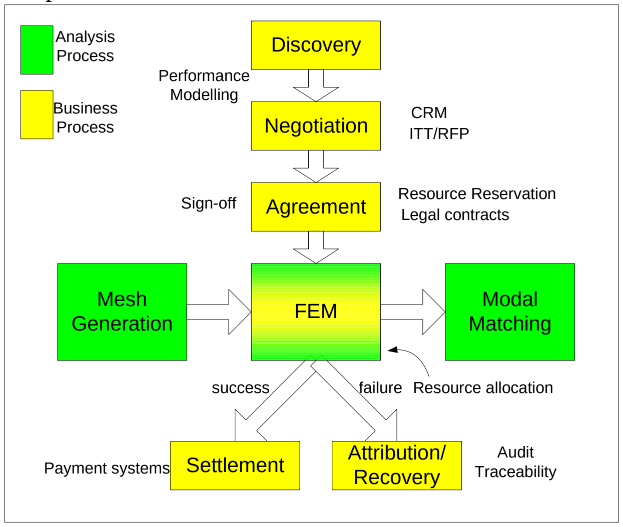

Figure 1: Example use of GRIA

The GRIA software consists of three evolving prototypes: GRIA v0, GRIA v1 and GRIA v2. Each consists of a client and service provider component. GRIA v0 was a basic first attempt at providing a basic business process for Grid job submission. GRIA v1 introduced a more extended business process to handle long-lived jobs and encompass business procurement processes, and is currently deployed for testing in a number of project partner sites across Europe. GRIA v2 is now in development, and with enhanced software architecture and a wider range of business models.

Section 2 of this paper covers the requirements for GRIA, and why these are important. Section 3 covers the business processes we have adopted. Section 4 discusses our approach to security. Section 5 discusses business process enforcement, which is a means of ensuring that the business processes are adhered to. Section 6 covers interoperability, and the importance of a standards-based approach. Finally, Section 7 concludes with a summary of lessons learned and some indications of future work in GRIA v2, and possible contributions to standards and best practice in this field.

2. Requirements

2.1 Business Processes

Our primary purpose in the GRIA project is to provide ways to do business on the Grid. GRIA builds on the work of the DISTAL project [1], which used an agent (rather than specifically Grid) platform to demonstrate the possibility of negotiating for software services. Lessons learned from DISTAL concerning business processes included:

- A need to keep people in the loop. Business decisions are complex, and people don't want them automated.
- A need to use standards. Open systems improve exploitation and take-up.

Business processes in GRIA have been designed to take account of these lessons. We aim to use existing business models and processes as far as possible, and avoid inventing new models for the Grid except where this is essential. Traditional business processes have operated successfully for many (hundreds of) years. By starting from existing processes, we make it easier to integrate Grid models into existing businesses, and reduce the risk that our models and processes will prove unstable in deployed systems. We note that the recently formed GGF Grid Economic Services Architecture Working Group [12] concurs with this philosophy, which is also reflected in the Computational Markets Project Grid Economic Services Draft Proposal to GGF 8 [7], [14] and in the discussion of a computational economy framework for the Grid by Buyya et al [6].

2.2. Security

Traditionally the Grid has been used by academics cooperating in benign environments. As soon as we wish to use the Grid for commerce, we face a completely new set of security requirements. Indeed, we must use commercial grade security to remain credible. The main issues are discussed by one of us in the Rough Guide to Grid Security [22], which was motivated partly by the needs of a commercial grid, and has guided the development of GRIA.

In the commercial Grid world, a service provider's primary concern is to provide computational resources to *traceable* and *creditworthy* users:

- For traceability, mutual authentication of client and service provider identities is paramount.
- All communication must be over secure channels, so that eavesdroppers cannot read or alter the data.
- A service provider must protect itself from breaches such as hacking and denial of service attacks.
- The system must support "security in depth" - employing a succession of barriers that limit the damage should any one barrier be breached.
- The system must not undermine existing security barriers. In particular, we must operate with conventionally configured firewalls, and not rely on tunnelling around them.
- We must base the system on off-the-shelf components that are mature and regularly patched.

2.3. Interoperability

GRIA is intended to provide a means to make money by exploiting a market in grid-based computation, as a consumer or a supplier to this market (or possibly both). To maximise the potential size of this market, we must make the GRIA technology as interoperable as possible.

2.4. Summary

In summary, GRIA had to be:

- conservative in its business models and processes;
- secured to normal commercial standards; and
- open and interoperable.

We now examine how we sought to meet these requirements, and what lessons we believe can be learned from our experiences.

3. Business models and processes

3.1. Negotiation model

The first business process investigated and implemented in GRIA v0 and v1 is a traditional *Invite–Tender–Contract* model. A client invites a number of service providers to tender for a particular piece of work. The service providers respond with tenders that indicate a run time for the work, and how much it will cost the client. The client selects the tender that best suits their needs, and exchanges contracts with the service provider to confirm the terms are accepted by both sides. Once both sides have digitally signed the contract, the job is submitted and run.

Our implementation has the (human) end-user decide which proposal to accept, leaving them in control of the trade-off between performance, cost and their level of trust in the service provider. This is consistent with our experience from the earlier DISTAL project, where users rejected the (apparently more attractive) notion that these decisions could be taken by software agents.

3.2. Quality of service models

The quality of service model used with the above negotiation process is based simply on the response time for a single job. We express quality of service as a set of obligations accepted by each party:

- The client must state the amount of work needed for their job, when they will submit the input data, and when they will collect the output.
- The service provider agrees to allocate a resource capable of delivering the required amount of work between submission and collection time, and to store the output until the collection time.

The noteworthy point about this approach is that we do not make absolute guarantees that resources will be available to run jobs by a certain time. We believe that this is not feasible in a real commercial environment. Instead, the parties state their obligations, and agree what should occur if those obligations are or are not met. This is a robust approach that recognises that guarantees are not absolute, and uses conventional business practices to address this.

To operate our quality of service model, we needed a measure of computational work that is independent of the resource that will be used, but allows the service provider to calculate the run-time and decide whether they can meet their side of the agreement. We used a very simple model in which the execution time for a job on a given machine is simply:

$$T_e = \left(\frac{W}{R} \right)$$

where W is the workload of the client's problem, and R is the relative performance of the target machine. We can measure R for a given machine by running a representative benchmark for which we know W. We can then estimate W for a given job from an estimate (or measurement) of T_e on a machine where R is known, and thus we can estimate run-time on any other machine. This allows the service provider to determine whether any of their machines is acceptable under a given quality of service agreement.

In GRIA v0, the client-side estimate of W was obtained by asking the user for an estimate of run-time on their own machine, which the GRIA client had benchmarked in the background. In GRIA v1, we have a more sophisticated model in which W and R are both vectors, covering different aspects of machine performance (processing, disk I/O, and memory bandwidth), and the W vector is estimated using a neural network developed by NTUA [10], [11], to model our applications (FEM analysis and 3D video rendering).

In both GRIA releases, we use a simple model of resource availability to estimate the queuing time for a non-dedicated compute cluster controlled by Condor or PBS. This model is also used to limit the total amount of work sent to the cluster by GRIA.

3.3. Discovering service providers

In real-life, organisations do not allow staff to engage arbitrary service providers to fulfil their day-to-day needs. This is certainly true for services involving commercially sensitive data. In sectors such as health-care, where personal data is involved, allowing an arbitrary service provider to be used as a processor of this data is actually illegal under the relevant EU Directives [16].

Instead, most organisations maintain an "approved supplier list", of suppliers with which the organisation has negotiated acceptable terms including price, quality and (where appropriate) confidentiality, etc. These suppliers have been vetted and approved, and possibly preferential rates negotiated with the supplier. Anyone within the organisation that needs to purchase a particular item or service should use an approved supplier, as these have been vetted and approved.

We use approved supplier lists in GRIA to record the relationships that a client organisation has with approved service providers. This fits our philosophy of modelling existing B2B processes as far as possible. For users, the approved supplier list acts as an internal registry that they can consult to find potential service providers.

This is not to preclude the use of external (or public) registries. In GRIA, these are a source of classified information about available services from which to populate the approved supplier list.

3.4. Financial processing

In GRIA we have assumed that all financial processing, from the credit checks performed when a client is first encountered, through to the transfer of transactions to the bank for processing, is out of band.

The DISTAL project [1] taught us that humans would much rather be responsible for decisions involving credit checks and credit limits, as an automated process cannot take responsibility. You cannot sue a computer because it gave a dishonest customer a large credit limit!

Clearly, there is no need to invent a new banking system for the Grid (given that we are using fairly conventional business processes). Because of this, GRIA does not provide accounting facilities beyond statements of usage (from which bills may be generated) and logging.

3.5. Lessons learned

The main lessons learned about Grid business processes in GRIA v1 are as follows:

- emulating existing processes where possible is the best approach; but
- the Invite-Tender-Contract model is probably too heavyweight to be used on a "per job" basis.

By emulating existing business processes, we have found it relatively easy to integrate with existing infrastructure, which means one doesn't have to develop Grid alternatives for things like banking services, etc.

However, having the user "sign-off" each individual job, with an exchange of signed QoS contracts, is not a good idea unless each job is very expensive. It may be sensible where a job involves heavy use of a high-end commercial software package, but this is not the case in the GRIA project, where the cost of each job can be as little as €1. In these circumstances, we need lightweight business processes where a client can pre-negotiate quality of service, after which they can just submit and run jobs, the QoS terms having been already taken care of.

We are investigating solutions to this problem for GRIA v2. We are currently focusing on a "resource rental" model for this multiple-job type of QoS agreement. Here, the QoS is agreed over a particular time period, for a particular total workload, and the client (or their delegate) can submit as many jobs as they like within these limits. The cost to the client may be presented as a tariff: cost per unit of workload, for instance.

4. Security Features

4.1. Rings of Security

In accordance with our "security in depth" philosophy, GRIA provides multiple "rings of defence". In most cases, this is achieved simply by adopting solutions from the world of web-based e-Commerce systems, and working within, rather than around the operational security best practices developed there.

GRIA uses mature, off-the-shelf software components as its application container and web server such as the Apache HTTP Server [3]. It is of primary importance that the user-facing applications are mature, well patched, and are kept up to date using a prompt security advisory service [4]. A critical feature of GRIA is that we have not modified or "Grid-enabled" any of the components it is based upon. If vulnerabilities are discovered, GRIA users can use the patches from Apache – they don't have to wait for a GRIA-specific version of the patch to appear.

We advocate that GRIA services and associated resource clusters be placed in a De-Militarised Zone (DMZ). Again, this is conventional best practice for most web-based business systems.

On the client side, we expect that the user's system will be behind a firewall configured to allow outgoing connections only. GRIA, like other recent Grid systems is based on Web Services, using SOAP messages to implement an RPC model of service invocation. Unlike some other Grid systems, GRIA's invocation mechanisms operate over a standard HTTPS transport. GRIA v1 services are designed so that each SOAP invocation is reasonably short-lived, so we don't have problems with TCP connection time-outs, and there are no "call-backs" from the service to the client (e.g. for notification). These features mean that GRIA works with conventional HTTPS proxies and firewall configurations. There is no need for the system administrator to open any non-standard tunnels in the firewall, and client-side users continue to benefit from its protection.

GRIA also uses both transport- and message-level security. At the transport level, GRIA (through Apache) enforces mutual authentication of HTTPS connections. This ensures that an unauthenticated client cannot access a GRIA service, as the HTTPS connection will be dropped in the SSL handshake, before any data reaches the Web Service message processor.

Once the HTTPS connection is established with an authenticated user, message-level security is used to enable separate authentication (and non-repudiation) of message content. GRIA v0 and v1 require that each message is signed in conformance with the W3C SOAP Digital Signature Extensions Note [21]. GRIA v2 will support end-to-end message authentication and possibly encryption using the WS-Security standards [26]. This will enable greater flexibility for intermediary message processors and action delegates.

4.2. Minimal authorisation

GRIA uses a so-called "minimal authorisation" security paradigm. Traditional academic Grids have supported shell access (or something equivalent), whereby the user accessing a remote resource can run essentially any command on the remote machine. This undermines the philosophy of "security in depth", in which one aims to restrict the user's rights to the absolutely [22].

On a commercial Grid, users are much less likely to know each other personally, and may have divergent (or even conflicting) interests. This means that we cannot rely on collaboration or goodwill to regulate behaviour. Under these circumstances we need to practice "minimal authorisation" - limiting user access to specific resources, rather than allowing them arbitrary rights associated with shell access. Web Services provide just this - a means of limiting clients' access to well-defined operations, thus allowing users to do "work" without giving them full shell access to the application server. In an academic Grid, this may be seen as a limitation, but for a commercial Grid, it is an essential security feature.

4.3. X.509-compliant PKI

Based on the requirement for commercial grade security, subscriber identification, privacy, and non-repudiation, GRIA has implemented and operates an open, fully X.509 compliant, Public Key Infrastructure (PKI).

Full compliance with the X.509 standard requires a Certification Policy (CP) defining the requirements of the PKI, and the constraints that must be met by its implementer(s). The GRIA PKI is intended to be open, so the CP allows for multiple Certification Authorities (CAs), provided they operate in accordance with the CP. Each GRIA site can choose which certification authorities to trust, and there is no need for "cross-certification" to support this. We anticipate that in future, GRIA user communities will choose their own CP and CAs.

The GRIA Project's own Certification Policy defines the rules for the GRIA PKI, and follows the recommendations of the IETF RFC for Certificate Policy and Certification Practices Framework [18]. *(We conform to the 2002 version of RFC 2727, the latest at the time of authorship of the CP and the CPS. This is now obsolete and replaced by RFC 3647, listed in the references.)*

The emphasis of the GRIA Project CP is on security, user education and support, rather than scalability or cost. This is acceptable within the project, where the systems are prototypes, users may be unfamiliar with PKI operation, and we need a high level of security and mutual trust. Obviously, we do not expect commercial markets to operate their PKI in the same way, but the GRIA technology is flexible enough to handle this.

Finally, GRIA does not support the use of Grid proxy certificates. The August 2003 Internet Draft concerning proxy certificates [19] defines a proxy certificate as:

- "…a certificate that is derived from, and signed by, a normal X.509 Public Key End Entity Certificate or by another Proxy Certificate for the purpose of providing restricted proxying and delegation within a PKI based authentication system."

Our main concern with proxy certificates is that they give (albeit temporary) credence to a private key that may not be under the associated user's control. For commercial use, where protection of identities and private keys is paramount, this is unacceptable. We therefore cannot use GSI-enabled protocols such as HTTPG [17] or Grid-FTP [15], as end-to-end authentication is needed for all process invocation. GRIA performance suffers somewhat as a result, but given the security drawbacks, we cannot permit the use of proxy certificates in a commercial environment.

4.4. Lessons learned

The main lesson from the GRIA security implementation work is that industrial and commercial users are far happier to accept a Grid that uses existing best practice, even if this means we are unable to exploit all the performance advantages of service-initiated messaging through special firewall tunnels, GSI-enabled protocols, and shell access (or equivalent).

Having said that, we recognise that application developers do not like the restriction to bilateral client-server interactions enforced by GRIA. We do not plan to address this in GRIA v2, but we are investigating alternative ways to support peer-to-peer application topologies without losing control in the UK E-Science project "The Semantic Firewall" [23].

5. Business process enforcement

5.1. Workflow enforcement

Given that we are operating a commercial service, with high security requirements, we must be able to control who can do what, and in which context. For example, we cannot allow the running of a computational job until there has been an agreement that payment will be made for the computation.

In GRIA, *workflow enforcement* provides this link between business process and security. Our approach has been to model all business processes as a hierarchy of bilateral "conversations" between clients and service providers. We then implemented a "conversational authorisation" system to support dynamic *authorisations* containing who can do what, and in which conversation. An operation will be executed for a particular user in a particular context only if an authorisation matching this specification is present. All other situations result in denial. Hence business process workflows are enforced.

Conversational authorisation is grounded in the Web Service interface of a GRIA service. It governs which exposed operations are accessible to a user in a particular conversation. A WSDL interface typically exposes many operations, all of which may normally be executed by a user. Our solution regulates the use of these Web Service operations to enforce the business workflow.

The operations themselves may open and close authorisations – for example, an upload of data may be permitted only once, and must precede the running of a job. The first thing the “UploadData” operation does is check that there is an open authorisation to upload data for the quoted user and conversation. Once the data is successfully uploaded, it closes the authorisation to upload data in the same conversation, and opens authorisations for subsequent operations (for example running a job). Thus, the workflow sequence is enforced.

5.2. Contexts

We have found that hierarchical contexts are extremely useful for representing business processes. At each level, there is a well-defined subject and conversational history that the context refers to. This approach is strongly based on existing business processing mechanisms.

For example, a B2B relationship is likely to involve a client opening an account with a service provider organization. The account is the subject of a conversation at this level, and the account number is the service provider's identifier for that particular relationship – in other words a context identifier. Within the context of this account, orders are raised by the client, and within an order, there may be many items, each of which may be delivered and invoiced separately. Thus there is a hierarchy of contexts, allowing each individual document to be traced back to the top-level account.

To represent this in GRIA, we use three nested levels of conversations, an example of which is shown in Figure 2:

- a conversation referring to the account opened by a Budget Holder client with a service provider;
- a conversation referring to a quality of service agreement between a User client and service provider, which covers a job (or in GRIA v2, jobs) that will be run on an account; and
- conversations referring to individual jobs and data stores, that use resources allocated under a quality of service agreement.

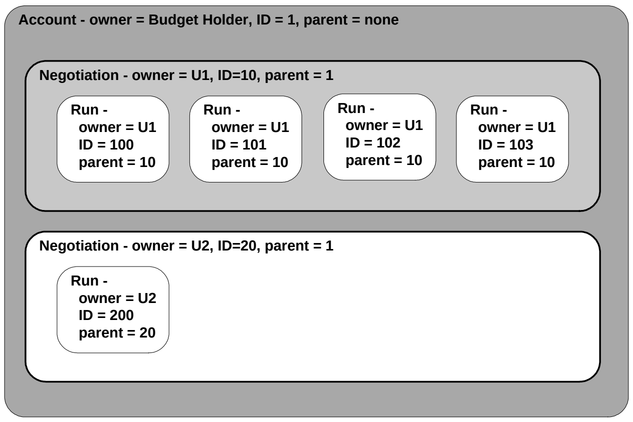
Account - owner = Budget Holder, ID = 1, parent = none
Negotiation - owner = U1, ID=10, parent = 1
Run - owner = U1 ID = 100 parent = 10
Run - owner = U1 ID = 101 parent = 10
Run - owner = U1 ID = 102 parent = 10
Run - owner = U1 ID = 103 parent = 10
Negotiation - owner = U2, ID=20, parent = 1
Run - owner = U2 ID = 200 parent = 20

Figure 2: Account context hierarchy

This forms a tree-like structure representing the relationship between a client and service provider. At any instant, the contextual hierarchy captures the status of the client-server relationship (e.g. outstanding jobs or total amount owed).

To represent contexts, we have found it useful to borrow an established method from traditional business processes – “our-ref”, and “your-ref”. This is a simple but effective means of allowing two communicating parties keep track of a particular subject. The key point is that in each bipolar communication, there are two references, one owned by each side respectively. Each side owns, uses and manages one reference ID (our-ref), and quotes the other (your-ref) back to its owner. Because each side is responsible for their own reference, they can ensure that this reference is unique within their own domain. Thus we do not need a global scope or schema for representing context. Of course, this tried and tested method has worked for many years in paper-based business processes.

When a new conversation is started, the conversational authorisation system supplies the conversation ID for it, and the service provider adopts it as their service provider reference.

5.3. Delegation

Each conversation has an owner who can extend access rights to other parties through delegation. This is the mechanism by which authority for a particular action in a particular context is conferred on a user. GRIA v1 provides a web-based account management portal, operated by the service provider (using certificate-authenticated HTTPS), enabling the top-level conversation owner to access and manipulate delegated access rights on their account.

An example of delegation and its relationship to the hierarchical context model is illustrated in Figure 3.

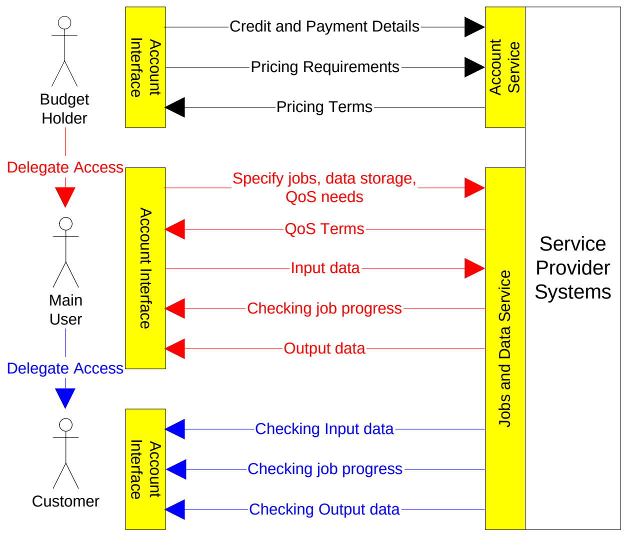

Figure 3: Delegation

The Budget Holder has opened an account with a service provider, and this is the top-level conversation. If the Main User wishes to negotiate QoS with the service provider, they must have permission from the Budget Holder. The account manager allows the Budget Holder to communicate this to the service provider by opening an authorisation on the account for the Main User to access the “OpenQoS” operation. Thus they can delegate responsibility for negotiating QoS on their behalf.

When the Main User opens a QoS negotiation, they quote the account number to the service provider. The service provider is then able to check this against their list of authorisations, and if the main user is permitted, a new conversation is created, whose parent conversation is the account. Given the hierarchical nature of the contexts, when the time comes to send the Budget Holder a bill, it is a simple matter to collect the relevant information as the service provider must only recursively look for is the set of unsettled children of the account.

Now further suppose that the job is run on behalf of the Customer. The Main User may now open an authorisation for a Customer to examine the results of the job, thus delegating access to the Customer. This is done exactly the same way the Budget Holder delegated access to the account for the Main User.

5.4. Lessons learned

The main lessons learned from our work are that it is possible to construct dynamic authorisation mechanisms that relate easily to business processes. In doing this, we have not found it necessary to use role-based access control. The usual reason for using role-based mechanisms is the need to manage access rights for large numbers of users, but in GRIA this becomes a fully distributed responsibility.

Neither have we used centralised authorisation services in GRIA. At present, we cannot see any use for such services. Business process enforcement is a matter for the participants in a process, and few businesses would trust an external agency to handle this, especially a centralised service that could become the focus for criminal attacks.

Thus, a distributed process-oriented authorisation scheme similar to our dynamic conversational model seems inevitable. We can envisage changing some details (e.g. adding support for roles), but see no benefits in a centralised approach.

6. Interoperability issues

GRIA aims to contribute to standards, but not by trying to promote small changes in existing Grid specifications (e.g. by contributing to the Resource Specification Language used in the Globus Resource Allocation Manager (GRAM) [13]). Instead, we are focusing on emerging activities to understand and express standardisation requirements in the Semantic Web and Semantic Grid [20] communities. We believe that this approach will lead to a mature understanding of needs, which can then feed into next-generation standards for things like resources, quality of service and security.

A number of challenges for commercial Grids in industrial applications have been presented at GGF 9 in [9]. The focus for that discussion was the challenges coming from business modelling for the representation of resources, scheduling and quality of service. This is our current focus in GRIA v2, but we see that in the longer term we will need to reflect processes (and especially process enforcement) in security standards, etc. Other semantic-level standards that will be of interest to GRIA are those for describing business processes and function, such as BPEL4WS [5], and DAML-S [8], [2].

We also perceive that the GRIA approach for contextualising business processes is consistent with the ideas for endpoints from the WS-Addressing [24] proposals. It is also consistent with the recent proposals from IBM and Argonne National Laboratory for a Web Services Resource Framework [25]. We believe that it will be both feasible and interesting to evolve GRIA in this direction, and at some future time demonstrate interoperability with a Globus 4 Grid.

Ultimately, we believe that a client organisation should be able to discover a service provider that may suit their needs, and download from that service provider a specification of how the service provider wants to do business. This implies standardisation over the descriptions even of service function.

7. Conclusions

The main conclusions of our work on GRIA are as follows.

Firstly, in an industrial Grid, one must start from business processes that industry is used to, and avoid extending them except where absolutely necessary. This approach makes the Grid more accessible and acceptable to industry, and also makes it easier to integrate the Grid with industry's existing infrastructure and processes.

Secondly, an industrial Grid must be based on existing best practice in security to gain acceptance. Our findings show both this goal is achievable, but that there may be some drawbacks, especially for performance. However, on an industrial/commercial Grid, security must always come first.

Finally, an industrial Grid will depend heavily on semantics to achieve interoperability, especially at the level of business processes and corresponding security mechanisms. We expect that progress in Semantic Web and Semantic Grid standards will provide a platform for the next generation of lower-level standards for resources, quality of service and security.

8. Acknowledgements

GRIA is an IST project (Project Number 33240) funded by the European Commission and coordinated by IT Innovation, University of Southampton, UK. The authors acknowledge the contributions of the other GRIA partners: National Technical University of Athens, Dolphin, CESI and Kino. The authors also acknowledge the contribution of Professor Dave De Roure, University of Southampton.

9. References

[1] Addis, M. J., Allen, P. J. and Surridge, M. (2000) "Negotiating for Software Services". In Tjoa, A. M., Wagner, R. R. and Al-Zobaidie, A., Eds. *Proceedings Eleventh International Workshop on Database and Expert Systems Applications (DEXA 2000)*, pages 1039-1043, London, UK.

[2] Ankolenkar, A., Burstein, M., Hobbs, J. R., Lassila, O., Martin, D. L., McDermott, D., McIlraith, S. A., Narayanan, S., Paolucci, M., Payne, T.R. and Sycara, K., "DAML-S: Web Service Description for the Semantic Web", *The First International Semantic Web Conference (ISWC)*, Sardinia (Italy), June, 2002.

[3] Apache Web Server, http://httpd.apache.org/

[4] Apache Web Server Security Advisories, http://httpd.apache.org/security_report.html

[5] Business Process Execution Language for Web Services (BPEL4WS) Specification, Version 1.1, http://www.ibm.com/developerworks/library/ws-bpel/

[6] Buyya, R., Abramson D., Giddy J., and Stockinger H., "Economic Models for Resource Management and Scheduling in Grid Computing", *Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, Wiley Press, USA, May 2002.

[7] The Computational Markets Project, http://www.lesc.ic.ac.uk/markets/

[8] DAML-S, http://www.daml.org/services/

[9] De Roure, D. and Surridge, M., "Interoperability Challenges in Grid for Industrial Applications", *Semantic Grid Workshop at Global Grid Forum 9*, 5-8 October 2003, Chicago, USA.

[10] Doulamis, N., Doulamis, A., Panagakis, A., Dolkas, K., Varvagirou, T. and Varvarigos, E. "Workload Prediction of Rendering Algorithms in GRID Computing," *European Multigrid Conference*, Oct 7 - 10, 2002, Hohenwart, Germany.

[11] Doulamis, N., Doulamis, A., Panagakis, A., Dolkas, K., Varvarigou, T. and Varvarigos, E., "A Combined Fuzzy - Neural Network Model for Non-Linear Prediction of 3D Rendering Workload in Grid Computing," submitted to IEEE Trans SMC, PART B.

[12] The Global Grid Forum Grid Economic Services Architecture Working Group (GESA-WG), http://www.gridforum.org/3_SRM/gesa.htm

[13] GLOBUS-GRAM RSL Specification, http://www-fp.globus.org/gram/rsl_spec1.html

[14] Grid Economic Services Draft Proposal to Global Grid Forum 8: http://www.lesc.ic.ac.uk/markets/Resources/gesaservicesdraft.pdf

[15] The GridFTP Protocol and Software, http://www.globus.org/datagrid/gridftp.html

[16] Herveg, J.A.M., Crazzolara, F., Middleton, S.E., Marvin, D., Poullet, Y., "GEMSS: Privacy and security for a Medical Grid", HealthGRID 2004, 29th-30th January 2004, Clermont-Ferrand, France.

[17] HTTPG: Globus Toolkit 3 Core – A Grid Service Container Framework. http://www-unix.globus.org/ogsa/docs/alpha/gt3_alpha_core.pdf

[18] Internet Engineering Task Force RFC for Certificate Policy and Certification Practices Framework - RFC 3647: ftp://ftp.rfc-editor.org/in-notes/rfc3647.txt

[19] Public-Key Infrastructure (X.509) (PKIX) Working Group Internet-Draft: Internet X.509 Public Key Infrastructure Proxy Certificate Profile, http://www.ietf.org/internet-drafts/draft-ietf-pkix-proxy-08.txt

[20] The Semantic Grid, http://www.semanticgrid.org

[21] SOAP Security Extensions: Digital Signature, W3C NOTE 06 February 2001, http://www.w3.org/TR/SOAP-dsig/

[22] Surridge, M, “A Rough Guide to Grid Security”, published via http://www.nesc.ac.uk by the UK E-Science Programme.

[23] Ashri, R., Payne, T., Marvin, D., Surridge, M., Taylor, S., “Towards a Semantic Web Security Infrastructure”, in Payne, T.R., Decker, K., Lassila, O., Mcilraith, S., Sycara, K., eds: *First International Semantic Web Services Symposium (2004 AAAI Spring Symposium Series), AAAI (2004).*

[24] Web Services Addressing (WS-Addressing) Specification, http://www-106.ibm.com/developerworks/webservices/library/ws-add/

[25] Web Services Notification and Web Services Resource Framework, http://www-106.ibm.com/developerworks/webservices/library/ws-resource/

[26] Web Services Security (WS-Security) Specification, http://www-106.ibm.com/developerworks/webservices/library/ws-secure/