A standards-based architecture for Grid Service Management

Keith Rochford, John Walsh, Eamonn Kenny, Brian Coghlan
Computer Architecture Group, Trinity College Dublin, Ireland
keith.rochford@cs.tcd.ie, john.walsh@cs.tcd.ie, eamonn.kenny@cs.tcd.ie, coghlan@cs.tcd.ie

Abstract

Managing the availability of services is an essential task in ensuring the availability of a grid infrastructure. This project aims to take the next step in the monitoring and management of grid computing systems by developing a standards-based distributed control plane for grid resources and middleware components. While many existing projects focus on the monitoring of grid resources and infrastructures, they make little or no provision for the active or pro-active management operations required to ensure the availability of the infrastructure.

This solution employs web service technologies with the intention of producing an extensible body of work that might be applied to a range of grid projects and middlewares. With the emergence of meta-grids and the increasing importance of grid middleware interoperability, the acceptance and adoption of standard interfaces will become an important step in the development of future grid components.

1. Introduction

The area of grid computing is currently attracting a considerable amount of attention from a large number of research and development groups. This has led to many projects adopting different approaches to satisfy common requirements, often re-inventing or rehashing existing work[27]. The resulting lack of standards has resulted in a variety of tools which cannot easily be combined or extended to form more complete management solutions. It is not possible for developers or administrators to easily combine the best features of each into a system that satisfies their individual needs.

One lesson that the developers and operators of grid infrastructures might learn is derived from the integration difficulties experienced within the telecommunications sector: they should differentiate their offerings based on service rather than technology. The increasingly important subject of interoperability is often hampered by each project developing their own specific means of achieving a given goal. With this in mind it seems appropriate to investigate and evaluate standard tools and mechanisms for data communications and representation when designing or planning any contribution to the field.

In this paper we provide an overview of one such standard and present an implementation of a grid service management tool based upon it. The requirement for this system has grown out of our research into grid monitoring, and the desire to ease the day-to-day administration of the infrastructure, whilst increasing its availability and speeding the response to security events.

In the interest of simplicity, we will define *service management* as the ability to exercise a degree of control over the status and availability of a service running on a remote host. In addition, it should be possible to query the status of a given service and request a list of managed services from a given resource.

The system presented in this paper is an enabling technology of Grid4C (Grid Foresee), our grand vision of a Command and Control system for grid infrastructures. Grid4C will bring together monitoring and status information from a variety of sources along with a distributed control plane based upon this work, and make it available to human and machine operators for rich interactive or autonomous control. In addition to a monitoring and administrative tool, it is envisaged that Grid4C will provide a Decision Support System for members of a Grid Operations Team.

The design and implementation of this system has been motivated by the activities of the Grid-Ireland operations team and so we now describe the context within which this work has taken place.

1.1. Context

Grid-Ireland is a managed layer providing grid services above the Irish research network. It allows researchers to share computing and storage resources using a common interface, and facilitates international collaboration by linking Irish sites to into European grid infrastructures being developed under such EU projects as EGEE[6], LCG[10],

Sixth International Symposium on Parallel and Distributed Computing (ISPDC'07)
0-7695-2936-4/07 $20.00 © 2007 IEEE

CrossGrid[4], and int.eu.grid[9].

The Grid-Ireland operations centre is based at Trinity College Dublin and is staffed by members of our research group. Among its responsibilities is the deployment and maintenance of an homogeneous core infrastructure[17] comprised of *grid gateways*. Installed at each site, these gateways provide a point of access both for external grid users to gain access to the site resources and local site users to gain access to remote grid resources.

The gateway[15] provides the essential functionality of a grid site, thus freeing site administrators to concentrate on the management of their own resources. The gateways are remotely managed by the operations centre. Local cluster administrators need not concern themselves with the provision of grid services as these are an integral part of the deployed gateways. Resources at a site remain under the control of the site administrators who are free to manage as they see fit, without having to concern themselves with the finer details of grid integration.

Thus the core infrastructure of grid gateways remain the responsibility of the operations team. The ensuing possibility of homogeneity of this core infrastructure presents a number of opportunities to the users, grid operators and the site administrators. These include:

- Minimizing the proportion of software components that need to be ported to non-reference platforms.
- Maximizing the availability of the infrastructure while reducing the effort required for software deployment. The common software and hardware components facilitate centralised management and push-button transactional deployment of middleware components[18], guaranteeing uniform responses to management actions.
- Decoupling of grid infrastructure and site management. The fact that the site resources and the grid infrastructure are independent of each other allows for variation in design, deployment, and management.
- The installation of heterogeneous site resources based on non-reference architectures is also supported. In order to support a wide range of connected resources, the operations team carryout porting of the necessary middleware components[22].

Each grid gateway is composed of a set of seven machines: a firewall, an install server, a compute element (CE), a storage element (SE), a user interface (UI), a test worker node that is used for gateway testing, and optionally a network monitor. In general these are implemented as virtual machines on one physical host. All sites are identically configured with grid software based on gLite3.

1.2. Aims

1.2.1 Ease of use

The grid service management solution should provide an intuitive and relatively transparent interface, readily accessible to the members of an operation team. The client components should be designed in such a way that they might easily be incorporated into a variety of user interfaces, including mobile, web-based, and advanced visualisation tools.

1.2.2 Flexibility

While the system described here illustrates the function of service management, the underlying architecture should be readily extensible to support a variety of administrative operations. Through the use of a plug-in architecture and standards-based interfaces it should be possible to extend and customise the control infrastructure to meet specific needs. Similarly, it should be possible to develop custom control interfaces provided all security concerns are satisfied.

1.2.3 Performance

Execution of management operations should occur in near real-time and provide adequate feedback regarding their progress or output. Due to the administrative function of the system, it must be robust and resilient. Efforts should be made to ensure its availability irrespective of the status of other resources within the administrative domain. In order to function as a control plane, it should be independent of its managed resources, fast, reliable and 'always-on'. In addition, the design must aim to minimise the impact of the management system on the normal operation of the managed resources.

1.2.4 Security

Since the proposed system is to be capable of executing tasks normally carried out only by users with administrative priviledges, security is very obviously of utmost importance. Invocation of the management web services must be restricted to authorised users or processes and all communication should take place over secure channels.

1.2.5 Minimal site-specific configuration

In order to facilitate deployment and administration, it should be possible to define a generic set of operations to be supported by the system for each of the resource types at each of the intended target sites. For example, the properties and operations defined for a Compute Element, such as querying current jobs or starting the resource allocation

manager, are likely to differ little from site to site. It should therefore be possible to deploy a largely pre-configured system with minimal reconfiguration and automatic detection of configuration variables where possible. Where possible, deployment via a fabric management system should be supported.

2. Related Work

While much work has been done in the area of Grid monitoring, few projects have attempted to close the control 'loop' by providing interfaces to rectify detected anomalies or reconfigure components. One promising early work is the C.O.D.E.[24] project developed for the NASA Information Power Grid. Unfortunately this project pre-dated much of the work on Web Service and management standards and the control functionality was never fully implemented.

With evolution towards grid middlewares exposing functionality via Web Services[21][25], the management of these services has become an increasingly important and active research topic[26][12][19]. However, few of these works take into account the requirements for managing 'legacy' grid services, i.e. non Web Service based, or their hardware platforms via standard Web Services interfaces.

We are concerned not only with the management of the grid software, but also of the physical hardware resources comprising our gateways. We present an architecture which leverages the benefits of agent based design [13][20] and allows the monitoring and management of 'legacy' grid services via a standards-based Web Service interface. Although this architecture could, and most probably will be extended to monitor and control Grid functionalities exposed via Web Services, we consider this a separate problem domain for which there are more relevant management standards such as the Web Service Distributed Management: Management Of Web Services (MOWS)[5]. Based on our proposed architecture, we present a concrete, extensible implementation of a grid management tool that attempts to build on established work and contribute to future development by promoting the use of standard interfaces.

3. Architecture

In this section, we describe the architecture of of the service management system at a high level so that the basic building blocks can be seen clearly without implementation details intruding.

The architecture of this system employs a multi-tier approach. A client/server architecture exists between the operations centre and the sites comprising the managed hosts or entities. Within the sites, a second hierarchy exists where incoming requests or operations are routed to the relevant request handlers on the managed entities.

Due the fact that the managed resources often reside within remote administrative domains, over which the grid operations team have little control, our architecture employs a single management server deployed to each site. This server acts as a gateway to the management endpoints within the site, in addition to containing middle tier logic such as the scheduling of service checks, persistence of event information and policy decision logic. This single point of presence allows us to limit the control traffic into and out of the site to known ports (HTTPS) on a single machine rather than requiring direct network access to each of the managed entities. This approach also allows the software deployed to each of the managed entities to be relatively lightweight as it need only to communicate with the local management server rather than supporting an entire web services stack.

The software components deployed to each managed entity are responsible for implementing the interfaces exposed via the web services on the management server. In this respect, much of the web service logic is composed of proxy objects responsible for routing the requests to their corresponding control elements on the managed entities.

The core components of the architecture, as illustrated in figure 1, include the following:

- Management Clients - Standalone or embedded components, capable of invoking the management capabilities exposed via Management Servers. These are typically executed within the systems of the operations centre. Mobile clients will also be investigated.

- Management Servers - A single management server is deployed to each site with the primary responsibility of exposing the management capabilities of the resources within that site or domain as web service endpoints. Additional functions include the scheduling of monitoring and administrative tasks, persistence of event information and in the case of autonomous control, the role of a Policy Decision Point.

- Control Elements - These are the lightweight components deployed to each of the managed resources. They provide a control interface to the resource and implement the logic defined in the services exposed via the Management Servers. Operations requested of the Control Elements may be used to monitor or alter some property of the resource.

- Final Elements - Using terminology borrowed from the field of process control, the Final Elements are the lowest level components of the architecture. Residing on the managed resources and accessed via the Control Element, Final Elements are used to sense state and perform action operations on the resource. They are

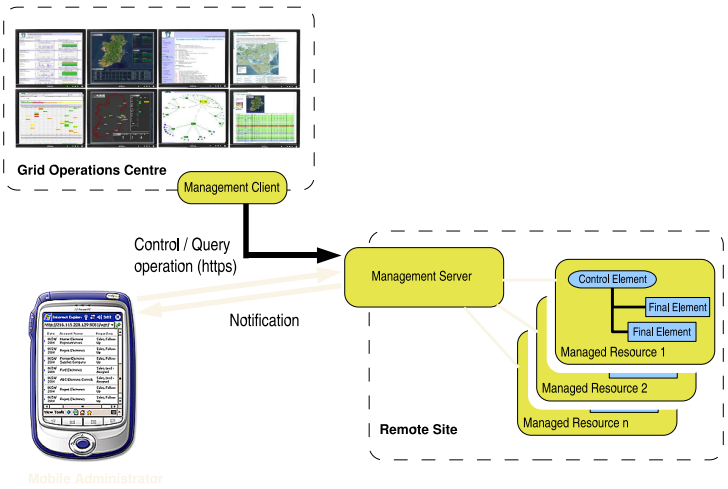

Figure 1. Architectural overview of control backplane.

implemented as plug-in objects and can be either self-contained or rely on some proprietary local application programming interface such as the Intelligent Power Management Interface from Intel. Examples of final elements might include:

- Service Check - used to determine the status or availability of grid services
- Service Control - used to alter the state of a given service
- Network control - Control over network interfaces and traffic. These might for example be used to 'quarantine' problematic hosts
- Hardware Management - user to expose hardware management functions via the secure web service interface, e.g. power cycling resources
- Job Execution - execute grid jobs such as those designed to evaluate performance and availability
- Job Management - provide remote control over local job schedulers, e.g. the removal of 'stuck' jobs
- Access Control - interface to user authorisation components
- Resource Control - more coarse grained control over the state or availability of the resource

4. The role of WSDM

The increasing complexity of computer networks and infrastructures has led to many operations teams struggling to deal with the additional overhead involved in their monitoring and management. This has led resource providers to provide monitoring and management tools intended to ease the burden of the deployment and maintenance of their systems. However, in heterogeneous environments, either software or hardware, the integration of these tools can become troublesome in itself. Operations staff may be forced to use multiple systems and manually aggregate or correlate information across them in order to aquire the necessary representation of system state. The use of proprietary interfaces and persistence models without common standardised interfaces limits the possibility of managing the components of heterogeneous systems in a truly cohesive manner.

One possible solution to these integration and interoperability problems is the Web Services for Distributed Management[5] (WSDM) standard from the Organisation for the Advancement of Structured Information Standards (OASIS). WSDM defines standard mechanisms for the representation of, and access to, resource management interfaces implemented as Web Services. In addition, it defines how web services themselves may be managed as resources. The standard defines two specifications; WSDM: Management Using Web Services (MUWS) and WSDM: Management of Web Services (MOWS). Mechanisms are defined for identifying, inspecting, and modifying characteristics of resources, thus ensuring the interoperability of management tools and management resources from different vendors or development groups.

Rather than being designed from the ground up, WSDM incorporates elements of numerous existing Web Service technologies, including WS-RF, WS-Notification and WS-Addressing. By employing web services, WSDM inherits essential distributed computing functionality, interoperabil-

ity and implementation independence.

4.1. The WSDM development model

While WSDM defines the interfaces and the reference implementation provides the necessary infrastructure, the creation of the custom manageability capabilities that will be exposed via WSDM if left to the developer of the resource management solution. These capabilities, comprising data and operations, are the fundamental building blocks of resource management endpoints. Developers create WSDL files that define the web service interface, and must include one port type for the resource that includes all the resources' public operations that a client would need in order to inspect and manipulate it.

In addition to custom capabilities defined for a particular resource, the WSDM specification includes a number of standard capabilities, these include:

- Identity - The only required capability, used to differentiate among resources. This capability contains exactly one property, *ResourceID*, which is unique to the resource.
- Description - The list of captions, descriptions and version information used to provide a human readable identity for the resource.
- Metrics - Defines how to represent and access information about a specific property as well as the current time on the resource.
- State - Defines how to change the state of a resource according to a specific state model
- Operational Status - Defines three status levels for a resource (*available*, *unavailable* and *unknown*) along with status change events.
- Advertisement - A standard event to be generated when a new manageable resource is created.

Other features of the WSDM standard that are of particular interest to our application include *Relationships* and *Notifications*.

4.1.1 Relationships

WSDM defines interfaces which can be used to query a manageable resource about the relationships in which it participates. In our application these might be used to arrange the manageable resources into groups based, for example, on their location or their resource type.

4.1.2 Notifications

WSDM also defines an extensible XML event format that defines a set of data elements allowing different types of management information to be transmitted, processed, correlated, and interpreted across different products and platforms using different technologies. Each WSDM capability has a corresponding WS-Notifications topic which can be used to identify and categorize capability-specific notifications such as property change events or state transition topics.

5. Implementation

5.1. Overview

Our system has been developed in Java using Apache Muse[1], a reference implementation of the WSDM specification. The Muse endpoints are deployed within an Axis2[3] SOAP engine on the management servers deployed to each site. Communications within each site, between the Management Server and a Control Element, are carried out using XML-RPC[2] with two-way authentication over SSL. Once the management server is in place and the Control Elements are installed and configured on the resources to be managed, the system makes the management functions provided by the Final Elements available to authorised users via web services interfaces compliant with the WSDM standards. The management server provides a single point of entry to administer the resources within its site. One of the advantages of using this reference implementation is that it includes Web Service Notification specifications which can be used to construct an event driven communication model for our management infrastructure. In our implementation, the WSDM Status capability of a managed resource is often representative of the logical value of a number of lower states. For example the status of a resource might only be set to *Available* following the successful start-up of all its services and the verification of its availability via a local or remote sensor. If this value is subsequently changed either by a local or remote process, all parties that had subscribed to the topic for that capability will be automatically notified via a property change event.

Figure 2 shows a more detailed architecture of the control and management system.

5.2. The Control Element & Final Elements

The Control Element is implemented as an XML-RPC server along with a set of request handlers. Examples of these request handlers include Final Element Managers, responsible for maintaining a list of registered Final Elements and triggering their execution in response to requests, and

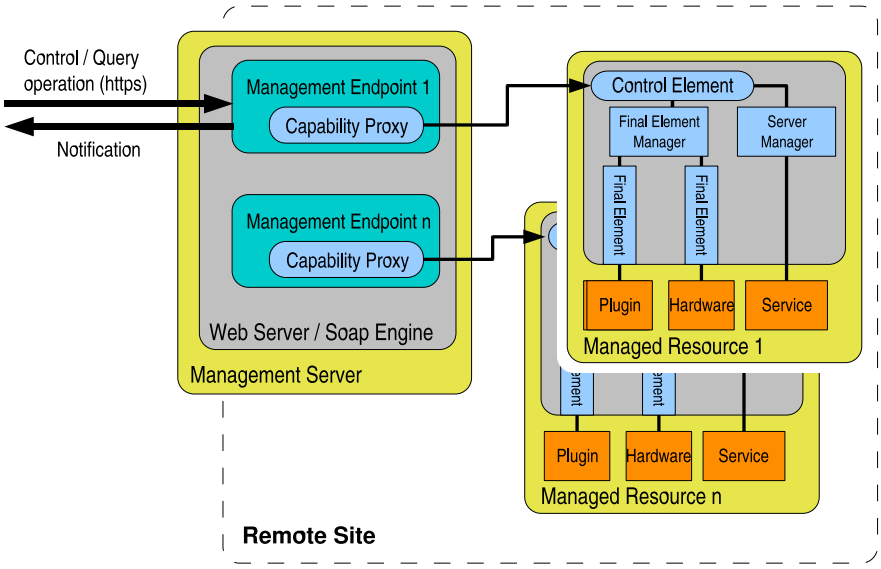

Figure 2. Architecture of remote management capabilities.

Server Managers, responsible for more generic resource management operations such as start and stop, etc.

The Control Element performs a similar duty to the Management Server, albeit on a smaller scale, in that it provides a single point of access to the management functions of a resource. The advantages of this architecture include a smaller footprint on the managed resource, a single location for authentication and authorisation, and a more straightforward usage model. A Final Element may be addressed in the form *resource.requesthandler.element* although a number of XML-RPC client objects have been written to encapsulate and simplify this functionality. Objects wishing to invoke some operation on the Control Element can simply instantiate one of these client objects and use methods such as *resource.stop*, *resource.getServiceStatus(serviceName)*, or *resource.executeFinalElement(elementName)*.

The term Final Element is used to describe a component that performs the duties of a sensor, actuator or both. They are the last elements in the chain of command and constitute the bridge between the control system and the managed resource. The Final Elements are implemented as java objects implementing a specific interface allowing them to be easily 'plugged-in' to the control hierarchy. Typically the output of the operations supported by the final elements is in XML form. Final Elements can be self-contained Java objects, include some functionality native to the resource platform, or make use of an existing application programming interface. One example of using Final Elements as wrapper objects is described in [23], this illustrates how the extensive range of sensors developed for the Nagios project can be exploited from a Java based monitoring application.

5.3. The Management Server

The Management Server can be hosted on any machine capable of running a servlet container provided it has the necessary network access to the Control Elements within its domain and also to the external control clients. For this reason, the Management Server would typically be installed as part of the site gateway. The WSDM management endpoints reside within an Axis container on this Management Server. These make the control functionality offered by the Control Elements available to the Control Clients via the published web interfaces. These interfaces make use of both standard and custom capabilities. Since the managed resources would typically not reside on the same host as the Management Server, capability proxy objects are employed to route the web service requests over XML-RPC to the Control Elements on the target resource.

5.4. The Control Client

Currently the Control Clients are invoked using command-line tools. More complex graphical interfaces are under development. Using the WSDL that defines the management endpoint, and the *wsdl2java* tool included in the Muse distribution, it is possible to automatically generate client code which can be used to invoke the operations defined in the WSDL. It is then relatively straightforward to

create user interface code to invoke management operations without having to write code at the web services layer.

5.5. Security

Since security is of critical importance in this application it is incorporated on several levels. All traffic between the Control Client and the Management Servers takes place over HTTPS connections with client and server authentication. Further security is provided at the web services layer using Apache Rampart[11]. Within a remote management domain, all traffic between the Management Server and Control Elements takes place using XML-RPC over SSL. The XML-RPC servers are also secured using access control lists which typically only allow their operations to be available to the local Management Server. In addition, in the Grid-Ireland case, because the gateways are implemented as virtual machines upon a physical host running Xen [14], the physical host communications represent an out-of-band control plane. This adds greatly to the control plane's security. In the event of a total failure of the physical host, a further secure out-of-band control plane is available using the host's remote management hardware, which employs IPMI[8].

6. Testing

This software is deployed and undergoing testing on a fully functional replicated grid environment [16] at the Grid-Ireland operations centre. Following further development, testing and security evaluation, the solution will be deployed across the production Grid-Ireland infrastructure. It is becoming an integral component of our operations strategy.

6.1. Performance

While there is some overhead incurred through the use of SOAP messages, security mechanisms, and the routing of requests from the Management Servers to the Control Elements, we feel that the benefits of the web service technologies and the reduced deployment effort justify the expense. Preliminary tests of the management operations show response times well within acceptable boundaries.

6.2. Scalability

It is expected that the architecture will scale favourably due to the devolution of much of the management functionality to the Management Servers and managed resources. The majority of the message sizes between the control clients and the managed resources will be relatively small, and multi-threaded request handlers within the Control Elements will serve to reduce execution overheads.

7. Use cases

- Service Outages - Upon receiving notification of a failed service on one of the grid gateways, a member of the operations team can use a Control Client to restart the service via one of the user interfaces. The Control Client will issue a service management request to the Control Element on the resource hosting the failed service,
- Managing batch queues - The system could be used to exercise control over the batch processing queues connected to the Compute Element at the grid site. An example of this would be the removal of 'stuck' jobs from a queue. A job may appear to be 'stuck' if the Worker Node on which it was executing hangs and the job status is not updated by the queue manager
- Security Events - In the event of an security incident or notification of a security flaw in the grid middleware, the operations staff can shut down or quarantine the relevant elements 'at the touch of a button'.
- Automatic service recovery - This system facilitates the creation of automated processes that can subscribe to status events and upon certain notification, execute pre-defined service recovery operations in order to attempt to return the system to an operational state. The processes would then request a service check operation to verify that the service was restored. If the service was not restored, an alternative operation might be attempted or failing that, an event would be generated to escalate the alert to a member of the operations team.

8. Conclusions

In this paper we have presented a number of motivations for the adoption of standards-based development practices. Following a brief description of one such standard, Web Services for Distributed Management, we described a prototype implementation of a grid service management tool, Grid4C, using that standard. This is now an open source project[7].

The system we have presented illustrates how the use of lightweight components deployed to each of the managed resources within a site can be used to provide a single point of access to resource management. We believe that this solution is an optimal configuration allowing the use of WSDM for fabric management without the necessity of deploying a web services container to each of the managed resources. We have also illustrated how this single point

of access is particularly important on an infrastructure such as ours, where a large proportion of the managed resources reside within networks beyond our control.

Although there is a learning curve associated with the adoption of any new technology, the use of standard interfaces will result in software components that can be more easily integrated into existing environments. With movement towards service oriented architectures, where users will piece together solutions to meet their needs, the flexibility offered by the use of common interfaces will be critical.

9. Future Work

While the system described in this paper forms the basis of a useful tool in it's own right, there is considerable scope for further work.

Here, we have limited our discussion to the use of [WSDM]MUWS for the management of resources and network services. With the migration to grid middleware technologies based on web services, there is considerable scope for investigation into how [WSDM]MOWS capabilities might be incorporated into such grid services so that standards-based manageability can be natively supported.

Continued development of the components of the Grid4C system is expected in addition to further work in the areas of autonomic control, policy based management and event persistence. The use of visualisation tools as user interfaces to the control system, along with the development of mobile clients, will be actively investigated.

References

[1] The apache muse project.

[2] Apache xml-rpc.

[3] Axis2 - apache webservices project.

[4] Crossgrid.

[5] Defining a web services architecture to manage distributed resources.

[6] Enabling grids for e-science (egee).

[7] Grid4c - command and control for grids.

[8] Intelligent platform management interface.

[9] Interactive european grid project.

[10] Lhc computing project lcg.

[11] Rampart : Ws-security module for axis2.

[12] S. Albayrak, S. Kaiser, and J. Stender. Advanced grid management software for seamless services. *Multiagent Grid Syst.*, 1(4):263–270, 2005.

[13] J. Cao, D. Spooner, J. D. Turner, S. Jarvis, D. J. Kerbyson, S. Saini, and G. Nudd. Agent-based resource management for grid computing. In *CCGRID '02: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 350, Washington, DC, USA, 2002. IEEE Computer Society.

[14] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. Deployment of grid gateways using virtual machines. In P. M. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, Amsterdam, The Netherlands, February 2005. Springer.

[15] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, and J. Walsh. A single-computer grid gateway using virtual machines. In *Proc. AINA 2005*, pages 761–770, Taiwan, March 2005. IEEE Computer Society.

[16] S. Childs, B. Coghlan, D. O'Callaghan, G. Quigley, J. Walsh, and E. Kenny. A virtual testgrid or how to replicate a national grid. In *Proceedings of the EXPGRID workshop on Experimental Grid testbeds for the assessment of large-scale distributed applications and tools*, Paris, June 2006.

[17] B. Coghlan, J. Walsh, and D. O'Callaghan. Grid-ireland deployment architecture. In P. M. Sloot, A. G. Hoekstra, T. Priol, A. Reinefeld, and M. Bubak, editors, *Advances in Grid Computing - EGC 2005*, LNCS3470, Amsterdam, The Netherlands, February February, 2005. Springer.

[18] B. Coghlan, J. Walsh, G. Quigley, D. O.Callaghan, S. Childs, , and E. Kenny. Transactional grid deployment. In M. Bubak, M. Turala, and K. Wiatr, editors, *Proc. Crakow Grid Workshop (CGW'04)*, pages 363–370, Cracow, Poland, December 2004. Academic Computer Centre CYFRONET AGH.

[19] K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, and M. Xu. Agreement-based grid service management (ogsi-agreement), 2003.

[20] I. Foster, N. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other, 2004.

[21] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration, 2002.

[22] E. Kenny, B. Coghlan, J. Walsh, S. Childs, D. O'Callaghan, and G. Quigley. Autobuilding multiple ports of computing nodes for grid computing. In *Cracow Grid Workshop (CGW'05)*, Cracow, Poland, November 2005.

[23] K. Rochford, B. A. Coghlan, and J. Walsh. An agent-based approach to grid service monitoring. In *Proc. International Symposium on Parallel and Distributed Computing (ISPDC 2006)*, July July, 2006.

[24] W. Smith. A system for monitoring and management of computational grids. In *ICPP '02: Proceedings of the 2002 International Conference on Parallel Processing (ICPP'02)*, page 55, Washington, DC, USA, 2002. IEEE Computer Society.

[25] M. Theimer, S. Parastatidis, T. Hey, M. Humphrey, and G. Fox. An evolutionary approach to realizing the grid vision, 2006.

[26] H.-L. Truong, R. Samborski, and T. Fahringer. Towards a framework for monitoring and analyzing qos metrics of grid services. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 65, Washington, DC, USA, 2006. IEEE Computer Society.

[27] S. Zanikolas and R. Sakellariou. A taxonomy of grid monitoring systems. *Future Gener. Comput. Syst.*, 21(1):163–188, 2005.