GWD-I (draft-ggf-ogsa-spec-018)
Open Grid Services Architecture
http://forge.gridforum.org/projects/ogsa-wg

Editors:
I. Foster, Argonne & U.Chicago
H. Kishimoto, Fujitsu

June 7, 2004

The Open Grid Services Architecture, Version 1.0

Status of this Memo

This document provides information to the community regarding the specification of the Open Grid Services Architecture (OGSA). Distribution of this document is unlimited. This is a DRAFT document and continues to be revised.

This document is being constantly updated. The latest version can be found at:
https://forge.gridforum.org/projects/ogsa-wg/document/draft-ggf-ogsa-spec/en/

Abstract

Successful realization of the Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration, virtualization, and management requires the definition of a core set of interfaces, behaviors, resource models, and bindings. This document, produced by the OGSA working group within the Global Grid Forum (GGF), provides a first, and necessarily preliminary and incomplete, version of this OGSA definition. The document specifies the scope of important services required to support Grid systems and applications in both e-science and e-business, identifies a core set of such services that are viewed as essential for many systems and applications, and specifies at a high-level the functionalities required for these core services and the interrelationships among those core services. The document also lists existing technical standards and standard definition activities within GGF, OASIS, W3C, and other standards bodies that speak to required OGSA functionality, and identifies priority areas for further work.

ogsa-wg@ggf.org

GLOBAL GRID FORUM
office@ggf.org
www.ggf.org

Full Copyright Notice

Copyright © Global Grid Forum (2002, 2003, 2004). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director (see contact information at GGF website).

Contents

1 Introduction

Grid systems and applications aim to integrate, virtualize, and manage resources and services within distributed, heterogeneous, dynamic "virtual organizations" [6, 7]. The realization of this goal requires the disintegration of the numerous barriers that normally separate different computing systems within and across organizations, so that computers, application services, data, and other resources can be accessed as and when required, regardless of physical location.

Key to the realization of this Grid vision is standardization, so that the diverse components that make up a modern computing environment can be discovered, accessed, allocated, monitored, accounted for, billed for, etc., and in general managed as a single virtual system—even when provided by different vendors and/or operated by different organizations. Standardization is critical if we are to create interoperable, portable, and reusable components and systems; it can also contribute to the development of secure, robust, and scalable Grid systems by facilitating the use of good practices.

In this document, we present an *Open Grid Services Architecture* (OGSA) that addresses this need for standardization by defining, within a service-oriented architecture, a set of core interfaces and behaviors that address key concerns in Grid systems. These concerns include such issues as: How do I establish identity and negotiate authentication? How is policy expressed and negotiated? How do I discover services? How do I negotiate and monitor service level agreements? How do I manage membership of, and communication within, virtual organizations? How do I organize service collections hierarchically so as to deliver reliable and scalable service semantics? How do I integrate data resources into computations? How do I monitor and manage collections of services?

In our presentation, we first identify required capabilities for Grid systems and then translate those capabilities into a coherent set of components that collectively define OGSA. First, in §2, we provide an abstract definition of the set of requirements that OGSA is intended to address. This analysis is based on requirements, technical challenges, use cases, previous experience, and the state of the art in related work. The abstract rendering is not constrained by any assumptions about the underlying infrastructure, but rather is intended to frame the "Grid" discussion and define solutions.

In §3, we present a refinement of the required functionality into capabilities. In §3.2, we describe infrastructure services and assumptions that constrain our development of the OGSA design. In particular, we explain how OGSA both builds on, and is contributing to the development of, the growing collection of technical specifications that form the emerging Web services (WS) architecture [2]. Further sections describe the identified capabilities: Execution Management, Data, Resource Management, Security, Self-management and Information services.

In a later draft we will provide descriptions of specific services, making clear where existing service specifications can be used unchanged and where modified or new service specifications are needed. We will also describe the current state of any work known to be underway to define such extensions and/or definitions.

This document, a product of the Global Grid Forum's OGSA working group, defines OGSA version 1. The OGSA working group is likely to release revisions of this document in the future as our understanding of OGSA's purpose and form, and the details of specific components, evolves.

2 Requirements

This definition of OGSA version 1 is driven by a set of functional requirements, which themselves are informed by the use cases listed in Table 1 and Table 2 and described in companion documents [4][?]. These use cases do not constitute a formal requirements analysis, but have provided useful input to the definition process.

Table 1 The OGSA use cases

Use case	Summary
Commercial Data Center (CDC)	Data centers will have to manage several thousands of IT resources, including servers, storage, and networks, while reducing management costs and increasing resource utilization.
Severe Storm Modeling	Enable accurate prediction of the exact location of severe storms based on a combination of real-time wide area weather instrumentation and large-scale simulation coupled with data modeling.
Online Media and Entertainment	Delivering an entertainment experience, either for consumption or interaction.
National Fusion Collaboratory (NFC)	Defines a virtual organization devoted to fusion research and addresses the need of software developed and executed by this community based on the application service provider (ASP) model.
Service-Based Distributed Query Processing	A service-based distributed query processor supporting the evaluation of queries expressed in a declarative language over one or more existing services.
Grid Workflow	Workflow is a convenient way of constructing new services by composing existing services. A new service can be created and used by registering a workflow definition to a workflow engine.
Grid Resource Reseller	Inserting a supply chain between the Grid resource owners and end users will allow the resource owner to concentrate on their core competence, while end users can purchase resources bundled into attractive packages by the reseller.
Inter Grid	Extends the CDC use case by emphasizing plethora of applications that are not grid enabled and are difficult to change, mixed grid and non grid data centers, grid across multiple companies, etc. Also brings into view generic concepts of utility computing.
Interactive Grids	Compared to the online media use case, this use case emphasizes a very high granularity of distributed execution.
Grid Lite	Extends the use of grids to small devices – PDAs, cell phones, firewalls etc, and identifies a set of essential grid services that enable the device to be part of a grid environment.
Virtual Organization Grid Portal	A VO gives its members access to various computational, instrument-based data and other types of resources. A Grid portal provides an end-user view of the collected resources available to the members of the VO.

Persistent Archive (PA)	Preservation environments handle technology evolution by providing appropriate abstraction layers to manage mappings between old and new protocols, software and hardware systems, while maintaining authentic records.
Mutual Authorization	Refines the CDC and NFC use cases by introducing the scenario of the job submitter authorizing the resource on which the job will eventually execute.
Resource Usage Service (RUS)	Facilitates the mediation of resource usage metrics produced by applications, middleware, operating systems, and physical (compute and network) resources in a distributed, heterogeneous environment.

Table 2 The OGSA Tier-2 use cases

Use case	Summary
IT Infrastructure and Management	Job execution, cycle sharing and provisioning scenarios.
Application Use Cases	Peer-to-Peer PC Grid computing, file sharing and content delivery scenarios.
Reality Grid	Distributed and collaborative exploration of parameter space through computational steering and on-line, high-end visualization.
The Learning GRID	User-centered, contextualized and experiential based approaches for ubiquitous learning in the framework of a Virtual Organization.
HLA-based Distributed Simulation	A distributed collaborative environment for developing and running simulations across administrative domains.
GRID based ASP for Business	An infrastructure for Application Service Provision (ASP) supporting different business models based on GRID technology.
Grid Monitoring Architecture	Grid monitoring system scalable across wide-area networks and able to encompass a large number of dynamic and heterogeneous.

Analysis of the use cases and other relevant input led us to identify both important and apparently broadly relevant characteristics of Grid environments and applications, and functionalities that appear to have relevance to a variety of application scenarios. We summarize our findings in the following sections.

2.1 Dynamic and heterogeneous environment support

Some use cases involve highly constrained or homogeneous environments that may well motivate specialized profiles. However, it is clear that, in general, Grid environments tend to be heterogeneous and distributed, encompassing a variety of operating systems (e.g., Unix, Linux, Windows, or embedded systems), hosting environments (e.g., J2EE, .NET), devices (e.g., computers, instruments, sensors, storage systems, databases, networks), and services provided by various vendors. In addition the environment is long lived, dynamic, with frequent changes and may therefore evolve in ways not initially anticipated.

OGSA must enable interoperability between these diverse, heterogeneous, and distributed resources and services as well as reduce the complexity of administrating heterogeneous systems. Moreover, many functions required in distributed environments, such as security and resource management, may already be implemented by stable and reliable existing legacy systems, and thus it is required to be able to integrate such legacy systems into the Grid.

- *Resource Virtualization.*
  Resource virtualization is essential to reduce the complexity of managing heterogeneous systems and to handle diverse resources in a unified way.
- *Common Management Capabilities.*
  Simplifying administration of a heterogeneous system requires mechanisms for uniform and consistent management of resources. A minimum set of manageability capabilities are required.
- *Resource Discovery and Query*.
  Mechanisms are required for discovering resources with desired attributes and retrieving their properties. Discovery and query should assume a highly dynamic system.
- *Standard Protocol.*
  Standard protocols are important for interoperability and also to simplify the transition to using Grids.
- *Standard Schema.*
  Standard schemas are important for interoperability.

2.2 Resource sharing across organizations

The Grid is not a monolithic system but is composed of resources owned and controlled by various organizations. One of the main purposes of OGSA is to support resource utilization across administrative domains. Mechanisms are needed to provide a context that can be used to associate users, requests, resources, policies, and agreements across organizational boundaries. Sharing resources across organizations implies various security requirements. These are described in §2.7.

- *Global Name Space.*
  A global name space is needed to ease data and resource access. OGSA entities should be able to transparently access, subject to security constraints, other OGSA entities without regard to location or replication.
- *Metadata Catalog*.
  Finding, invoking, and tracking entities by means of metadata. It should be possible to share metadata catalogs such as data catalogs across administrative domains.
- *Site autonomy.*
  Mechanisms are required for accessing resources across sites while respecting site autonomy (see Delegation in §2.7).
- *Accounting (billing).*
  Utilizing resources across organizations introduces stricter accounting requirements. Mechanisms for collecting and exchanging necessary information for accounting are required, including standard schemas. It should be possible to connect to existing accounting systems.

2.3 Optimization

Optimization includes optimizing both the supply and the demand sides. One common case of supply side optimization is resource optimization. For example, resource allocation often provides for the worst case scenarios (e.g., highest expected load, recovering from failures, etc) and leads to resource underutilization as resources allocated to address such scenarios remain unused for long periods. Resource utilization can be improved by flexible resource allocation including sharing of backup resources. Demand side optimization includes different ways of managing workloads. Mechanisms for monitoring resource usage, changing resource allocation, and provisioning resources dynamically and on demand are the required foundation.

- *Resource Utilization.*
  Capabilities to kept track of resource utilization including metering, monitoring and logging are essential to realize the high-level requirements in this section. These resource utilization functions should be designed considering large-scale systems.
- *Resource Allocation Planning.*
  Based on policies so resources can be used in a well-planned way.
- *Workload Management*

2.4 QoS Assurance

Services such as job execution and data services must provide the agreed upon QoS. Key QoS dimensions include but are not limited to: availability, security, and performance. These cornerstone QoS dimensions should be expressed in terms of measurable intent which can be captured in Service Level Agreements.

- *Service Level Agreement*
  QoS should be represented by agreements which are established by negotiating between service requester and provider prior to service execution. Standard mechanisms should be provided to create and manage agreements.
- *Service Level Attainment*
  If the agreement requires it the resources used by the service should be adjusted so that the required QoS is maintained. Therefore mechanisms for monitoring services, estimating resource utilization, planning for and adjusting resource usage are required.
- *Migration*
  To adjust workloads for performance or availability

2.5 Job execution

OGSA must provide manageability for job execution throughout the lifetime of a job. Functionalities such as scheduling, provisioning, and starting or stopping and error-handling of jobs while maintaining their integrity even when the job is distributed over a great number of heterogeneous resources.

- *Support for Various Job Types.*
  Various types of jobs must be supported, both simple jobs and complex jobs such as workflow and composite services. Thus, mechanisms are required for controlling execution of job steps as well as orchestrating or choreographing services.
- *Job Management.*
  Managing jobs during their entire lifetime is essential. Also the consistency of the entire job, even in the case of complex jobs such as workflows or other composite services should be maintained.

- *Scheduling.*
  Scheduling and executing tasks based on such information as specified priority and workload of resources is required. It is also required to realize mechanisms for scheduling across administrative domains, using multiple schedulers.
- *Provisioning.*
  Provisioning is required to automate the complicated process of resource allocation, deployment, and configuration.
- *Deployment and Configuration.*
  It must be possible to deploy the required applications and data to resources and configure them automatically. If necessary deploying and re-configuring hosting environments such as OS and middleware to prepare the environment needed for job execution.

2.6 Data Services

Efficient access and movement of huge quantities of data is required in more and more fields of science and technology. In addition, data sharing is important, for example enabling access to information stored in databases managed and administered independently. In business areas, archiving of data and data management are essential requirements.

- *Data access.*
  Easy and efficient access to various types of data (such as streaming and cache) independent of its physical location, by abstracting underlying data sources is required. Mechanisms are also required for controlling access rights at different levels of granularity.
- *Data consistency.*
  Consistency must be maintained when the data is cached or replicated.
- *Data persistency.*
  Data and its association with its metadata should be maintained for their entire lifetime.
- *Data integration.*
  OGSA should provide mechanisms for integrating heterogeneous and distributed data. It is also required to be able to search data available in various formats in a uniform way.
- *Data provisioning.*
  The required data should be made available at the requested location. OGSA should allow for selection from various ways such as transfer, copying, and caching, according to the nature of data.

2.7 Security

Safe administration requires controlling access to services through robust security protocols and according to some security policy. Thus, authentication mechanisms are required so that the identity of individuals and services can be established, and service providers must implement authorization mechanisms to enforce policy over how each service can be used. Policy management and key management for cryptographic functions are also needed. Mechanisms are also required for integrating and interoperating with existing security infrastructures. In addition, standard, secure mechanisms are required which can be deployed to protect Grid systems while supporting safe resource sharing across administrative domains. Sharing of resources by service users requires some kind of isolation mechanism.

- *Authentication, Authorization, and Accounting*.
  Obtaining application programs and deploying them into a Grid system may require authentication/authorization. The Grid system may have to identify users' security policies. Authorization should accommodate various access control models and implementation.

- *Multiple Security Infrastructures*.
  Distributed operation implies a need to integrate and interoperate with multiple security infrastructures. OGSA needs to integrate and interoperate with existing security architectures and models which are typically difficult to be replaced.

- *Perimeter Security Solutions*.
  Many use cases require applications to be deployed outside the client domain. OGSA needs standard and secure mechanisms that can be deployed to protect institutions while also enabling cross-domain interaction without compromising local control of equipment for protecting domains, such as firewall policy and intrusion-detection policy.

- *Isolation*.
  Various kinds of isolation must be ensured, such as isolation of users, performance isolation, and isolation between content offerings within the same Grid system.

- *Delegation*.
  Mechanisms which allow for delegation of access rights from requestors to services are required. Minimum access rights required for job execution should be delegated. The authority transferred through delegation is scoped only to the task(s) intended and within a limited lifetime to minimize the risk of misuse of delegated authority.

- *Policy Exchange*.
  Service requestors and providers should be able to exchange dynamically security (among other) policy information to establish a negotiated security context between them.

- *Intrusion detection and protection*.
  Strong monitoring is required for intrusion detection, defects, as well as identification of misuses, malicious or otherwise, including virus or worm attacks. Ability to migrate attacks away from critical areas.

2.8 Administrative cost reduction

The complexity of administrating large scale distributed, heterogeneous systems increases administration costs and the risk of human errors. Support for common administration tasks, by automating administrative operations and consistent management of integrated resources, is needed.

- *Policy-based Management.*
  Policy-based management is required to automate Grid system control, so that its operations conform to the goals of the organization that operates and utilizes it. From the low-level policies that govern how the resources are monitored and managed to high-level policies that govern how business processes such as billing are managed, there may be policies at every level of the system. Policies may include SLAs, security, scheduling, and brokering.

- *Application Contents Management.*
  In order to install and keep up to date complex systems more efficiently and

automatically it is necessary to be able to specify and manage as a single logical unit a lot of different application related information. This allows administrators to maintain application components in a concise and reliable manner, even without expert knowledge about the applications.

- *Problem Determination.*
  Support is needed for the basic components required for automated problem determination in order for administrators to recognize and cope quickly with emerging problems.

2.9 Scalability

A large-scale Grid system can create added value such as drastically reducing job execution time, allowing for higher availability and scalability, thereby enabling new services. The large scale of the system can also present problems since it places novel demands on the management infrastructure.

- *Management of numerous resources.*
  The management architecture needs to scale to potentially thousands of resources of a widely varied nature. Management needs to be done in a hierarchical or peer-to-peer (federated/collaborative) fashion.
- *High-throughput computing.*
  Mechanisms for adjusting and optimizing parallel job execution in order to improve throughput of the entire computation process, as well as optimizing a single calculation process.

2.10 Availability

High availability is often realized by expensive fault-tolerant hardware and complex cluster systems. Because of the widespread use of IT systems to provide essential public infrastructure services, an increasing number of systems are required to operate at high availability while minimizing costs. Since Grid technologies enable transparent access to a wider resource pool, across organizations as well as within organizations, they can be used as one building block to realize stable, highly-reliable execution environments at reasonable cost. But due to the heterogeneity of the Grid, components with longer or more unpredictable Mean Time To Repair (MTTR) than existing high-reliability systems have to be used presenting difficult problems.

In such a complex environment policy-based autonomous control (see Policy-based management in §2.8) and dynamic provisioning (see Provisioning in §2.5) are keys to realizing systems of high flexibility and recoverability.

- *Disaster recovery.*
  Operation of a Grid system must be recovered quickly and efficiently in case of natural or human-caused disaster, avoiding long-term service disruption. Remote backup and simplification or automation of system reconstruction at reasonable cost is required.
- *Fault Management.*
  Running jobs must not be lost because of resource faults. Mechanisms are required for fault detection, and diagnosis of causes or impacts on running jobs. In addition, if possible, automation of fault handling, such as checkpoint recovery is required.

2.11 Design goals

2.11.1 Ease of use

The user should be able to use OGSA to mask the complexity of the environment if so required. As much as possible, tools, acting in concert with run-time facilities, must manage the environment for the user and provide useful abstractions at the desired level. Tempering this ease-of-use objective is the knowledge that there are "power users" with demanding applications that will require, and demand, the capability to make low-level decisions and to interface with low-level system mechanisms. Therefore it should be possible for end-users to choose the level at which they wish to interact with the system.

2.11.2 Extensibility

It is not possible to predict all of the many and varied needs that users will have. Therefore, mechanism and policy must be realized via extensible and replaceable components, to permit OGSA to evolve over time and allow users to construct their own mechanisms and policies to meet specific needs. Further, the core system components themselves must be extensible and replaceable. This will allow third party (or site local) implementations which provide value-added services to be developed and used.

3 Capabilities

3.1 Overview

The Open Grid Services Architecture (OGSA) facilitates the seamless use and management of distributed, heterogeneous resources. In this architecture, the terms “distributed”, “heterogeneous” and “resources” are used in their broad sense. For example: “distributed” could refer to a spectrum from geographically contiguous resources linked to each other by some connection fabric to global, multi-domain, loosely and intermittently connected resources. “Resources” refers to any artifact, entity and/or knowledge required to complete an operation in/on the system. The utility provided by such an infrastructure is realized as a set of capabilities. Figure 1 shows the logical, abstract, semi-layered representation of some of these capabilities. Three major logical and abstract layers are envisioned in this representation.

The bottom layer of the representation in Figure 1 depicts the *base resources*. In the OGSA context, resources are the virtualized representation (virtualization) of the entities called out in this layer. This allows the concept of resource to be extended to services that don’t virtualize underlying entities but meet the generalized concept of resources mentioned earlier (i.e. every resource is a virtual). Base resources are those resources that are supported by some underlying entities/artifacts that may be physical or virtual and have relevance outside of the OGSA context. Examples of such physical entities can be CPUs, memory, disk etc and of such virtual artifacts can be licenses, data etc. The virtualizations are tightly coupled with the entities that they virtualize and hence the same nomenclature used to name the underlying entities/artifacts is also used to name their virtualizations. These resources are usually locally owned and managed but may be shared remotely. The configuration and customization is also done locally. Since the actual entities/artifacts can change rapidly and can be from multiple sources, these resources can be highly variable in their characteristics, quality of service, version, availability etc. Though, in this discussion, a distinction of base resources is made to tie OGSA concepts to traditional notions of resources, further discussion in this document makes no specific distinction between base resources and other services as resources and only the generalized notion of resources are used.

The second layer represents a higher level of virtualization and logical abstraction when compared to the lower layer. The virtualization and abstraction are directed toward defining a wide variety of *capabilities* that are relevant to OGSA grids. These capabilities can be utilized individually or composed as appropriate to provide the infrastructure required to support higher level applications or "user" domain processes. These set of capabilities as defined in OGSA are relatively invariant and standard. The manner in which these capabilities are realized/implemented and further composed and extended by user domain applications determine the macro Quality of Service of the larger infrastructure as experienced by an end user. *It should be noted that the capabilities shown in the diagram only represent a sample and that a more complete set is discussed in the rest of this chapter.*

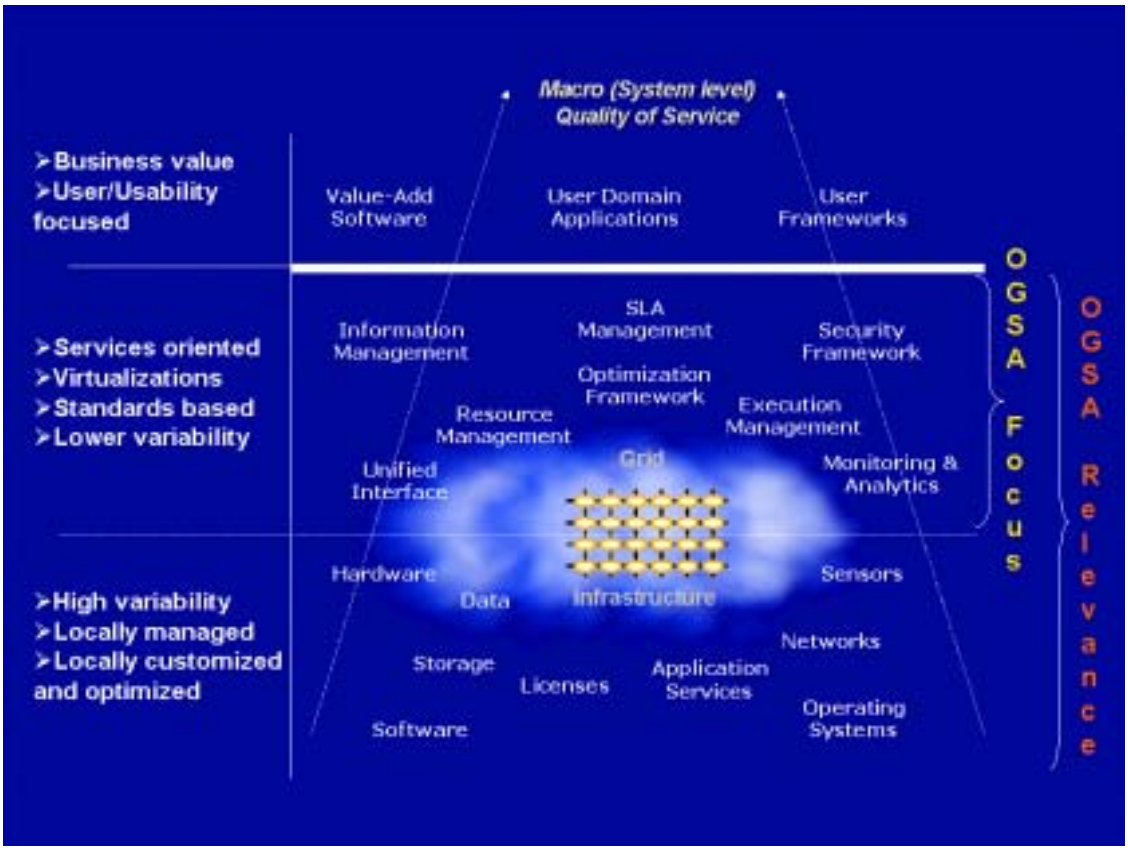

Figure 1: Conceptual view of Grid infrastructures

(*NOTE: The capabilities and resources depicted in the diagram are not an exhaustive list and have been kept minimal for clarity*)

There is no clear boundary between the middle and bottom layers in the logical diagram. This is because OGSA has to understand and comprehend the forms and types of resources and use this understanding to frame the discussion of capabilities in the middle layer. In addition these resources (i.e. virtualizations) will be driven, used and/or managed in realizing many of the capabilities in middle layer. This close relationship in the OGSA discussion is indicated by the finer (thin line) demarcation of the bottom and middle layers in Figure 1. Hence the entities in the lower layer may be regarded as relevant and part of the OGSA discussion.

At the top layer in the logical representation are the applications and other entities that use the OGSA infrastructure to realize user and domain oriented functions and process (like business

processes). These are, for the most part, outside the purview of the OGSA but drive the definition of the architecture from the use cases that the infrastructure should support.

3.1.1 OGSA Framework

The OGSA realizes the logical middle layer in Figure 1 in terms of *services*, the interfaces these services expose, the individual and collective state of these services and the interaction between these services i.e. a *services oriented architecture* (SOA). The OGSA services framework is shown in Figure 2 and Figure 3. The services are built on Web service standards with semantics, additions, extensions and modifications that are relevant to Grids.

A few important points are to be noted.

1. An important motivation for the OGSA is the *composition paradigm* or a 'building block' approach where a set of capabilities or function is built or adapted as required, from minimalist set of initial capabilities, to meet a need. No prior knowledge of this need is assumed. This provides the adaptability, flexibility and robustness to change that is required in the architecture.
2. OGSA represents the services, their interfaces, semantics/behavior and interaction of these services. It should be noted that the software architecture driving the implementation of the internals of these services is outside the OGSA scope.
3. In addition, the architecture is *not*
   - Layered where the implementation of one service is built upon and can only interact with the layer that it is logically dependent on,
   - Object-oriented though many of the concepts may seem to be object-based.

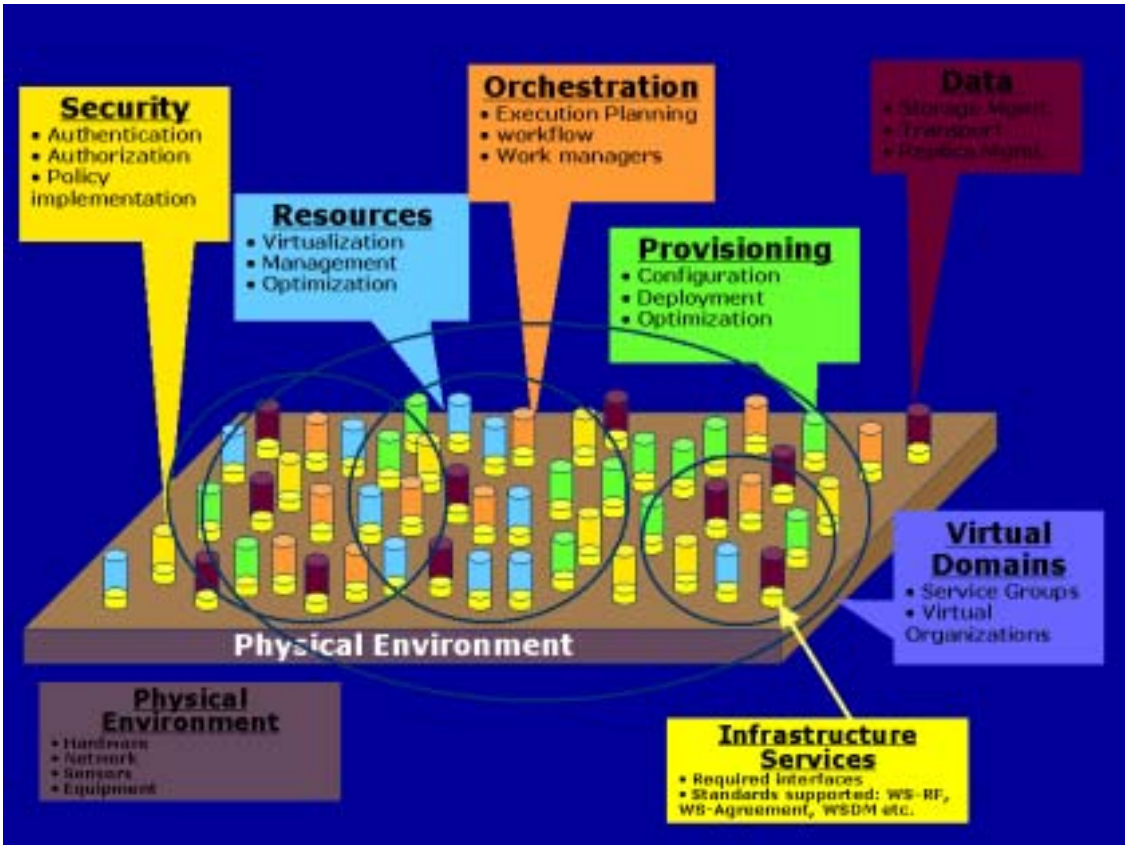

Figure 2: OGSA framework

(NOTE: The capabilities listed here are not exhaustive. Please see other parts of the OGSA document for more details)

The services are loosely coupled peers that either singly or as part of an interacting group of services realize the capabilities of the OGSA through implementation, composition and/or interaction with other services (see Figure 2). For example: to realize the "orchestration" capability is it important that a group of services interact where a set of services in the group drive the orchestration (i.e. the orchestrator) whereas the other services provide the interfaces and mechanisms to be orchestrated (i.e. the ochestratee). A specific service may implement, and/or participate in multiple collections and interactions to realize, different capabilities. On the other hand, it is also not necessary that all services participate to realize a particular capability.

The services may be part of or participate in virtual collections called *virtual domains* either to realize a capability as in a *service group* and/or share a collective context or manageability framework as in *virtual organizations*.

The services implemented as part of OGSA require and assume a *physical environment*. Currently physical environment includes well know physical components and interconnects like computing hardware, networks and could even include physical equipment like telescopes etc.

It is expected that there may be a core set of non-null interfaces, standards and common knowledge/bootstrap that services may need to implement to be part of the OGSA. This set of common implementations and manifestations to support the OGSA is referred to as *infrastructure services (*also known as the Grid fabric). Currently it is believed that some aspects of the Web Services Resource Framework (WS-RF) that is currently under development would be part the

initial Grid fabric. Additional standards that could potentially be a part of the Grid fabric are under development.

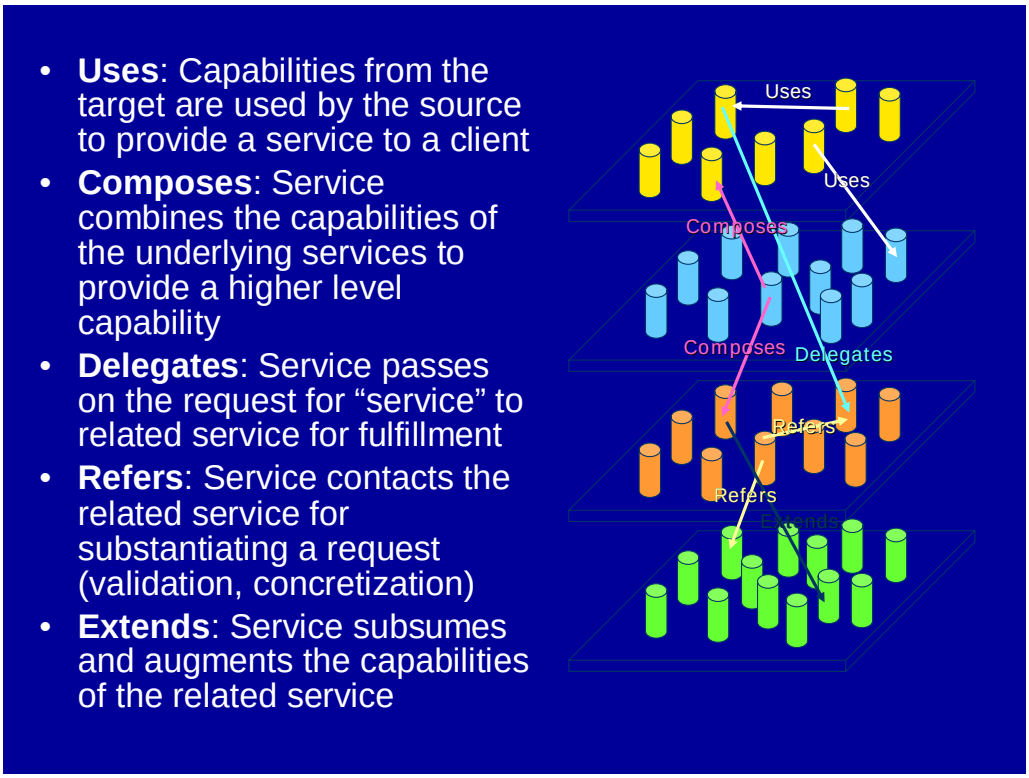

Figure 3: Service Relationships

The OGSA services are related to other services. These *relationships* capture and modulate the varied interactions and dependencies that are important to motivate the compositional paradigm. Aspects of service interactions for other purposes like management/manageability, declarative specification of service profiles etc. can also be modeled with relationships. Some examples of the types of relationships that services may exhibit are show in Figure 3.

3.2 Infrastructure Services

Our goal in defining the Open Grid Services Architecture is to define a coherent and integrated set of components that collectively address the requirements identified above, within the context of a service-oriented architecture. We must necessarily make assumptions about the infrastructure on which we build if we are to make concrete statements about higher-level services. There are many examples of specifications that tried to be too abstract and did not achieve their goals. For example, CORBA 1.1 failed to achieve interoperability because service names depended on implementation. Thus, just as the designs of TCP, DNS, and other higher-level TCP protocols and services are informed by the properties of the IP substrate on which they build, so our design of OGSA is influenced by our choice of underlying mechanisms. In this section we list, and to some extent justify, those choices.

The primary assumption is that work on OGSA both builds on, and is contributing to the development of, the growing collection of technical specifications that form the emerging Web services (WS) architecture [2]. Indeed, OGSA can be viewed as a particular profile for the application of core WS standards. We made this choice because of a strong belief in the merits of

a service-oriented architecture and our belief that the Web services architecture is the most effective route to follow to achieve a broadly-adopted, industry-standard service-oriented rendering of the functionality required for Grid systems.

This choice of Web services as an infrastructure and framework means that we assume that OGSA systems and applications are structured according to service-oriented architecture principles, and that service interfaces are defined by the Web Services Description Languages (WSDL). For now, we assume WSDL v1.1, with a move to WSDL 2.0 planned once that latter specification is finalized. We also assume XML as the lingua franca for description and representation (although recognizing that other representations may be required in some contexts, for example when performance is critical) and SOAP as the primary transport binding for OGSA services. In addition, we seek to develop service definitions that are consistent with the interoperability profiles defined through the WS Interoperability (WS-I) process.

While thus working within a Web services framework, we do not take it as a given that Web services standards as currently defined meet all Grid requirements. In some cases, existing specifications may require modification or extension. Thus, OGSA architects have been involved in the definition of WSDL 2.0 and in the review of WS-Security and related specifications, and we identify in this document other areas in which extensions to existing specifications are desirable. In other cases, Grid requirements motivate the introduction of entirely new service definitions.

One key area in which Grid requirements motivate extensions to existing specifications is security. Security issues arise at various layers of the OGSA infrastructure. We use WS-Security standard protocols to permit OGSA service requests to carry appropriate tokens securely for purposes of authentication, authorization, and message protection. End-to-end message protection is required by some scenarios addressed by the OGSA infrastructure, and thus OGSA must also provide for higher-level protection mechanisms such as XML encryption and digital signatures in addition to, or in place of, point-to-point transport-level security, such as TLS and IPSec. In addition to message-level security, an interoperable and composable infrastructure needs security components to be themselves rendered as services. There are various efforts underway to specify service definitions for these security services. For example, an OGSA authorization service may use the proposed WS-Agreement standard along with evolving OASIS standards SAML and XACML to express security assertions and access control descriptions. When and where appropriate, OGSA will adopt, or define, those security services.

A key area in which Grid requirements have motivated new specifications—specifications that have relevance beyond Grid scenarios—is in the area of state representation and manipulation. In particular, we assume that we have available as building blocks the interfaces and behaviors defined by the WS Resource Framework (WSRF), the recently-proposed refactoring of the Open Grid Services Infrastructure (OGSI) [9]. WSRF defines an approach to modeling, accessing, and managing state; to grouping services; and to expressing faults. These mechanisms (defined after a 18-month design process within the GGF OGSI-WG) all have a fundamental role to play in constructing Grid systems. In addition, WSRF is being strongly considered for use in a variety of resource modeling and systems management standards efforts, such as the OASIS WSDM Technical Committee. Thus, OGSA-related standards that build on WSRF are likely to be well-positioned for composition with efforts from these standards bodies.

Finally, we assume notification/eventing capabilities such as those defined within the recently-proposed WS-Notification specifications. These specifications, derived like WSRF from OGSI,

define notification mechanisms that build on WSRF mechanisms to support subscription to, and subsequent notification of changes to, state components[1].

3.3 Execution Management Services

3.3.1 Objectives

Execution Management Services (OGSA-EMS) are concerned with the problems of service instantiation, and service management. For example, an application needs a cache service. Should it use an existing service or create a new one? If it creates a new service, where should it be placed? How will it be configured? How will adequate resources (memory, disk, CPU) be provided for the cache service? What sort of service agreements can it (the cache service) make? What sort of agreements does it require? Similarly, a user wants to run a legacy program, e.g., BLAST. Where will it run? How are the data files and executables staged to the execution location? What if it fails? Will it be restarted, and if so how?

More concretely EMS addresses problems of placing, "provisioning", and lifetime management of services. This includes (but is not limited to)

- Where can a service execute? What are the locations it can execute because of resource restrictions such as memory, CPU and binary type, available libraries, and available licenses? Given the above, what policy restrictions are in place that may further limit the candidate set of execution locations?
- Where should the service execute? Once it is known where the service can execute, the question is where should it execute? This may involve different selection algorithms that optimize different objective functions or are trying to enforce different policies or service level agreements.
- Prepare the service to execute. Just because a service can execute somewhere does not necessarily mean it can execute there without some setup. Setup could include deployment and configuration of binaries, libraries, staging data, or other operations to prepare the local execution environment to execute the service.
- Get the service executing. Once everything is ready, actually start the service and register it in the appropriate places.
- Manage (monitor, restart, move, etc.). Once the service is started in must be managed and monitored. What if it fails? Or fails to meet its service agreements. Should it be restarted in another location? What about state? Should the state be "checkpointed" periodically to ensure restartability? Is the service participating in some sort of fault-detection and recovery scheme?

These are the major issues to be addressed by EMS. As one can see it covers the gamut of services, and will involve interactions with many other OGSA services that will not be defined in the EMS (e.g., provisioning, logging, registries, security, etc.).

Why is EMS important? Why not just assume a static service set and use registries such as UDDI? We expect Grids to be used in a large number of settings where the set of available resources, and the load presented to those resources, are both highly variable and require high levels of dependability. For example, in an On-Demand computing environment, the set of resources in use by an application may vary over time, and satisfying the application requirements and service level agreements may require temporarily acquiring the use of remote resources. Similarly, to respond to unexpected failures and meet service level guarantees may require finding available resources and restarting services on those resources. The common theme

[1] The WS-Eventing specification provides similar mechanisms, but does not yet connect to WSRF.

is the need dynamically instantiate new service instances in response to application needs and monitor them through their lifetimes.

3.3.2 Solution Philosophy

The solution consists of a set of services that decompose the EMS problem into multiple, replaceable plug points. Different use cases will use different subsets of these services to realize their objectives. In general though one can think of EMS consisting of a supply side and a demand side. Suppliers provide resources – CPU, disk, data, memory, and services. Consumers use those services. One the demand side are tools such as workload management systems, workflow systems, application management systems, and so on. On the supply side are services that manage and supply resources – and enforce policy and service agreements. This can be seen in Figure 4 below that illustrates a generic framework.

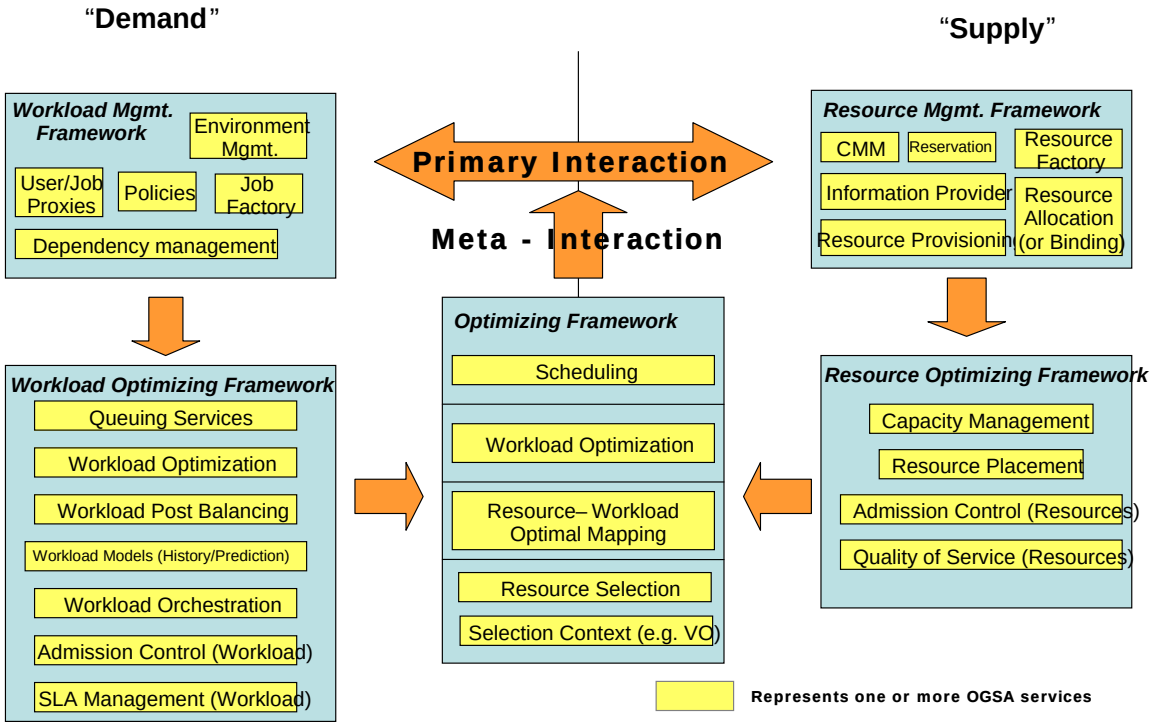

Figure 4 Notional meta-model for EMS divides the world into a supply side and demand side. The supply side represents the resources used – the demand side represents the users of resources. *[This figure needs revision for consistency with the text.]*

EMS services enable applications to have *coordinated* access to underlying resources, regardless of their physical location or access mechanisms. EMS services are the key to making resources easily accessible to end-users, by automatically matching the requirements of a Grid application with the available resources.

EMS consists of a number of services working together. Below we describe these services. Before we proceed though, a few caveats and comments are in order.

First, not all services will be used all of the time. Some grid implementations will not need some of the services – or may bury some of the services within other services and not make them available. In general we have tried to factor the services around those functions that we have seen again and again in different grid implementations – in other words the common functions that not only does one normally need to use – but also that are plug points where one or more different implementations may be desirable,.

Second, this is the first pass at the definitions. An OGSA design team has been formed to hammer out the service in more detail. It was not our objective in this document to completely define the services. Rather it was our intention to identify key components and their higher level interactions.

Third, we want to emphasize that these definitions and services will be applicable to general grid/web service execution – not just the execution of legacy "jobs".

One final assumption. Traditionally, distributed systems have had two or three layer name schemes. Earlier versions of OGSA envisioned using as a base the two-level name scheme of OGSI (GSH and GSR) with an optional "human" based name scheme on top of that. The "human" name spaces were directories for path-oriented naming of services, and registries of metadata for attribute based naming of services. We are no longer assuming the existence or dependence on OGSI. In this document we will assume the existence of a "service handle". A service handle is an abstract name for a service and its associated state (if any). For example, in OGSI the service handle would be a GSH (Grid Service Handle), in Legion a LOID (Legion Object Identifier) is a service handle. We will assume that mechanism exists (defined outside the scope of this document) that binds a service handle to a "service address", where a service address contains protocol-specific information needed to communicate with the service. For example, an EPR [*cite*] is a service address. We will use "SH" to denote a service handle, and "SA" to denote a service address.

3.3.3 Example Scenarios

The best way to understand these services is to see how they are used to realize concrete use cases. We have selected three use cases to demonstrate: a system patch tool, deploying a data caching service, and a legacy application execution.

3.3.4 Case 1 – System Patch Tool

Often operating system patches or library updates need to be applied to a large number of hosts. This can be done in several ways. One commonly used technique is to run a script on each host in a system that copies the appropriate files. These scripts are often initiated using tools like "rsh" or "ssh", and called from shell scripts that iterate over a set of hosts to be patched. Alternatively, hosts may periodically check if they need an update, and if so, run some script to update the OS.

Using EMS this can be done many ways. Suppose the OS version number is a piece of metadata maintained by OS containers and collected by an information service, and that the objective is to patch all operating systems that don't have all the patches. Perhaps the simplest way to approach this problem is to first query information services for a list of containers whose OS version number is below some threshold. Then, instruct a job manager to run the patching service on each container in the list. In this case the job manager does not need to interact with Execution Planning Services because it knows where it wants to run the service. Instead the JM interacts directly with each container – and possibly a deployment and configuration service, to execute the patching service on the container. (The deployment and configuration service may be needed to install the patch service first.)

3.3.5 Case 2 – A Data Cache Service

Imagine a data cache service that caches data (files, executables, database views, etc.) on behalf of a number of clients and maintains some notion of coherence with the primary copies. When a client requests a cache service one can either deliver a handle to an existing cache service – or create a new cache service depending on the location of the client, the location of the existing caches, the load on existing cache's, etc.

Once a decision has been made to instantiate a new cache service an execution planning service would be invoked to determine where the data cache will be placed. The EPS could use CGS to determine where it is possible to run the service – constrained by a notion of locality to the client. Once a location has been selected, the service would be instantiated on a container.

3.3.6 Case 3 – A Legacy Application

Our third example illustrates a rather typical scenario. Suppose a user wants to run a legacy BLAST[2] job. Further, suppose the user is interacting with a portal or queue job manager. There are four basic phases in getting the BLAST job started:

1. Job definition phase. What are the input files? What are the service level requirements? E.g., job must complete by noon tomorrow. What account will be billed for the job? Etc.
2. Discover the resources available and select the resources required to execute the service.
3. Enact the schedule and all that may be involved, e.g., provisioning of resources, accounting, etc.
4. Monitor the job through its lifetime(s). Depending on the service level agreements the job may need to be restarted if it fails to complete for any reason.

To realize this case using EMS the job manager creates a new legacy job with the appropriate job description (in JSDL). The JM then calls an EPS to get a schedule. The EPS in turn calls a CSG, which calls information services to determine where the job can be executed based on binary availability and policy settings. The EPS selects a service container after first checking with the service container that the information is accurate. The EPS returns the schedule to the JM. The JM then interacts (if necessary) with reservation and deployment and configuration services to set up the job execution environment. This may involve interaction with the data container as well. The service container is invoked to start the job. Logging services are used for accounting and audit trail. Assuming the job terminates normally the job manager is notified by the container. If it terminates abnormally, the whole cycle may repeat again.

[2] BLAST is one of the most commonly executed bio-informatics applications. It compares a DNA or protein sequence against a library and generates similarity scores,

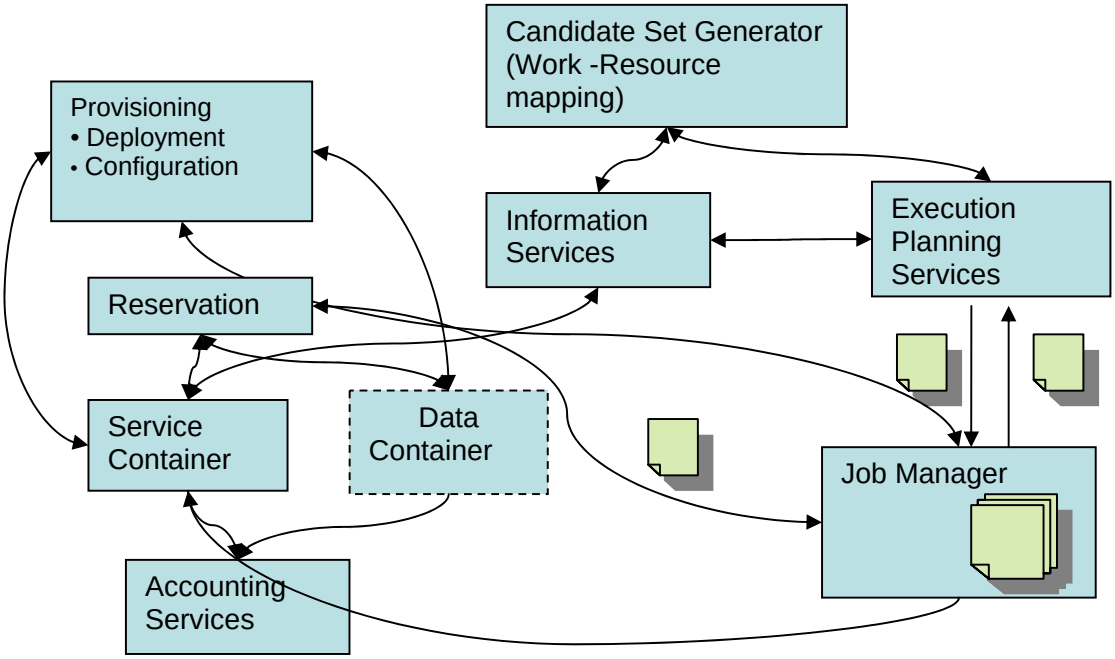

Figure 5 Interactions of EMS services to execute a legacy BLAST job

3.3.7 EMS Services

There are three broad classes of EMS services: 1) resources that model processing, storage, executables, resource management, and provisioning; 2) job management and monitoring services; and 3) resource selection services that collectively decide where to execute a service.

There are several other services outside the scope of this document that we assume are available. The include data management services (currently being considered in the OGSA-data design team); security services (currently being considered in the OGSA-security design team); and logging services.

3.3.8 Resources

3.3.8.1 Service Container:

A Service Container, hereafter just a container, "contains" running services, whether they are "jobs" (described later) or a running grid service instance. A container may for example be a queuing service, a Unix host, a J2EE hosting environment, or a collection of containers (a façade or a VO of job containers). Containers have attributes (metadata) that describe both static information such as what kind of executables they can take, e.g., OS version, libraries installed, policies in place, security environment/QOS, etc., as well as dynamic information such as load, QOS issues, etc.

A service container implements the interface of a WSDM managed resource. Extended interfaces that provide additional services beyond the basic service container are expected.

Containers will have various relationships to other resources that will be exposed to clients. For example, a container may have a "compatibility" relationship with data containers that indicates that services running "in" a container can access persistent data "in" a particular data container.

Similarly other managed resources might be a deployed operating system, a physical network, etc.

The relationships with other resources are critical. We expect that sets of managed resources will be composed into higher level services – for example a "container" may be extended to a host-container for example that includes a "container", a "PSHS", an OS, etc.

Similarly we expect containers to interact with reservation services, logging services, accounting services, information services, job management services, and provisioning services.

3.3.8.2 Persistent State Handle Service (PSHS)

A PSHS keeps track of the "location' of persistent state for services. It may be implemented many different ways, by a file system, by database, by a hierarchical storage system, etc. A PSHS will have methods to get a "handle" to persistent state that it is managing. The form of the handle will depend how the state is actually stored. A persistent address may be a path name in a file system or a primary key value in a database. The key idea is that the persistent address can be used to directly access the data.

A PSHS implements the interface of a WSDM managed resource. Extended interfaces that provide additional services beyond the basic data container are expected.

PSHS's will also have methods for managing their contained state, including passing it to other data containers. This will facilitate both migration and replication.

Another way to think about a PSHS is that it is a meta-data repository that tells you how to get to the data really fast using native mechanism, e.g., a mount point, a db key, a path.

Note that PSHS's are not data services. Rather they are how we keep track of where the state of a service is kept so that we can get to it quickly if necessary.

3.3.9 Job Management

3.3.9.1 Job

A job is a service (named by a distinct SH) and is created at the instant requested even though no resources have been committed. The job encapsulates all there is to know about a particular instance of a running application or service. Jobs are NOT workflows or array jobs. A job is the smallest unit that is managed and has a distinct service handle. The job is not necessarily the same as the actual running application. Instead the job keeps track of job state (started, suspended, restarted, terminated, completed, etc.), resource commitments and agreements, job requirements, and so on. Many of these are stored in a "job document".

A *job document* describes the state of the job, e.g., the agreements that have been acquired, the JSDL, job status, meta data about the user (credentials etc), how many times it has been started. By state we do not mean for example the internal memory of a blast job. The job document is encapsulated by a job and exposed as service data of the job. The logical view is of one large document that consists of possibly many subdocuments. These sub documents can be retrieved independently. The organization of the sub-documents will be subject to further specification.

3.3.9.2 Job Manager

The job manager manages jobs. It encapsulates all of the aspects of executing a job or a set of jobs from start to finish. A job manager may manage a single job instance or a set of job instances. A set of job instances may be structured (e.g., a workflow or dependence graph) or unstructured (e.g., an array of non-interacting jobs). Similarly it may be a portal that interacts with users and manages jobs on the user's behalf.

The job manager will likely interact with execution planning services, the deployment and configuration system, containers, and monitoring services. Further, it may deal with failures and restarts, it may schedule them to resources, it may collect agreements and reservations.

The job manager is very likely to implement the interface of a WSDM collection, which is a collection of manageable entities. A WSDM collection can expose as its methods some of the methods exposed by the members of its collection.

The manager is responsible for orchestrating the set of services to start a job or set of jobs, e.g., negotiating agreements, interacting with containers, monitoring and logging services, etc. It may also aggregate job service data from underlying "related" job instances.

Examples of job managers:

- A "queue" that accepts "jobs", prioritizes them, and distributes them out to different resources for computation. (Similar to "JobQueue" in [], or Condor []). The job manager would track the jobs, may have QoS facilities, prioritize jobs, have a maximum number of outstanding jobs, and a set of service containers in which it places jobs.
- A portal that interacts with end-users to collect job data and requirements, schedule those jobs, and return the results.
- A workflow manager that receives a set of job descriptions, QoS requirements, their dependence relationships, and initial data sets (think of it as a data flow graph with an initial marking), and schedules and manages the workflow to completion – perhaps even through a number of failures. (In this case a node could be another workflow job manager). (Similar in concept to parts of DAGman [10]).
- An array job manager that takes a set of identical jobs with slightly different parameters and manages them through completion. (e.g., Nimrod)

3.3.10 Selection Services

3.3.10.1 Execution Planning Services (EPS)

An EPS is a service that builds mappings called "schedules" between jobs and resources, e.g., containers and vaults. A schedule is a mapping (relation) between grid services and resources, with possibly, time constraints. A schedule can be extended with a list of alternative "schedule delta's" that basically say, if this part of the schedule fails, try this one instead".

An EPS will typically be attempting to optimize some objective function such as execution time, cost, reliability, etc. An EPS will not enact the schedule, it will simply generate it. An EPS will likely use information services and candidate set generators. For example, first call a CSG to get a set of resources, then get more current information on those resources from an information service, then execute the optimization function to build the schedule.

3.3.10.2 Candidate set generator (CSG)

The basic idea is quite simple – determine the set of resources on which a job or service can execute, i.e, where is it possible to execute, not where will it execute. This will involve issues such as what binaries are available, any special requirements of the resource that the application might have (e.g., 4GB memory and 40GB scratch, xyz library installed), security and trust issues (I won't let my job run on a resource unless it is certified Grade A plus by the pure computing association, or they won't let me run there until my binary is certified "safe", or will they accept my credit card), and so on.

A CSG generates a set of containers (really their SH's) where it is possible to run a service named by a SH. The set of resources to search over may be either a default for the particular service, or be passed in as a parameter.

We expect CSG's to be primarily called by EPS's – or other services such as job managers that are performing an EPS function. We expect CSG's to use information services, access jobs to acquire appropriate pieces of the job document, and interact with provisioning and container services to determine if it is possible to configure a container to execute a particular service.

3.3.10.3 Reservation services

Reservation services manage reservations for resources, interact with accounting services (there may be a charge for making a reservation), revoke reservations, etc. This may not be a separate service, rather an interface to get and manage reservations from containers and other resources. The reservation itself is likely to be a document that is signed.

A reservation service presents a common interface to all varieties of reservable resources on the Grid. Reservable resources could include (but are not limited to) computing resources such as CPUs and memory, graphics pipes for visualization, storage space, network bandwidth, special purpose instruments (e.g. radio telescope), etc.

A Reservation could also be an aggregation of a group of "lower level" reservations as might be negotiated and "resold" by a broker of resources.

Reservation services will generally be used by many different services. The Job Manager might create reservations for the groups of jobs which are being managed, or an EPS might use reservations in order to guarantee the execution plan for a particular job. It could also be the case that the creation of reservations will be associated with the provisioning step for a job.

3.3.11 Interactions with the rest of OGSA

This section details the interactions between the EMS and other parts of OGSA.

3.3.11.1 Deployment & configuration service

Often before a job or service can execute in a container the service container and/or data container must first be configured or provisioned with additional resources. For example, before I can run BLAST on a host, I must ensure that the BLAST executable and all of its myriad configuration files are accessible to the host. A more in depth example is the configuration of a complex application and installation of appropriate data bases, or installing Linux on a host as a first step to using the host as a compute resource.

3.3.11.2 Information Service

The basic idea is simple; information services are databases of attribute metadata about resources. Clients want to query the information service to retrieve the names (SH's) of services that match some criteria, or perhaps just get related attribute data. Each information service will have a domain or virtual organization over which it maintains information. How it gets the information is unspecified, although we expect "freshness" to be an attribute on data. Similar to MDAS services in Globus [11] and collections in Legion [12].

3.3.11.3 Monitoring

Simply starting something up is often insufficient. Applications (which may include many different services/components) often need to be continuously monitored, both for fault-tolerance reasons as well as QOS reasons. For example, the conditions on some hosts that caused the scheduler to select it may have changed, possibly indicating that the task (service instance) needs to be rescheduled.

3.3.11.4 Fault-Detection and Recovery Services

These may or may not be a part of monitoring, and may include support for managing both simple schemes for stateless functions that allow trading off performance and resource usage, to slightly more complex schemes that manage checkpoint and recovery of single thread (process) jobs, to more complex schemes that manage applications with distributed state, e.g., MPI jobs.

3.3.11.5 Auditing/billing/logging services

Auditing, logging, and billing services are critical for success of OGSA outside of academia and the government. This will include the ability for schedulers to interact with resources to establish prices, as well as for resources to interact with accounting and billing services. Logging is the basis of the whole chain.

Metering is using the log to keep track of resource usage.

Auditing is using the log in persistent fashion, possibly non-repudiation as well.

Billing is yet another service, not defined by OGSA, that may use auditing, and/or, metering logs and other data to generate bills, or charge backs.

3.3.11.6 Accounting

Like a credit card, some resources (e.g., containers) need to see if the user has enough to pay. The scheduler may need to interact with this – as may the resources, e.g., containers.

3.4 Data Services

3.4.1 Objectives

The OGSA data services are concerned with the movement, access and update of data resources. They provide the capabilities to move data to where it is needed, manage replicated copies, run queries and updates, and transform data into new formats. They also provide the capabilities necessary to manage the metadata that describes OGSA services, in particular the provenance of the data itself.

For example, suppose an execution service needs to access data that is stored elsewhere. Does it use data services to access the data remotely or to stage a copy to the local machine? Does it cache some of the data locally? Is the data available in multiple locations? What policy limitations or service guarantees does the data service provide?

Conversely, suppose that a data service wishes to provide a virtual schema of data stored in several different places. How should queries against the virtual schema be mapped to the underlying resources? Where should any joins or data transformations be executed? To where should the data be delivered? What quality of service can be guaranteed?

An OGSA design team has been formed to design the data architecture and services in more detail than given here. In addition, several working groups are working on the detailed design of some of the data services. This document is not intended to completely define the services. Rather it was our intention to identify key capabilities and their higher level interactions, including interactions with the rest of OGSA.

With the exception of certain classes of metadata, the data capabilities do not specify the meaning of any particular data. Some other OGSA services may use the data services and add meaning of their own. The information services are an example of this.

3.4.2 Models

3.4.2.1 Types of Data Resource

A data resource is any entity that can act as a source or sink of data. The heterogonous nature of the Grid means that many different types of data must be supported. These include (but are not limited to):

- *Flat Files*. The simplest form of data is a file with application-specific structure. For OGSA these files are opaque. Some file formats support database-like queries. Examples include comma-separated values, which can be queried like relational tables, and XML files, which can be queried using XMLQuery. The data access services support these and can also be extended to support specialised queries over new file formats.
- *Streams*. Potentially infinite sequences of data values are called streams. The data access services support queries and transformations over streams.
- *DBMS*. Several kinds of database management systems may be part of Grids. These include relational, XML, and object-oriented databases, among others.
- *Catalogues*. A catalogue structures and tracks other data services. A simple example of a catalogue is a directory, which lists a set of files. Nested directories are equivalent to a hierarchic namespace.
- *Derivations*. Some data is the result of asynchronous queries or transformations on other data. These derivations are managed like finite streams rather than single items.

Data Services themselves can be data resources for other services, as can sensor devices and programs that generate data.

3.4.2.2 Example Scenarios

Data management capabilities are fundamental to the Grid. This section describes a very few examples of how they can be used to support a range of activities.

- *Remote access.* The simplest use of the OGSA data services is to access remote data resources across the Grid. The services hide the communication from the client; if necessary they can also hide the exact location of the remote data. An optimisation of this is to cache some of the data in a local resource.
- *Staging*. When jobs are executed on a remote resource, the data services are often used to move input data to that resource ready for the job to run, and then to move the result to an appropriate place.
- *Replication.* To improve reliability and to reduce access times, the same data can be stored in multiple locations across the Grid.
- *Federation*. The OGSA data services allow the creation of a virtual data resource that incorporates data from multiple data sources that are created and maintained separately. When a client queries the virtual resource, the query is compiled into sub-queries and operations that extract the appropriate information from the underlying federated resources and return it in the appropriate format.
- *Derivation.* The OGSA data services support the automatic generation of one data service from another.
- *Metadata.* Data that describes services or other data is fundamental to the Grid. In the simplest form, this is just another use of the OGSA data services. In addition, OGSA provides support for maintaining links between data and metadata.

3.4.2.3 Supply & Demand Meta Model

The solution consists of a set of services that decompose the data management problem. Different use cases will use different subsets of these services to realize their objectives. As with other parts of OGSA, the data service architecture can be seen as composed of suppliers and consumers. Suppliers provide resources – files, databases, streams and services. On the demand side are services that virtualise data resources, manage distributed queries, manage service level agreements, and so on. On the supply side are services that cache, replicate and manage data, and enforce policy and service agreements. This can be seen in

Figure 6 below, which illustrates a generic framework.

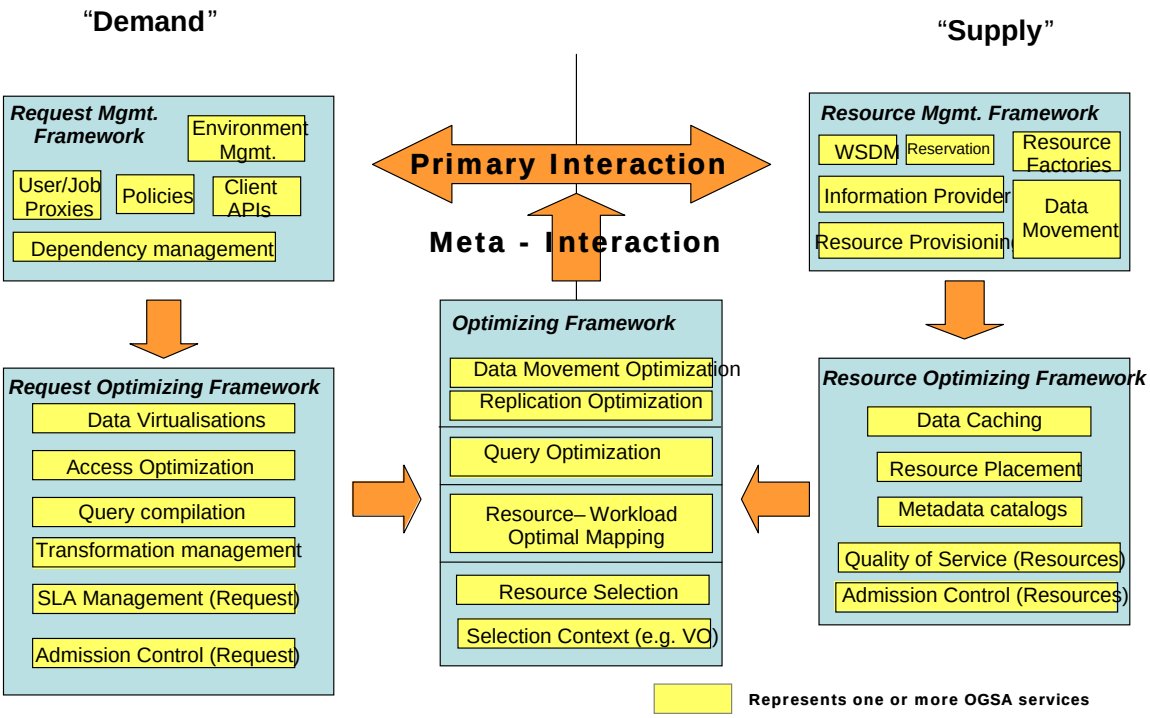

Figure 6 Notional meta-model for Data divides the world into a supply side and demand side. The supply side represents the resources available for use; the demand side represents the users of resources.

3.4.3 Functional Capabilities

This section describes the functional capabilities provided by the OGSA data services. Different subsets of the services are needed to implement the different capabilities. Applications and infrastructures are free to provide only some of these capabilities.

3.4.3.1 Transparency, Virtualization and Layered Interfaces

A distributed system may contain a variety of data resources. These resources may use different models to structure the data, different physical media to store it, different software systems to manage it, different schema to describe it, and different protocols and interfaces to access it. The data may be stored locally or remotely; may be unique or replicated; may be materialized or derived on demand. The OGSA data services can define virtualisations over these resources.

Virtualisations are abstract views that hide these distinctions and allow the data to be manipulated without regard for them.

Conversely, although the data services allow the clients to ignore these distinctions, some clients may prefer to exploit them. For example, a client may wish to make use of a particular query language for a given database, or to specify the location of the particular data resource to use. One client may require native access to the data, while another needs to tune the performance parameters of the data resource. To support such clients, the OGSA data services allow clients to bypass the virtualisation interfaces and access the resource-specific interfaces directly. This layered approach allows clients to choose the combination of power and abstraction that suits them best.

3.4.3.2 Client APIs

Many users will wish to use the OGSA data services with legacy applications that use existing APIs such as NFS, CIFS, JDBC, ODBC, ADO, POSIX IO or XQuery. Often it will be too expensive or impractical to rewrite the clients to use new interfaces. However, the OGSA services can be easily wrapped with a veneer that emulates these existing APIs, although OGSA itself will not define such wrappers.

Typically these wrappers will map the operations of the existing APIs to the corresponding messages to send to the OGSA data services. Therefore they may provide access to only a subset of the full OGSA functionality. The extent of the OGSA functionality that they make available will depend on the scope of the API concerned.

3.4.3.3 Extensibility

It is not possible to predict all the data resources with which the OGSA data services will be used. In any case, new data resources will be created in the future. The layered architecture described above allows the addition of new data services that access new types of resources.

Similarly, it is not possible to anticipate all the operations that may be needed on a given resource. The layered architecture supports services that provide additional operations beyond those specified in the OGSA document. The virtualisation layer gives clients the option of ignoring these extensions, while those clients that require them can bypass the virtualisation layer as necessary.

3.4.3.4 Data Management

The OGSA data services offer reliable data transfer from one location to another. This may be to create a copy of the original or to migrate the original completely. Data may be cached at a given location to avoid unnecessary additional transfers. Data caches can be configured in terms of, for example, lifetime and update consistency. Data may also be replicated between multiple copies, thus increasing availability through redundancy.

Services for individual users allow them to upload and manage their own data with the above facilities. The security settings control the access permitted to other users.

3.4.3.5 Queries

The OGSA data access services provide mechanisms for querying data resources. In simple cases these may run an SQL query over a relational database, an XML query over an XML database, or a regular expression over a text file. Other services may implement text mining over a set of documents, or distributed queries over federated databases.

Synchronous queries return the data in the response to a request, while asynchronous queries expose the derived data as new resources. Services may also deliver the results of a query to a specified set of other services.

Distributed query processors analyse each query and create sub-queries to be run on the resources that comprise the federation. They may also determine where intermediate processing is done, in order to minimise network traffic. As such they have some similarity to workflow enactment engines.

3.4.3.6 Transformation

Data services may themselves transform data. For example, they may convert data from one format to another, or filter it, before moving it. They may support stored procedures that execute within the service, making the service a form of container. These transformations may be instigated explicitly by certain operations, or they may be programmed to be triggered automatically in response to certain conditions.

3.4.3.7 Data Update

OGSA data services provide a range of mechanisms for updating data resources, depending on the semantics of the data resource. For catalogues, the operations include creation, renaming and deletion. For structured files and databases the operations include the update of entries. For streams and other files the operations are largely limited to appending new data.

When a data resource has replicated versions or is the source for derived data services, the updates may be propagated to the replicated or derived versions. In this case, and in the case where several clients are updating the same data resource, the services may implement various forms of consistency maintenance, e.g. to ensure that a client always sees the results of its own updates in any queries that it issues itself.

Update operations may be part of transactions. Transaction support is provided by other OGSA services.

3.4.3.8 Security Mapping Extensions

Database management systems often implement sophisticated security mechanisms. Some of these provide a large range of possible operations and access control at the level of individual elements. Therefore the OGSA data services support extensions to the standard OGSA security infrastructure to allow users, operators and applications to access the greater control provided by such systems.

3.4.3.9 Data Resource Configuration

Data resources often provide sophisticated configuration options. These can be made available to clients via the OGSA data services. In addition, the services may provide additional operations for configuring the virtualisation of the resource provided by the service. For example, a relational data service might allow for specific tables from an underlying database resource to be made part of the data service's data virtualization, or for views on the underlying database to be made available as tables within that virtualization.

3.4.3.10 Metadata

Metadata services are data services that store metadata about services. OGSA provides support for maintaining links between OGSA services and the metadata that describes them. This support includes provision for maintaining the consistency of the metadata.

The metadata for data services may include information about the structure of the data, including references to the schemas that describe the data. For some services this is not practical, as the data resources include many schemas that are modified frequently, and in these cases schema information will be provided by the services themselves.

3.4.3.11 Provenance

Metadata for data services may also include information about the provenance of the data. This may be at the level of the whole resource or of its component parts, sometimes to the level of individual elements. This in turn requires the services or other processes that generate the data to also maintain the consistency of the metadata. Complete provenance information can allow the data to be reconstructed by following the workflow that originally created it.

3.4.4 Properties

3.4.4.1 Quality of Service

The OGSA data services can implement various levels of QoS, such as guaranteed delivery and referential integrity.

3.4.4.2 Coherency

When data is replicated, cached or derived, a range of coherency operations are available.

3.4.4.3 Performance

The OGSA data services are designed to minimize the copying and movement of data to the minimum. This is a key factor in the overall performance of the Grid.

The data transfer services use monitoring information such as bandwidth, utilisation patterns and packet size. This enables them to choose the best approach for moving a given data set to suit the agreed quality of service.

3.4.4.4 Legal and Ethical Restrictions

The OGSA data services may have to operate within an environment where a variety of legal and ethical policies affect their operation. For example, some policies may restrict the entities that can access personal data and limit the operations that they can perform (confidentiality). Privacy concerns may limit the queries that can be made about individuals. In some cases the policies may permit other queries that return results about a group as a whole, such as average income or total salary.

Much data is covered by copyright limitations. This restricts the copies that can be created of the data. In the European Union, the similar “Database right” applies specifically to databases. The security mechanisms provided by OGSA allow these restrictions to be specified, but care must be taken when using the data services.

3.4.5 Interactions with the rest of OGSA

This section details the interactions between the data services and the other parts of OGSA.

3.4.5.1 Transactions

Data services are the classic example for transactions, and many data resources provide independent transactional support. OGSA allows data services to be part of transactions. Support is provided for conventional ACID transactions and for two phase commit. In addition, “time warp” co-ordination can also be supported.

3.4.5.2 Logging

In addition to the usual OGSA logging capabilities, the data services use the logging services to support (non-local) logging. Conversely, the logging services may use the data services to store the logs.

3.4.5.3 Execution Management Services

Data services have a close relationship with the Execution Management Services (OGSA-EMS). OGSA-EMS uses the data services in order to stage data where it is needed. The EMS execution planner service has to take into account the costs of accessing the data from the candidate computational resources.

3.4.5.4 Workflow

The data access services are closely entwined with OGSA workflow. Workflows can instruct the data access services to send query results to third parties and to apply transformations as the data is moved. This requires intimate knowledge of the capabilities of the resource by the workflow services. Also, a call to a distributed query service may cause a variety of data movements and operations on other machines. The data service can provide information to the workflow manager to ensure that the workflow enactment takes full account of these effects on the network and other resources.

3.4.5.5 Provisioning

In addition to the provisioning of storage space and of services themselves, data services also require provisioning support for uploading data sets to data resources. Some may also require support for uploading filters and cutters to a given data service.

3.4.5.6 Resource Reservation

Data services may need to reserve certain resources in order to operate. As examples, a file transfer will require storage space and network bandwidth, while a distributed query system may additionally require compute power in order to perform join operations.

3.4.5.7 Discovery

The data services may use the discovery services not just for registering services themselves, but also for registering the data sets that are stored by those services. They may also register the locations of schema definitions.

3.4.5.8 Security

The security services support the mapping of OGSA identities and roles to resource-specific identities and roles. The virtual organisation management services similarly provide control of these mappings. The security services also provide support for checking the integrity of data transfers.

3.4.5.9 Network management

Network management can be crucial when planning the transfer of large amounts of data. It may be necessary to configure the network parameters to suit the amount of data to be transferred and the time constraints specified on those transfers.

3.4.5.10 Naming

The naming of data is a key feature of the OGSA data services. These use the OGSA three-level naming schemes. The physical name (WSRF end-point reference) includes the address where a

data set can be found, which may use a format specific to the particular resource. The logical name identifies the data set regardless of its actual location (or locations, if it is replicated). The context-mapped name is a user-friendly short name that requires some context to disambiguate.

For example, in a replicated file system a physical name may specify the location of a file on a particular machine. The logical name will identify the file in a location-independent but universal way, e.g. by including a UUID. The context-mapped name may be a user-friendly short file name that can be disambiguated by referring to the context in which it is used.

3.4.5.11 Notification

A particular use of the notification service is to externalise database triggers. Database management systems often provide a mechanism that specifies actions to be taken when certain conditions (triggers) are met. OGSA can easily cause these triggers to invoke the notification services.

In addition, some implementations of the notification services may use the data services to store a history of Grid activity and status that can be queried by a range of clients.

3.4.5.12 WS-Agreement

As with other OGSA services, the data services use WS-Agreement to negotiate and record quality of service agreements and payment agreements.

3.5 Resource Management Services

The following contents have been developed by the CMM-WG; a more detailed description can be found in their informational document "Resource Management in OGSA".

3.5.1 Objectives

Resource management performs several forms of management on resources in a Grid. In an OGSA Grid there are three types of management that involve resources:

- Management of the resources themselves (e.g., rebooting a host, or setting VLANs on a network switch)
- Management of the Grid resources (e.g., resource reservation, monitoring and control, etc.)
- Management of the OGSA infrastructure, which is itself composed of resources (e.g., monitoring a registry service)

3.5.2 Model

Different types of interfaces realize the different forms of management in an OGSA Grid. These interfaces can be categorized into three levels, shown in the middle column of Table 3, and also on the right in Figure 7.

Table 3: Relationships between types of management and interfaces

Type of management	Level of interface	Interface
Management of the resources themselves	Resource level	CIM, SNMP, etc.
	Infrastructure level	WSRF, WSDM, etc.
Resource management on the Grid	OGSA functions level	Functional interfaces

Management of OGSA infrastructure		Specific manageability interfaces

A detailed description of each level and its interfaces is given below. Note that the descriptions focus on the manageability interfaces, *not* on the locus of implementation (e.g., on the services that implement them). Also note that a service may implement multiple interfaces (which are possibly unrelated in terms of functionality), and that a service may be separated from the functionality that it represents (e.g., a manageability provider for a resource that is separate from this resource). Therefore a description based on services would be imprecise, and a description based on interfaces is chosen instead.

In Figure 7, the OGSA capabilities cover all levels, extending to capabilities in the resources that are needed to implement these OGSA capabilities. The interfaces are shown as small circles.

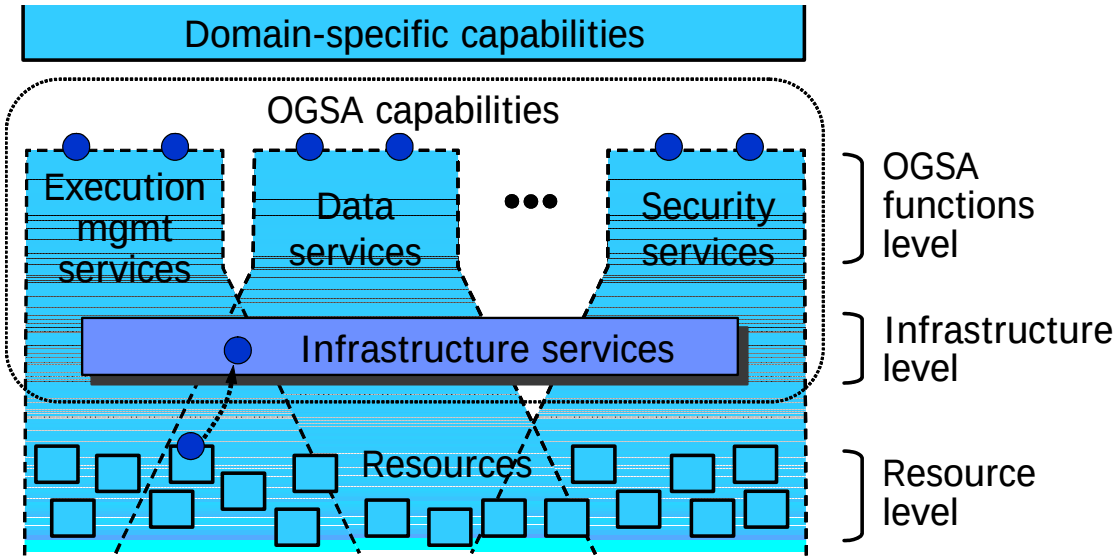

Figure 7 Levels of management in OGSA

At the resource level, the resources are managed directly through their native manageability interfaces (for discrete resources, these are usually SNMP, CIM/WBEM, JMX, or proprietary interfaces). Management at this level involves *monitoring* (i.e. obtaining the state of the resource, which includes events), *setup and control* (i.e. setting the state of the resource), and *discovery*.

The infrastructure level provides the base management behavior of resources, forming the basis for both manageability and management in an OGSA environment. Standardization of this base management behavior is required in order to integrate the vast number and types of resources—and the more limited set of resource managers—that are introduced by multiple suppliers. The infrastructure level provides:

- The *base manageability model*, which represents resources as services and allows resources in OGSA to be manipulated through the standard Web services means for discovery, access, etc. This model allows the resources to become manageable, at least to a minimum degree, by enabling discovery, termination, introspection, monitoring, etc.

- A *generic manageability interface* that is common to all services implementing OGSA capabilities. This manageability interface has functionality such as introspection, monitoring, and creation and destruction of service instances.

At the OGSA functions level there are two types of management interfaces, denoted by the two circles on the top of each of the capabilities shown in Figure 7:

- *Functional interface*: Some common OGSA capabilities (such as job management) are a form of resource management. Services that provide these capabilities expose them through functional interfaces.
- *Manageability interface*: Each capability has a specific manageability interface through which the capability is managed (e.g., monitoring of registries, monitoring of a job manager, etc.). This interface could extend the generic manageability interface, adding any manageability interfaces that are specific to the management of this capability.

3.5.3 Functional Capabilities

The functionality at the infrastructure level will be provided by OASIS WSDM, which is expected to become a key standard for manageability across the IT landscape. WSDM is developing separate documents to address management *of* Web services (MOWS) and Management *using* Web services (MUWS). MUWS will provide the base manageability model, i.e., how to represent resources as services, plus basic functionality that is common to the OGSA capabilities, such as state representation and operations, and relationships among resources. MOWS will provide the generic manageability interface to manage the services on an OGSA Grid.

At the OGSA functions level, the resource management capability includes (but is not limited to) typical distributed resource management (DRM) activities and IT systems management activities. The following are examples of functionalities that the resource management capability in OGSA comprises:

- Resource reservation
- Monitoring and control
- VO management
- Security management
- Problem determination and fault management
- Service groups and discovery services
- Metering
- Deployment
- Discovery

3.5.4 Properties

- **Scalability**: Management architecture needs to scale to potentially thousands of resources. Management needs to be done in a hierarchical and/or peer-to-peer (federated/collaborative) fashion to achieve this scalability, so OGSA should allow these forms of management.
- **Interoperability**: Management architecture must be able to span software, hardware and service boundaries, e.g., across the boundaries between different products, so standardized and broad interoperability is essential to avoid “stovepipes.”
- **Security**: There are two security aspects in management:
  - Management *of* security: the management of the security infrastructure, including the management of authentication, authorization, access control, VOs and access policies.

    - Secure management: using the security mechanisms on management tasks. Management should be able to ensure its own integrity and to follow access control policies of the owners of resources and VOs.
- **Reliability**: A management architecture should not force a single point of failure. For purposes of reliability, a resource may be virtualized by multiple services exposing a single URL as the management endpoint. In such situations, the system that provides manageability capabilities must be aware that for certain queries, such as metrics, the manageability provider must aggregate the results from the multiple services that virtualize that single resource.
- **Policy**: Management must be able to enforce policy assertions that are put in place to support requirements and capabilities such as authentication scheme, transport protocol selection, QoS metrics, privacy policy, etc.

3.5.5 Interactions with other OGSA services

The services in the resource management capability are an enabler for the services in other OGSA capabilities. Thus services in most OGSA capabilities, especially Execution Services, Data Services, and Self-Management Services, will use functionality provided by the resource management capability.

The resource management capability uses basic services in other OGSA capabilities, such as registries (used for resource discovery).

3.6 Security Services

This section describes objectives, models, functional capabilities, and properties of the security services, and discusses the interactions with the rest of OGSA.

3.6.1 Objectives

OGSA Security Services are to facilitate the enforcement of the security-related policy within a (Virtual) Organization.

In general, the purpose of the enforcement of security policy is to ensure that the higher-level business objectives can be met. This is often a delicate balance, as there is an increased cost associated with increased policy enforcement, and an increased loss exposure with less stringent enforcement, while potential profits or rewards may be adversely affected by the increased complexity and inflexibility of stricter policy requirements. Note that in some cases, legislative rules and regulations mandate conformance to associated security policy.

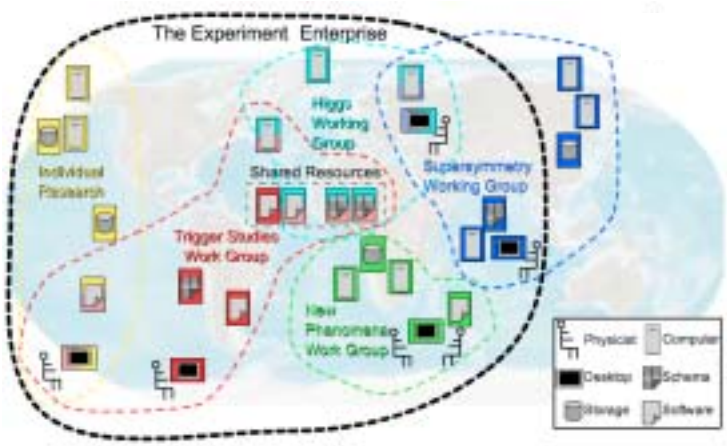

Figure 8. Cross-Organizational Collaborations

The property that Grid-specific applications span multiple administrative domains (see Figure 8) implies that each of these domains will have its own business objectives to meet, which translates in an enforced security policy by each domain that can differ greatly in complexity and strictness between domains. Note, that all interactions associated with a thread of work in a Grid application have to adhere to the domain-locally enforced policies as well as the policies established for the VO, i.e. the cross-organizational (business) agreement.

To meet the business and associated security objectives, the security policy can for example specify the enforcement of:

- Message integrity and confidentiality
- Authentication of interacting entities
- Minimum authentication strength
- Secure logging and audit
- Separation of responsibilities
- Intrusion and extrusion detection
- Authorization policy checks
- Least privilege operations
- Mandatory access control mechanisms
- Discretionary access control mechanisms
- Trust and Assurance level of the environment
- Application Isolation
- Avoidance of DoS attacks
- Redundancy
- Training
- …

OGSA security architectural components must support, integrate, and unify popular security models, mechanisms, protocols, platforms, and technologies in a way that enables a variety of systems to interoperate securely. The components must be able to support integrating with

existing security architectures and models across platforms and hosting environments. This means that the architecture must be *implementation agnostic*, so that it can be instantiated in terms of any existing security mechanisms (e.g., Kerberos, PKI); *extensible*, so that it can incorporate new security services as they become available; and *integratable* with existing security services. Also, services that traverse multiple domains and hosting environments need to be able to interact with each other, thus introducing the need for interoperability at multiple levels: protocol, policies and identity. Also, in some cases can make it impossible to establish trust relationships among sites prior to application execution. Given that the participating domains may have different security infrastructures (e.g., Kerberos, PKI) it is necessary to realize the required trust relationships through some form of federation among the security mechanisms.

A preliminary OGSA Security Architecture document, developed within GGF's OGSA-Sec working group seeks to address these goals in a manner consistent with the security model that is currently being defined for the Web services framework. An associated OGSA Security Roadmap document enumerates the security related specifications that will be needed to ensure interoperable implementations of the OGSA Security Architecture.

3.6.2 Model

This section describes a model to facilitate the reasoning about and the specification of the OGSA security services. The presented model is not a formal model in the mathematical sense, but is meant to provide a language to describe, and to obtain common understanding about, the security policies that are to be enforced.

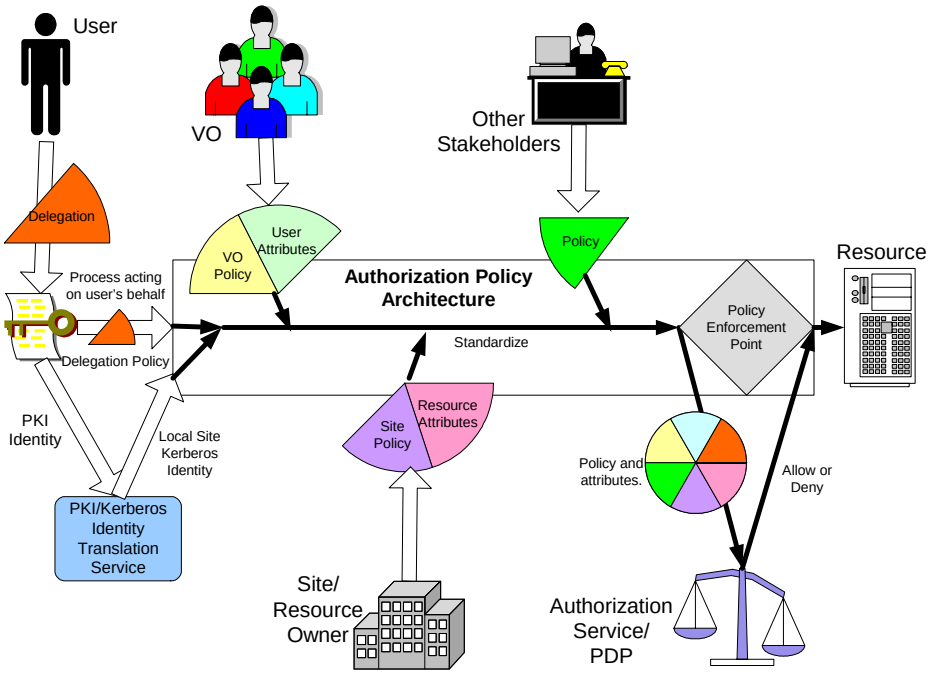

Figure 9. Example input for the policy decision and enforcement

In general, one can make the observation that entities are interacting through mechanisms within a context. Entities are things like users, subjects or services. Interaction mechanisms span all the different communication methods, like mail, telephone, http, soap, ssl, etc. A context puts the interaction in perspective, and could localize the interaction to a single machine, or could be associated with a service invocation within a VO, and/or with an established secure association, and/or with a distributed transaction.

All these entities, mechanisms and contexts can be described by sets of attributes or properties. Some of these attribute types and values may be used for unique identification, others are used for classification or grouping. Furthermore, some of these attributes are so-called inescapable, like a shared secret, private/public key, fingerprints, etc. Each of these inescapable attributes will identify an entity by themselves. All other attribute values are essentially bound to an entity's inescapable attribute by an issuer or attribute authority. An example is the X.509 identity certificate, where a trusted third party, the Certification Authority (CA), binds the subject's name to a public key.

Security policies are statements about these different entities, interaction mechanisms and contexts, and specify restrictions on the associated attribute values, properties and their relationships. The policy statements (or rules) will be able expressed in terms of these entities (e.g. identities, user attributes), resources (i.e., end points), and environment characteristics (e.g. time, location, purpose, trust level of the requester or request path, etc). These policies will be about various aspects including authorization, authentication, trust, identity mapping, delegation, assurance levels, etc.

The model defines the security services as entities with interaction patterns that facilitate the administration, expression, publishing, discovery, communication, verification, enforcement and reconciliation of the security policy. In other words, the security policy enforcement is the ultimate goal, and the security services are designed and deployed to support that goal.

To make this model more concrete, we will identity a number of these entities, interaction mechanisms and contexts, and discuss some of their attributes and common relationships. Figure 9 shows an example of the policy enforcement to protect the use of a resource. In order to understand the model, it helps to walk through the example in Figure 9 from the two opposite ends: on one side the initial user authentication, and on the other the enforced authorization, and realize that a user will only be allowed to access the resource if the authorization policy evaluated at the enforcement point will yield permit. The resource's authorization policy will be expressed as rules for the attribute values of the relevant entities, like for example the user's name, her role, her VO-membership, etc. With the initial authentication of the user, only inescapable attribute values are presented and verified, like the possession of the private key associated with the presented public key. Normally, there is a mismatch between the fact that the policy is expressed in derived attributes and the initial authentication only yields a key, and this can be resolved by finding the matching attribute assertions that bind the key to the attributes used in the policy expression. One can see in Figure 9 that several security services are depicted, which are used to federate the user's public key credentials to Kerberos ones, and to obtain attribute from different sources, like the VO or the local site that assert specific attribute bindings used in the enforced policy statements.

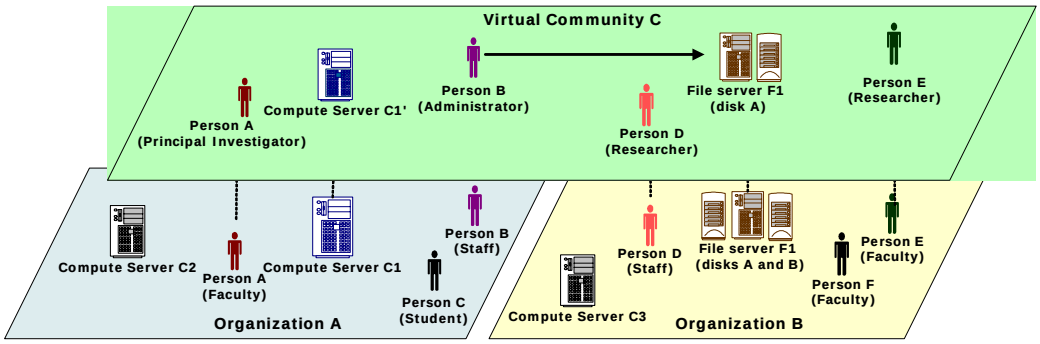

Figure 10. The Virtual Community Concept

Conceptually, the presented model is no different from the general Web Services Security model and other distributed computing architectures. The Grid applications, however, put a distinct focus on the entities and patterns related to cross-organizational settings, and the security services that enable those interactions. It can be noted that such interactions are also prevalent in business applications where business-to-business interactions (or organizations boundaries within an enterprise) bring about similar requirements and settings. For example, Figure 10 shows two organizations and an overlaid virtual community that is governed by its own policy. Note that the entities within this virtual community context have their own specific attributes and properties, which are different and have no relation to those in their home domain. This highlights that our security services model has to support the concurrent enforcement of multiple policies that each have to be evaluated within their own proper context.

3.6.2.1 Example VO-Scenario: Digital Library

A digital library program for educational material is operated by a public organization. A number of schools and public libraries in the nation participate to the library program. The program provides teachers and students of the participating schools or libraries with a method to share educational materials such as digital books, videos, photos and all other digital materials for education that are originally stored by each schools or libraries. Each of the participating schools and libraries is responsible for its users and resource (educational materials) management, such as registration or removal of users and resources. There is also a case where some schools, like universities, consist of several sub organizations. In such a case, each sub-school in a university could form a VO by itself and the university VO becomes an aggregation of these school VOs. This scenario is depicted in Figure 11.

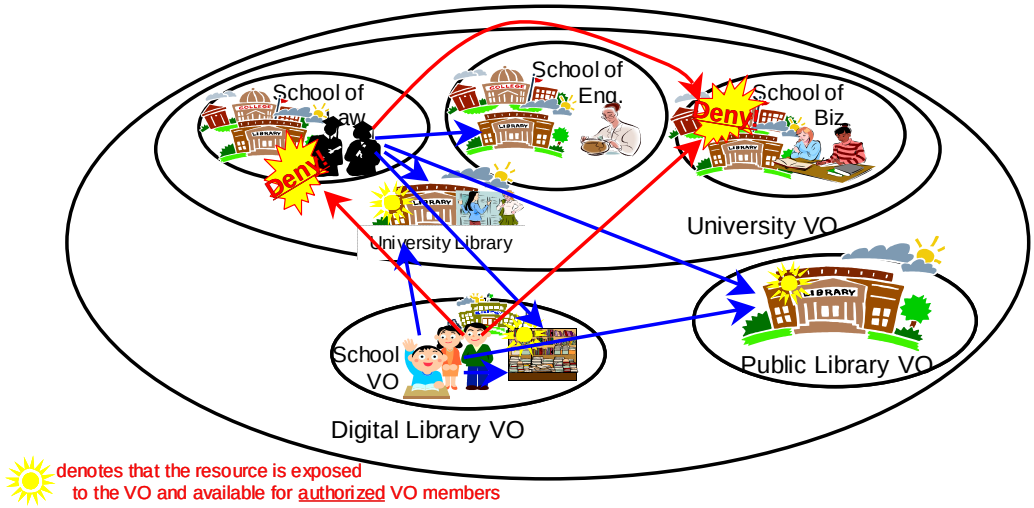

Figure 11. VO example: Digital Library

The following is a list of example policies of how users of the library program have access to the shared materials.

- All the students enrolled in a participant school can have read access to the materials for students.
- All the teachers can have read access to materials both for students and faculties.
- Some certified teachers by the participant school can also register new materials.

The school libraries in the university may allow access from inside the university, but will not allow accesses from the Digital Library VO.

3.6.3 Functional Capabilities

In the OGSA model, the functional capabilities are translated into service descriptions and usage patterns.

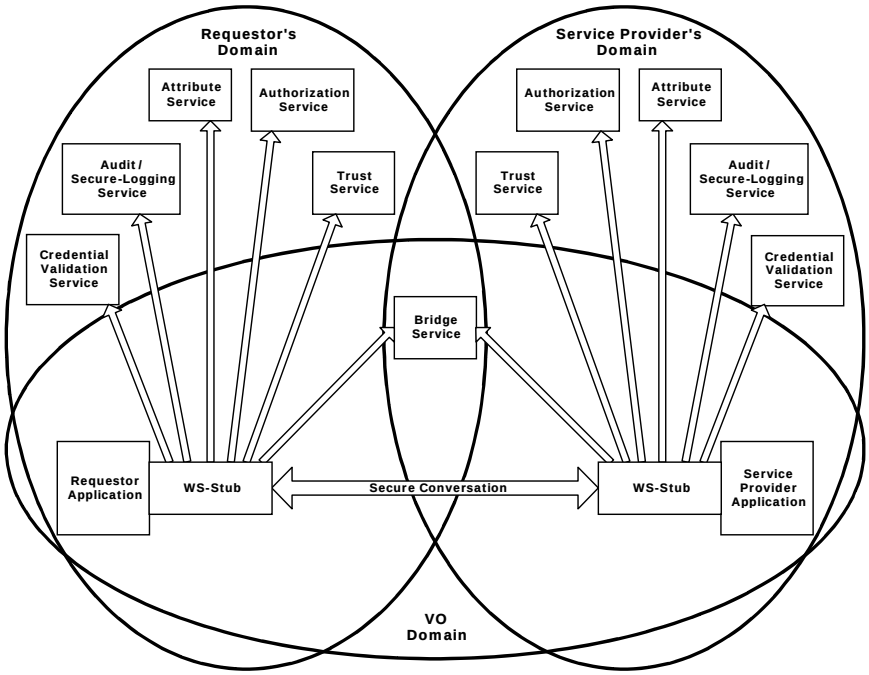

Figure 12: Security services in a virtual organization setting

As an example, Figure 12 illustrates how the requester and service both call-out to different infrastructure security services to ensure policy compliance. Note that in the picture the call-outs are made from within the stubs, and outside of and transparently to the application. The expectation is that most of the policy enforcement could be taken care of this way, which has desirable benefit of keeping the security specific code to a minimum for the application developers.

The functional capabilities can be enumerated as a set of Grid security services:

- An authentication service: An authentication service is concerned with verifying proof of an asserted identity. One example is the evaluation of a User ID and password combination, in which a service requestor supplies the appropriate password for an asserted user ID. Another example involves a service requestor authenticating through a Kerberos mechanism, and a ticket being passed to the service provider's hosting environment, which determines the authenticity of the ticket before the service is instantiated.
- Identity mapping service: The identity mapping service provides the capability of transforming an identity that exists in one identity domain into a identity within another identity domain. As an example, consider an identity in the form of an X.500 Distinguished Name (DN), which is carried within a X.509 V3 digital certificate. The combination of the subject DN, issuer DN and certificate serial number may be considered to carry the subject's or service requestor's identity. The scope of the identity domain in this example is considered to be the set of certificates that are issued by the certificate authority. Assuming that the certificate is used to convey the service requestor's identity the identity mapping service via policy may map the service requestor's identity to a identity which has meaning (for instance) to the hosting environment's local platform registry. The identity mapping service is not concerned with the authentication of the service requestor; rather it is strictly a policy driven name mapping service

- Authorization service: The authorization service is concerned with resolving a policy based access control decision. The authorization service consumes as input a credential that embodies the identity of an authenticated service requestor and for the resource that the service requestor requests, resolves based on policy, whether or not the service requestor is authorized to access the resource. It is expected that the hosting environment for OGSA compliant services will provide access control functions, and it is appropriate to further expose an abstract authorization service depending on the granularity of the access control policy that is being enforced.
- VO Policy service: The VO policy service is concerned with the management of policies. The aggregation of the policies contained within and managed by the policy service comprises a VO's policy set. The policy service may be thought of as another primitive service, which is used by the authorization, audit, identity mapping and other services as needed.
- Credential Conversion service: The credential conversion service provides credential conversion between one type of credential to another type or form of credential. This may include such tasks as reconciling group membership, privileges, attributes and assertions associated with entities (service requestors and service providers). For example, the credential conversion service may convert a Kerberos credential to a form which is which is required by the authorization service. The policy driven credential conversion service facilitates the interoperability of differing credential types, which may be consumed by services. It is expected that the credential conversion service would use the identity mapping service.
- Audit Service: The audit service similarly to the identity mapping and authorization services is policy driven. The audit service is responsible for producing records, which track security relevant events. The resulting audit records may be reduced and examined as to determine if the desired security policy is being enforced. Auditing and subsequently reduction tooling are used by the security administrators within a VO to determine the VO's adherence to the stated access control and authentication policies.
- Profile Service: The profile service is concerned with managing service requestor's preferences and data which my not be directly consumed by the authorization service. This may be service requestor specific personalization data, which for example can be used to tailor or customize the service requestor's experience (if incorporated into an application which interfaces with end-users.) It is expected that primarily this data will be used by applications that interface with a person.
- Privacy Service: The privacy service is primarily concerned with the policy driven classification of personally identifiable information (PII). Service providers and service requestors may store personally identifiable information using the Privacy Service. Such a service can be used to articulate and enforce a VO's privacy policy.

The security of a Grid environment must take into account the security of various aspects involved in a Grid service invocation. This is depicted in following Figure 13.

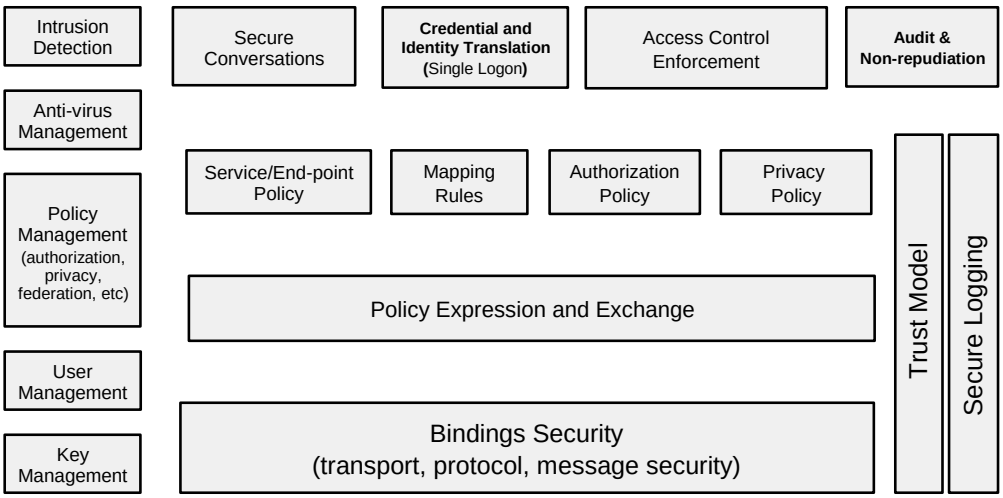

Figure 13. Components of Grid Security Model

All security interfaces used by a service requester and service provider need to be standardized within OGSA. Compliant implementations will be able to make use of existing services and defined policies through configuration. Compliant implementations of a particular security related interface would be able to provide the associated and possibly alternative security services.

3.6.4 Properties

In general, the properties of the security services depend on the technological requirements that follow from the policy that has to be enforced. For example, in order to be able to enforce a stated policy, certain service levels have to be met by the security services, which translates in properties like:

- Maximum latency
- Response-time
- Availability
- Recovery

In many cases, the implementation of the security services will be able to obtain the desired properties through the use of other services. For example, the attribute information service could use the data services to access the assertion information in ldap or rdbms, and it could make use of the data mirroring features of those services to achieve the desired availability property needed to meet the enforcement of the stated security policy.

3.6.5 Intersections with other OGSA services

In general, the invocation of any OGSA service will be subject to security policy enforcement. In some cases, this enforcement is implicit and hard-coded; in other cases the ability to plug-in or call-out to external security infrastructure services is essential for deployment.

In some cases, the security services and requirements are intimately connected to other OGSA services on a higher level. A few examples are discussed below.

3.6.5.1 Least Privilege Delegation

The delegation of rights is a fundamental capability needed to let services work on behalf of other entities. With this rights delegation comes the associated risk that any of these services may be

compromised and use those right in inappropriate ways. To limit the exposure, one would like to limit the delegated rights to only those rights truly needed by the service. This least privilege delegation model requires that one is able to match the invoked service operations with the exact "amount" of rights, which is a non-trivial requirement. Many Grid applications use the concept of jobs, which job directives are specified in their own language. The job requirements are then matched with the capabilities and availability of resources by discovery, brokers, and scheduler services. The language used for the expression of these job directives and resource capabilities should be able to match up with the directives used to express the equivalent rights needed. Any mismatch will result in deployment where essentially too many rights will have to be given to services to ensure the possible execution of the job directives. This issue is illustrated inFigure 14.

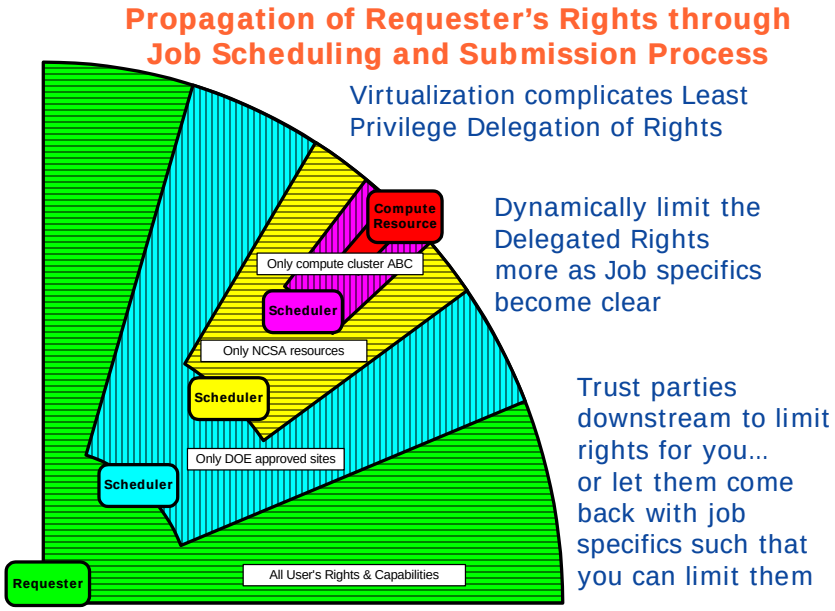

Figure 14. Least Privilege Delegation and Job Description

3.6.5.2 Secure logging in a distributed environment

Logging services and the secure access to the logs for reconciliation purposes, becomes a much harder problem in a distributed setting where the services and associated logs reside in different administrative domains as is depicted in Figure 15.

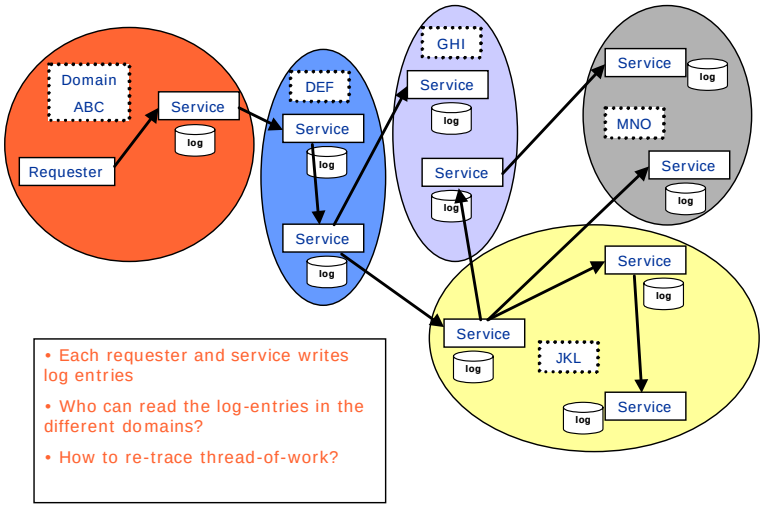

Figure 15. Secure logging in a distributed environment

Logging services should have security characteristics so that logs are secured, capable of being tamper-proof and ensure integrity of the messages. When we get to that level of sophistication, it leads to the notion of auditing – where events are recorded in a secure fashion. These events can be any events including security events, business events, transaction events, etc. Security services and infrastructure will need to deal with generating security events that can get consumed by event infrastructure so that those can be audited, or acted upon (e.g. intrusion defense system may react to a set of denial of service events).

3.7 Self-Management Services

3.7.1 Objectives

Self-management was conceived as a way to help reduce the cost and complexity of owning and operating an IT infrastructure. In such an environment, system components—from hardware such as desktop computers and mainframes to software such as operating systems and business applications—are self-configuring, self-healing and self-optimizing.

These self-managing attributes, described further in §3.7.4, suggest that the tasks involved in configuring, healing and optimizing the IT system can be initiated based on situations that the components themselves detect, driven by business needs, and that these tasks are performed by those same technologies. Collectively, these intuitive and collaborative characteristics enable enterprises to operate efficiently with fewer human resources, while decreasing costs and enhancing the organizations' ability to react to change. For instance, in a self-managing system, a new resource is simply deployed and then optimization occurs. This is a significant shift from traditional implementations, in which a significant amount of analysis is required before deployment, to ensure that the resource will run effectively. Though it is expected that the self-managing attributes will be pervasive in OGSA, it is not the case that every single service will demonstrate all or a subset of these attributes. Rather these attributes are part of the autonomous nature of the system as a whole.

One of the main objectives of self-management is to support service-level attainment for a set of services (or resources, depending on the taxonomy) – with as much automation as possible to reduce the costs and complexity of managing the system. In an operational environment, it is often necessary to control various aspects of the behavior of a solution component in a manner that cannot be determined a priori by the component developer. This is achieved in a self-managing system through the deployment of policies to govern the behavior of system components derived from business objectives; a role called *Service Level Manager*. Service Level Managers (SLMs) are responsible for setting and adjusting policies, and then changing the behavior of the managed resource or service in response to observed conditions in the system to ensure overall compliance with business objectives. SLMs are themselves managed by policies that are either embedded in their implementation or retrieved from other SLMs.

Thus composition and hierarchy are expected between different SLMs, thereby significantly reducing the complexity of operation of the system and initial system design, since complex SLMs can be built by involving simple SLMs in the process.

While the self-management set of services/mechanisms are a significant part of the OGSA, this work is still at a preliminary stage and hence only some aspects of self-management are described here. More detailed analysis and architecture will be addressed in later versions of this document.

3.7.2 Basic Attributes

The collection of attributes collectively needed for various stages of self-management are given below. Their existence at a conceptual level is discussed; no assumptions are made about their physical (binary) existence.

3.7.2.1 Service Level Agreement

Service Level Agreements (SLAs) include business or IT agreements between the service provider and the users of the service. SLAs provide guidance, expressed in terms of measurable intent, as to the purpose and delivery objective of the service or resource that is being managed.

3.7.2.2 Policy

A Policy is used to govern the behavior of an SLM and the manageable resources under its control. Policies governing the behavior of self-managing entities are derived from service level agreements (SLAs) and deployed as part of the management context under which the manageable resources are used. SLMs orchestrate real-time changes in the dynamic management infrastructure based upon policy which governs their behavior.

3.7.2.3 Service Level Manager Model

This model provides the framework to instantiate and work with an SLM such that various SLMs and human operators can interact and understand each other without having knowledge about each other built in at design time. There is an interface component to the SLM model, as well as a representation of a SLM and its component management services (not the managed services) and control loops that they form, and instantiation and maintenance of SLMs.

SLMs are typically modeled after a *generic control loop pattern* that may be used to control and adjust various service activities. This pattern is a cycle consisting of Monitoring, Analysis and Projection, and Execution phases. SLMs carry out *service level attainment* activities. Service Level Management is further described in §3.7.4.1.

3.7.3 Example Scenarios

Self-management capabilities are fundamental to the Grid. This section describes two examples of real world usage: Job level management and Grid system level management. Similar concepts can be derived by analyzing how self-management works for other scenarios, e.g., security and so on. In the following scenarios the terminology used is that of the Execution Management Services and the Commercial Data Center use case.

3.7.3.1 Job Level management

The IT business activity manager submits a job and negotiates the job's SLA. The job must be executed so that it satisfies the agreement. At a later stage, the IT business activity manager may wish to renegotiate the agreement to address new business requirements. It does so with the Job Manager and updates the existing agreement. When the agreement is updated the resource requirements are recalculated, and in the service level attainment loop the provisioning steps are triggered as a result of the changing conditions. After the provisioning step, the resources are in a ready state for the required components of the job to start, including starting executable resources such as application server or DBMS.

The IT business activity manager can monitor the load and resource utilization of the provisioned resources using the Monitoring Service or the Metering Service. The IT business activity manager also can obtain the state of the running job from the Job Manager.

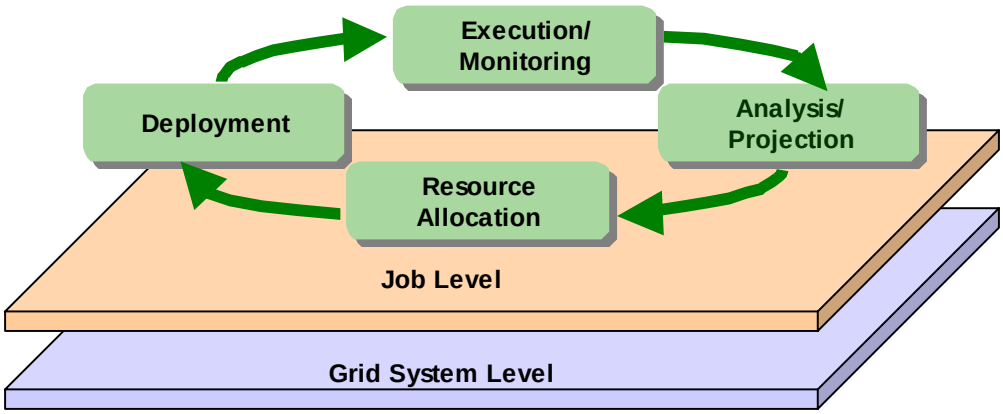

Figure 16 Example scenarios

3.7.3.2 Grid System Level management

At the Grid system level one aim is to improve resource utilization while maintaining the SLAs of running jobs. The Grid system service level manager may have to add new resources to prepare for expected load increases, and to release surplus resources in order to reduce costs. The resources allocated to a job may also have to be adjusted based on policy – e.g. the priority of a job relative to other jobs in the Grid system.

In the Analysis and Projection phase information about the available resources, current load, estimates of the expected future utilization, and load are evaluated. An expected increase in utilization may trigger provisioning steps to add resources to the Grid system's available pool, e.g. by arranging to shift resources from other systems or by releasing resources used by jobs.

3.7.4 Functional Capabilities

Self Management is essentially self-configuration, self-healing, self-optimization. This shows the essential difference of self management with other OGSA categories – namely it is not just about the components that are involved in doing self-management, but the method by which it is done – the *sauce* or the way in which the components interact, control loops are formed and systems behave intelligently, based on environmental changes. The mechanisms that bring self-management about can be described as follows;

- Self-configuring mechanisms adapt dynamically to changes in the IT system, using policies provided by the IT professional. Such changes could trigger provisioning requests leading to, for example, the deployment of new components or the removal of existing ones, maybe due to a significant increase or decrease in the workload.
- Self-healing mechanisms can detect improper operations of and by the resources and services, and initiate policy-based corrective action without disrupting the IT environment. Self-healing has an element of self-protection included in it as well. This means that components can detect hostile behaviors as they occur, and take corrective actions to make themselves less vulnerable. The hostile behaviors could include unauthorized access and usage, virus infection and proliferation, and denial-of-service attacks.
- Self-optimizing mechanisms are able to tune themselves to the best efficiency to meet end-user or business needs. The tuning actions could mean reallocating resources to improve overall utilization or optimization by enforcing an SLA. Self-optimization makes use of self-configuration in its implementation.

It is important to note that the self-configuring, self-healing, self-optimizing attributes are not independent of one another. They work together to allow changes to be made to the configuration of one or more aspects of the IT system.

All of the OGSA service categories are utilized in achieving self-management. In addition, the following sections highlight some special functional requirements that are important for self-management, but that may not be reflected in the same way in other OGSA service categories.

3.7.4.1 Service level management

Service level management ensures that the desired Quality of Service (QoS) is maintained. This is done through the activities of the SLMs as described in §3.7.1. These *service level attainment* activities are described in more detail here.

- **Monitoring**: The SLM receives and processes resource instrumentation through a monitoring component. Service execution and monitoring of resource utilization, for example, monitoring the load and utilization of resources, the running states of service components, and detecting faults, may be made possible by using the monitoring services from the general Resource Management category in OGSA. Such monitoring information is used as input to the Analysis phase.
- **Analysis and Projection:** Analysis is performed against the instrumentation to evaluate and determine compliance with the established policy and SLAs. The manager gets to know if resources are meeting QoS objectives and operating within defined policies. Analysis can also predict future resource behavior based on history and projection requirements. When system behavior is not consistent with overall goals, the manager evaluates alternative courses of action to effect changes in the set of configured resources in its sphere of influence, and selects a plan of action in accordance with its configured policies.
- **Execute (Actions):** The manager then executes the plan, either by interacting with underlying managed resources or by communicating with other managers responsible for other aspects of

the system. For example a workload manager may adjust priorities and process shares of tasks executing within a cluster of processors to meet service level objectives and policies. Or, in cases where local action against a pool of resource is either impossible (in violation of policy or constrained) or ineffective, a resource manager may "appeal" to other management functions to remedy the situation. For example, a workload manager might make a provisioning request to add additional processors to a cluster.

This control loop executes continuously, assessing the current state of the system against the expressed service level objectives and making adjustments as necessary to bring the system into compliance. Other services are needed for these activities to be successful –for example *capacity planning, entitlement of resources, problem and root cause analysis, predictive analysis,* etc.

A number of management components may be involved in service level attainment activities**.** The QoS and SLA requirements addressed by these management activities can be extremely varied. For example, they may include performance attainment objectives such as processor capacity or utilization (workload management), or more qualitative objectives such as security levels. Service level attainment activities affect each other directly or indirectly. Therefore the relationship and order of execution between the different management components in the control loops and service level managers should be fairly loose.

3.7.4.2 Policy and Model based Management

Service level managers orchestrate real-time changes in the dynamic management infrastructure based upon policy which governs their behavior. The intelligence in the self management is basically directed by the policies and models about the SLMs and the services that are being managed.

3.7.4.3 Entitlement

Entitlement is about negotiating with other resource holders (other SLMs, human operators, resource pools etc.) to obtain the right resources. During this process different SLAs are compared to find out which resource need is more important. Entitlement is an earlier phase to resource reservation and provides a looser binding – an option to reserve. In contrast, a resource reservation guarantees access to the resource.

3.7.4.4 Planning

Planning refers to calculating the optimum requirements of a service in terms of the resources it needs. This can be at an initiation stage, due to some problem/root cause being identified, or due to a command from a higher level.

3.7.4.5 Capacity Management

Capacity management includes the actual actions relating to updating the current state and requirement of resources. It includes things such as linking with inventory, asset management, moving the resources from the current location into the service domain, moving the resources from the service domain into a storage area or to another domain, etc.

3.7.4.6 Provisioning

Provisioning, including its sub-activities of deployment and configuration, is an important activity that is done in support of self-managing actions as resources are prepared for their expected use. Provisioning is supported by a number of other OGSA services, such as the Application Contents Service that maintains the deployment contents and configuration descriptions.

3.7.4.7 Analytics

3.7.4.7.1 Problem and root cause analysis

Analyze monitored data to find out the express issue and the root cause of detected problems. Includes filtering capabilities, correlation etc.

3.7.4.7.2 Predictive analysis

Analyze monitored data to predict future behavior – for example to plan for future resource needs, or to predict future problems.

3.7.5 Properties

Service Level Management components implement self-managing, policy-based capabilities across multiple QoS dimensions of the management infrastructure. Key QoS dimensions include, but are not limited to, availability, security, and performance. These cornerstone QoS dimensions are expressed in terms of measurable intent captured in SLAs. Availability might be measured in terms of "minutes of outage." Desired security capabilities might be described as privileges provided by a class of service for associated users. Performance characteristics could be specified in terms of well-known throughput or response-time objectives.

3.7.5.1 Performance

Gives measurements and metrics – for example, quantifiable data about how the system is performing, such as cumulative CPU load factor, that go into computing the performance metrics of a specified service.

3.7.5.2 Availability

Specifically denotes the metrics that show the failure rate of the service – of all types, including the macro cases such as system crash, and other less observable cases such as the failure of a service to deliver due to throughput drop and buffer flushing.

3.7.5.3 Security

Security is an extremely important aspect of the IT environment. These measures may include security violations, probability of security violations, and identity management metrics such as the time to set up or delete a user, etc.

3.7.6 Interactions with the rest of OGSA

Self-management interacts with almost all other aspects of OGSA. As the required interactions are still being worked out, only some services are listed here as an example of what interactions are expected, with no attempt at being exhaustive.

- Discovery – to find and integrate new resources and services.
- Logging and Monitoring – to provide the information needed to determine the state of the system.
- Resource Reservation – to facilitate more predictable resource usage.
- Workflow – to automate the actions that the SLMs have to carry out when addressing abnormal conditions.
- Composition – to construct complex or higher level SLMs from simple ones.
- Security, and in particular Authentication and Authorization, are essential as different system components may be involved in management actions.

- Resource Management, and in particular Service or Resource Manageability Models, are required to provide the representation of the service or resource that is being managed. The SLMs can read this information and can act on the intelligence in this, in response to a changing environment or a command from a higher level. Further work as to how best to model a service or resource needs to be done.

3.8 Information Services

3.8.1 Objectives

A vital capability of OGSA is to allow all necessary information about applications, resources and services in the Grid environment to be manipulated in an efficient manner. The term *information* is used here to refer to dynamic data or events used for status monitoring, relatively static data used for discovery, as well as any data used for logging purposes. The information service needs to respond to queries in a timely fashion and provide the required delivery guarantees and quality of service (QoS). The scope of the OGSA information service only covers the stages from when information is published into it to when it is consumed from it. Stages prior to publishing information (e.g. sniffing network packets) or following its consumption are out of scope.

While it is conceivable to design one single information service that deals with all types of information delivery patterns and QoS, OGSA is best served by multiple services, some general, and some optimized and tailored to a set of use cases or categories of information. However, caution should be exercised so as not to end up with a number of fractured information services each capable of answering a very limited number of use cases. The challenge is to balance the desire for generality and other requirements on performance, publication and delivery semantics etc. In theory, generality can be achieved by abstracting common (high-level) functionality and features covering the requirements of as many use cases as possible. By defining standard interfaces of the resulting services, many interoperable implementations are then possible. The question of how far can the level of abstraction be raised without compromising the usability of the services, and also still allowing for the required concrete interface definitions, needs to be looked into.

Clients of the OGSA information services include, but are not limited to, an execution management service, an accounting service, a problem determination service, a resource reservation service, a resource usage service, and application monitoring. To facilitate interoperability, the information services themselves should be built on top of OGSA infrastructure capabilities such as notification (WS-Notification), and they could also make use of other OGSA capabilities such as data access and distributed query processing.

3.8.2 Models

3.8.2.1 Information Producer/Consumer paradigm

The categorization of information depends greatly on factors such as the demand placed on the source of information (static versus dynamic, publication rate, …), its purpose (discovery, brokering, logging, monitoring, …) and its QoS requirements. However, for all information we see similar structures. Information is made available for consumption, either from the originating producer, or through an intermediary (e.g. logging service, broker) acting on behalf of the originating producer. Either one or more consumers wish to obtain information from one or more producers, or one or more producers wish to send information to one of more consumers. Producers and consumers should be decoupled and not be required to have any prior knowledge of each other. In the first case, for example, the consumers contact the producers (or intermediaries) that hold the information they are interested in, and get that information either in

one call or by subscribing to receive it continuously as it becomes available. Mechanisms are required for allowing information to flow from producers to consumers in a timely fashion.

3.8.2.2 Data models/Query languages

OGSA is not prescriptive on the data model used to implement an information service or the language used to query for information. Current systems broadly fall into those that are based on XML and Xpath/Xquery query languages (e.g. Globus MDS) and those that use the relational model and the SQL query language (e.g. R-GMA).

3.8.2.3 MetaData models

3.8.2.3.1 Message schema

Metadata is associated with information (messages) for describing its structure, properties and usage. For interoperability, a standard event scheme for OGSA information service is necessary. In some cases, such as when performance is paramount and interoperability is not a concern, use-defined, optimized events are more appropriate.

3.8.2.3.2 Producer/consumer descriptions

An information service might allow producers and consumers to discover each other by making detailed descriptions about themselves available for querying. A special distributed registry or point-to-point scenario could be used for that purpose. The descriptions could include, for example, the type of producer or consumer, what information they produce or consume, and their endpoint URLs.

3.8.2.4 Example scenarios

3.8.2.4.1 Directory scenario

A user needs to locate a service description that meets some desired functional and other criteria. He queries a central directory service where service descriptions are published, and receives an answer. The directory owner decides who can publish information into the directory and who is authorized to query it. UDDI is an example legacy directory service. OGSA directory services could be based on WSRF service group concepts.

3.8.2.4.2 Logging scenario

A number of distributed computing usage scenarios require log services. These scenarios include those based on problem determination, usage metering, failure recovery, transaction processing, and security.

A problem determination scenario might proceed as follows. A developer is writing code for a Disaster Recovery application. Following corporate programming guidelines for logging, he inserts log statements into his code to provide information regarding any unexpected situations. He is responsible for coding a section of the application that accesses an RDBMS to obtain attributes for candidate fail-over resources. He codes this section of the application such that RDBMS failures are trapped and log records are generated to reference the RDBMS failure. After development has been completed, the application is deployed into an operating environment where log records are processed by a handler chain. In this environment, based on the severity level (e.g., fatal, warn, info), records exceeding an operator-specified level are stored in a log by a file handler. Due to the deployment of an erroneous new security policy, the code reaches an abnormal state and logs a record indicating that the underlying RDBMS has denied access to the

application. The operator had configured the operating environment to store log records of this severity. After the failure, an analyst, in the role of a log consumer, uses a console application to peruse the logs. He finds the log statement indicating the RDBMS problem and directs the data base administrator to correct the security policy.

3.8.2.4.3 Grid Monitoring Architecture (GMA) scenario

A scientist wants to run an interactive simulation/rendering job that must be finished within the next half hour. He must find 100 computing elements that are fast enough, have enough memory between them, and are linked by fast network connections. He needs up-to-date information and submits a complex query to the OGSA information service. After a very short time he receives the information on the computing resources that satisfy his query. He selects the nearest ones and proceeds with his job.

In answering the query, the information service first discovers the information producers for the computing, storage and network resources that are relevant and then gets the necessary information before performing a "join" to identify the resources that satisfy the query. The scientist then needs to monitor his job and respond to partial failures in the execution environment, possibly using the same GMA-based service (provided that the information has an associated timestamp). GMA is an example of a general information services architecture that can be used for more than one purpose (see GMA document (GFD-1.7) from the GGF).

3.8.2.5 Producer/Consumer patterns

The basic pattern of decoupling producers from consumers using an intermediary is widely used in computing. The Producer-Intermediary-Consumer pattern is shown at the top of Figure 17. Producers put data *into* an intermediary, and consumers extract data *from* an intermediary. In a general sense, this is the pattern followed by any data store. That is, producers write to and consumers read from a file, RDBMS, ODBMS etc.

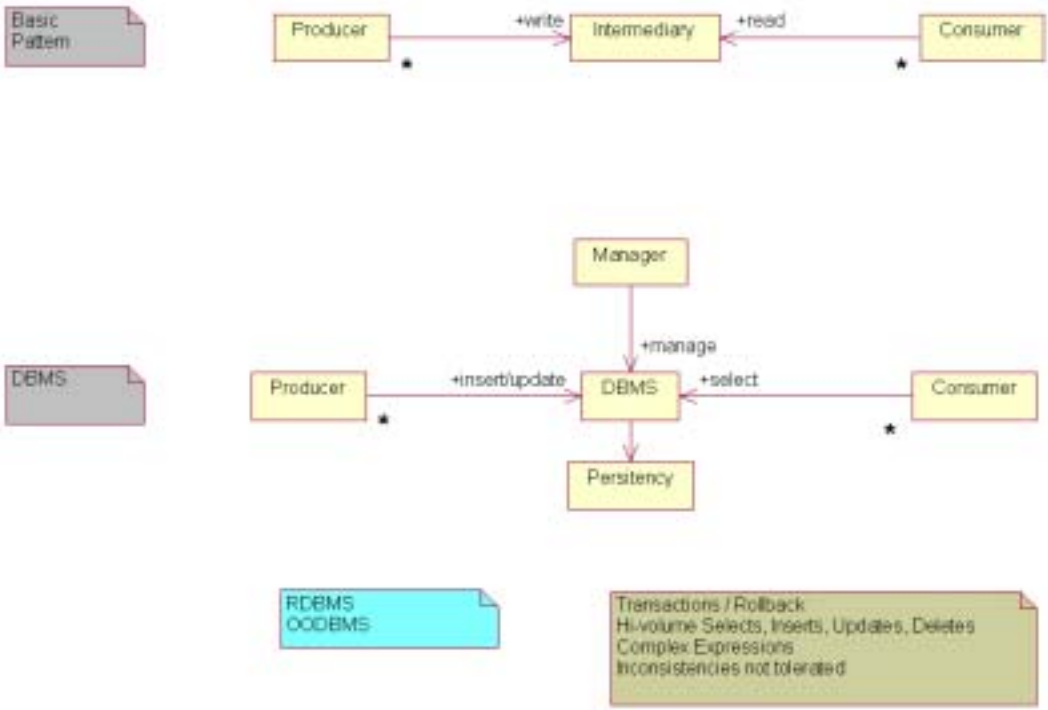

Figure 17: The Producer - Intermediary - Consumer Pattern.

Figure 17 also shows that, in addition to supporting producer and consumer interfaces, a data store may support a management interface. The management interface controls those functions that are not directly associated with reading or writing to the data store. Operations controlling retention policy, backup policy, etc. would be accessed through a management interface.

In distributed computing, the above pattern is followed by several infrastructure services whose task it is to provide information to consumers. We broadly refer to these services as information services. They include:

- Name services
- Log services
- Notification services

We assume that the reader is generally familiar with these concepts and we defer OGSA-specific discussions on each service.

Each of these services has evolved to support a distinct set of domain-specific semantics and qualities of service. While they each could be implemented using a general-purpose RDBMS for an underlying data store, this would most likely result in trading off some desirable, specialized qualities of service for unneeded functionality. For example, a log service that must support extremely fast writes would most likely find the performance and footprint costs of a full-function RDBMS to be prohibitive. Similar observations could be made about name services and notification services.

Figure 18 attempts to characterize each of these information services. Each service is labeled on the left. Service semantics and data store requirements are depicted using UML, and examples of existing technologies, along with a brief summary, are shown to the right of each service.

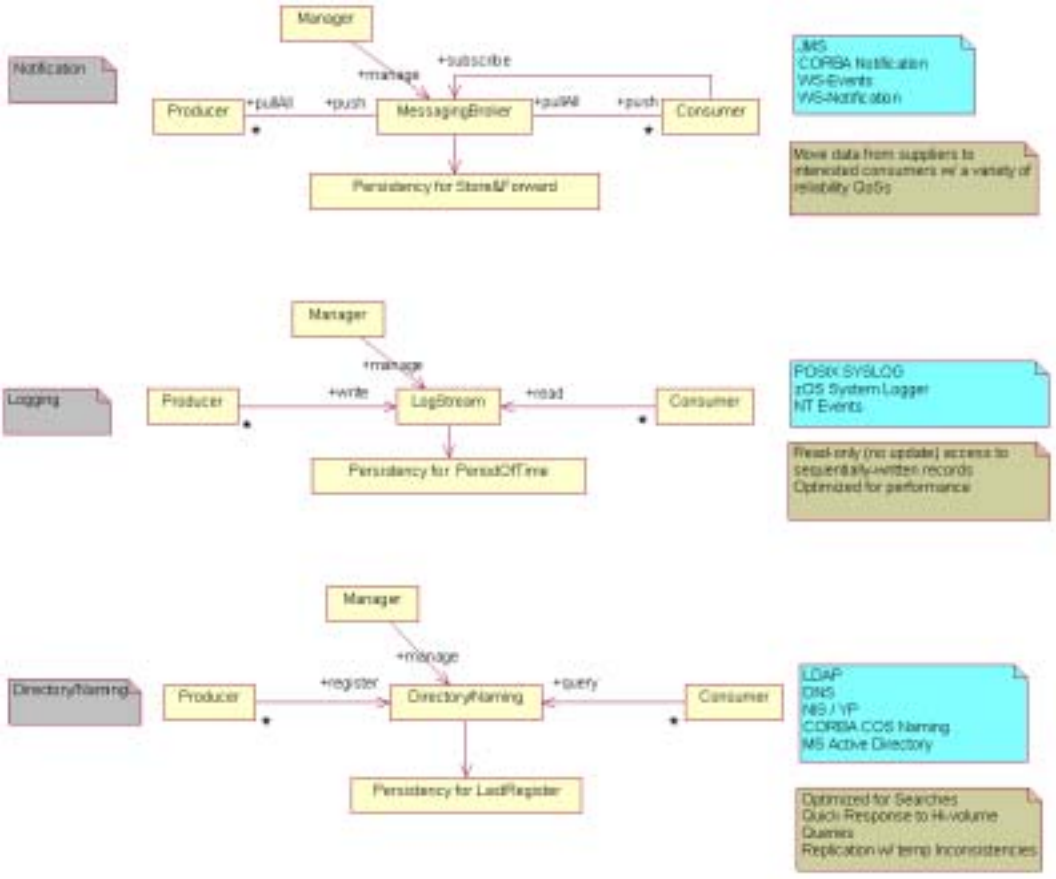

Figure 18: Basic OGSA Information Services.

A specific information service could be based on either one single pattern or a combination of these basic patterns. Adhering to a single pattern can result in a very specialized service, applicable to only a subset of use cases. This might be desirable when other requirements (for example performance or specific semantics) are paramount. Analysis of the OGSA use cases reveals that a service that is based on a combination of patterns is often desirable (both directory and notification patterns are relevant for GMA, for example).

3.8.3 Functional capabilities

3.8.3.1 Discovery

One capability that is universally needed in OGSA is service and resource discovery. A directory (or registry) is an obvious solution, but not the only one. A directory is distinguished from other possible solution in that it has persistent storage for the "latest" information and is optimized for searches. Low latency response to a high volume of queries is required. The directory may be replicated for scalability.

Alternatively, a compilation or guide of information can be stored in an *index* (such as Google). Unlike a registry, which tends to be centrally controlled, anyone can create an index.

Another alternative is peer-to-peer discovery, where a Web service is a node in a network of peers and dynamically queries its neighbours in search of a suitable match. The query propagates through the network from one node to the other until a match is found, or a particular hop count is reached, or some other termination criterion is satisfied. Another alternative for a discovery service is a general GMA-based service (see below).

3.8.3.2 Message delivery

Producers and consumers interact by exchanging messages, and this can be handled by a common messaging infrastructure. This infrastructure is only concerned with how to distribute copies of messages to interested parties, not how these messages are constructed in the first place. Producers either send messages directly to relevant consumers or make use of an intermediary (message broker) that decouples producers from consumers. In the latter case the producers publish their messages to the broker, which takes the responsibility for forwarding the message to interested parties. A producer (or the intermediary) may provide notification capabilities and additional function such as a finder service (allowing its producers and consumers to find each other). Message brokers may store and forward messages in stable storage – not necessarily for persistency, but for reliable message delivery purposes.

3.8.3.3 Logging

An OGSA logger service acts as an intermediary between log artifact producers and consumers. Producers write log artifacts sequentially, and consumers may read (but not update) the log records. To ensure the general acceptance of the basic log semantics and to enable the exploitation of existing implementations, OGSA logging services should support key features found in existing logging implementations. There may be multiple producers and consumers for a given logging intermediary, and both may set filters for records. In the logger service the message exchange is optimized for performance and the records are kept in a persistent store for a period of time.

3.8.3.4 Monitoring

Information that carries a field for ordering purposes (typically a time stamp or sequence number) can be used for monitoring. An OGSA monitoring service could be equally used for applications or resources. Some situations (for example real-time applications) might impose strict requirement on the monitoring service (e.g. high update rates and high performance). In such a case a special-purpose service might be needed.

3.8.3.4.1 General Information and Monitoring service

A general OGSA service that provides a combination of the above capabilities can provide more flexibility to the end user. For Grid resources in general (including services and applications), the amount of available information about resources could be very large, dispersed across the network and updated very frequently. Searches in this space will have unacceptable latencies. In order to manage such information in a controllable way, it is important to separate information source discovery from information delivery. Searches should be only used to locate information sources or sinks. A special-purpose directory is used to hold metadata about the resources. In this case the directory (or registry) must cope with high rates of updates that are expected in the dynamic OGSA environment. Individual producer/consumer pairs can limit the amount of data flowing between them to that satisfying the consumer query. This model differs from a message broker that combines the mechanisms for finding sources and sinks of information and its

delivery into a single searchable channel. The merits of this approach are described in the GMA document (GFD-1.7) from the GGF.

A user of such a general service should be able to put any information (irrespective of its intended use: e.g. discovery, monitoring, …) into it without needing to understand the complexity of the system. He first must specify what information is to be made available in OGSA, where the type and structure of that information should be well defined. The user might also wish to specify certain properties and policies. This could include how the information is held, retention period, guarantee of delivery, persistency, and access control. A consumer could filter information of interest using a subscription topic for example, and it could support a query to further refine the events delivered to it through predicates defined in the query expression.

Some power-users might need a deeper understanding of the internal workings of the service. Following a request for information in a general (GMA-based) service, expected behavior is that a mediator capability is used to perform a registry/schema lookup and locate suitable sources of information. For long-term queries the mediator ensures that, as sources are dropped or new relevant sources come online, the subscribed consumers are updated. Mediation is also concerned with planning the distributed queries to the relevant producers as well as merging the results.

3.8.4 Properties

3.8.4.1 Security

Authentication and authorization rules for consumers and producers allow them to exchange information in a secure fashion. Discovering metadata information about what producers and consumers exist could also be subjected to security rules. Some applications (e.g. metering, Authentication, Authorization and Accounting) require that messages be secure (e.g., encrypted) while being delivered.

3.8.4.2 Quality of service

Various levels of QoS can be provided by OGSA information services, such as reliable, guaranteed delivery. WS-Reliability, defined by the OASIS WS-RM TC, provides these reliable message delivery properties.

3.8.4.3 Availability/Performance/Scalability

Information systems play a critical role in OGSA. Since almost every other capability in OGSA makes use of them, they need to be available at all times and to be especially tolerant of partial failure. Many clients of the information systems expect to receive information within a short enough time period and cannot afford to wait too long. High-performance systems are needed in this case. Because potentially a very large number of resources services and applications wish to produce and consume information, the system must also be scalable across wide-area networks.

3.8.5 Interaction with the rest of OGSA

3.8.5.1 Schema

Standards for events and event descriptions should exist. This enables a consumer to discover possible events it can consume, and allows the validation of query requests and the information produced. The use of recognized standards for events enables these events to be compared and related between potentially different sources within a VO. The possibility should exist to extend the schema to include user-defined events. The GGF DAMED working group is active is this area.

3.8.5.2 Notification

Notification mechanisms, or an extension of them, could be used to deliver events from producers to consumers once they have discovered each other, and between other components of the system.

3.8.5.3 Security

OGSA security services are needed for authentication/authorization between different components.

3.8.5.4 Distributed Query Processing

Used in the query planning phase of the mediation between producers and consumers of information.

3.8.5.5 Replication

Repositories of metadata (directories and registries) need to be distributed and replicated to avoid being a single point of failure and to improve scalability. Some temporary inconsistence between replicated copies might be acceptable in some situations. In this case the Information system needs to be robust in the event of inconsistent/out of date metadata

3.8.5.6 WS-Agreement

WS-Agreement allows the various components of an OGSA information system to negotiate "contracts" that fulfill certain criteria.

3.9 Context Services

To be filled in.

4 Security Considerations

This specification defines requirements for interfaces, behaviors and models used to structure and achieve interactions among services and their clients. While it is assumed that such interactions must be secured, the details of security are out of the scope of this specification. Instead, security should be addressed in related specifications that define how abstract interactions are bound to specific communication protocols, how service behaviors are specialized via policy-management interfaces, and how security features are delivered in specific programming environments.

5 Editor Information

Ian Foster
Distributed Systems Laboratory
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
Phone: 630-252-4619
Email: foster@mcs.anl.gov

Hiro Kishimoto
Grid Computing and Bioinformatics Laboratory
Fujitsu Laboratories
4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Phone: +81-44-754-2628
Email: hiro.kishimoto@jp.fujitsu.com

6 Contributors

We gratefully acknowledge the contributions made to this specification by Mario Antonioletti, Takuya Araki, Jamie Bernardin, Dave Berry, Abdeslem Djaoui, Shel Finkelstein, Jeffrey Frey, Dennis Gannon, Andrew Grimshaw, Bill Horn, Kate Keahey, Peter Kunszt, Simon Laws, Tan Lu, Fred Maciel, Susan Malaika, David Martin, Nataraj Nagaratnam, Jeffrey Nick, Dave Pearson, Benny Rochwerger, John Rofrano, Andreas Savva, Frank Siebenlist, Chris Smith, David Snelling, Latha Srinivasan, Ravi Subramaniam, Jem Treadwell, Jay Unger, Jeffrin Von Reich, Sachiko Wada, James Warnes, and Ming Xu.

References

1. Handle System, 2004. www.handle.net.
2. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. Web Services Architecture. W3C, Working Draft http://www.w3.org/TR/2003/WD-ws-arch-20030808/, 2003.
3. Cabrera, F., Copeland, G., Cox, B., Freund, T., Klein, J., Storey, T. and Thatte, S. Web Services Transaction (WS-Transaction), 2002. http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/ws-transaction.asp.
4. Czajkowski, K., Dan, A., Rofrano, J., Tuecke, S. and Xu, M. Agreement-Based Grid Service Management (WS-Agreement). Global Grid Forum, Draft, 2003.
5. Foster, I., Gannon, D. and Kishimoto, H. Open Grid Services Architecture Use Cases. Global Grid Forum OGSA-WG, Draft draft-ggf-ogsa-usecase-1, 2003.
6. Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002. www.globus.org/research/papers/ogsa.pdf.
7. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 15 (3). 200-222. 2001.
8. Kahn, R. and Wilensky, R. A Framework for Distributed Digital Object Services. Corporation for National Research Initiatives, 1995. http://www.cnri.reston.va.us/home/cstr/arch/k-w.html.
9. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S. and Kesselman, C. Grid Service Specification, 2002. www.globus.org/research/papers/gsspec.pdf.
10. http://www.cs.wisc.edu/condor/dagman/
11. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Grid Information Services for Distributed Resource Sharing. *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.
12. S.J. Chapin, D. Katramatos, J.F. Karpovich, and A.S. Grimshaw, “Resource Management in Legion,” *Journal of Future Generation Computing Systems*, vol. 15, pp. 583-594, 1999.