# Building a Grid of Grids: Messaging Substrates and Information Management

Geoffrey Fox, Shrideep Pallickara, and Marlon Pierce
{gcf, spallick, marpierc}@indiana..edu
Community Grids Lab, Indiana University
501 N. Morton Street, Suite 224
Bloomington, IN 47404-3730

## 1. Introduction: Classifying Grid Families

This article reviews our efforts to build science application Grids [Foster, 2003] using Web Service Architecture [Booth, 2004] principles. We review several aspects of the problem, including a) designing and integrating families of Grid Web Services; b) Grid messaging substrates, and c) developing client ("requester agent") managing environments such as computing Web portals.

The merger of Grid and Web Service standards first introduced by the Open Grid Service Architecture (OGSA) [Foster, 2002] opened up the possibility for creating and integrating disparate Grid families that formerly used incompatible technologies. As we outline in the following review, Web Services provide a unifying architecture for diverse Grid family groups. We may begin to think of Grids as collections of services assembled for specific tasks. These services include not only traditional Grid tasks such as accessing remote high performance computing resources, but also information, collaboration, and semantic services.

### *1.1. Basic Grid Family Groups*

Expanding on earlier classifications of Fox and Walker [Fox, 2003], we identify the following major Grid Web Service families:

**Data Grids**: these may range from High Energy Physics-style data grids [Allcock, 2002] to more database-oriented Web Service systems for bioinformatics [Goble, 2003a; Goble, 2003b], multiscale chemsitry [Myers , 2004; Pancerella, 2004], and other scientific digital libraries. The OGSA-DAI effort (www.ogsadai.org.uk), Storage Resource Broker [Rajasekar, 2003], and Project Mobius [Hasting, 2004] are other examples from the Grid community.

**Execution Grids:** these are traditional "remote operating system" grids that support secure remote command execution, access to computing resources for scientific computation, and remote file management. The Globus Toolkit (www.globus.org) is a well-known example.

**Desktop Grids:** the SETI@home activity is probably the most well-known example of this type of Grid, which seeks to harness idle computing power for pleasingly parallel

problems. More generally, desktop grids are sophisticated resource schedulers that harness idle computing power from diverse resources. From the Grid community, the Condor cycle scavenger scheduler is the best known example. Web Service style desktop grids are underdeveloped, probably in part because of the underdevelopment of the information services needed to manage them.

**Information and Collaboration Grids:** Virtual Organizations require information services for managing their constituent services. Basic Information Grids may be composed of straightforward information registries, such as UDDI. More sophisticated information management capabilities are the provenance of Semantic Grids such as MyGrid [Gobles, 2003a; Gobles, 2003b] and the Collaboratory for Multiscale Chemical Science [Myers, 2004; Pancerella, 2004].

**Sensor and Streaming Data Grids:** while Data Grids are typically concerned with managing archives of distributed data and in supporting data provenance and reconstruction, sensor grids are responsible for supporting real time, time-stamped data streams. Typically, raw sensor data must go through multiple stages of filtering, so these may be coupled to Execution Grids.

## *1.2. Collective and Derived Groups*

In addition to the above basic Grid styles, we may build more specialized Grid families, which incorporate elements of the basic Grid styles. The following list is not intended to be definitive, as the basic components may be combined in numerous ways, and the number of science domain-specific grids is unlimited.

**Audio/Video Grids:** Grid Web Services are normally associated with remote procedure calls and request/response semantics, but they may also be used to manage streaming data, such as in Sensor Grids. A/V Grids are a variation of this theme, using Web Services to manage collaborative sessions. The GlobalMMCS project (www.globalmmcs.org), discussed in more detail in a companion article, is an example of this. A/V Grids are thus a combination of collaboration and streaming grids.

**Geographical Information System (GIS) Grids:** these tend to be specific applications of Data, Information, and Sensor Grids that manage geographic data and information. They may in addition be combined with Execution Grid services for geo-processing applications, such as the generation of earthquake hazard maps. Grid Web Service versions of the Open Geospatial Consortium (www.opengeospatial.org) standards such as the Web Feature Service, Web Coverage Service, Web Map Service, and Sensor Grid Services are examples. Our efforts in this area are summarized by the information and resources at www.crisisgrid.org and are described in [Aktas, 2004].

**Visualization Grids:** these represent a combination of data, collaboration, streaming, and execution grids. Scientific visualization is a well-known high performance computing problem itself and quite often is applied to massive data sets. Efficient data streaming, demands of interactivity, and the usual requirement for supporting diverse, specialized clients make this a rich area. One particularly interesting area for research is that these

services push the boundaries of internet-speed messaging systems: millisecond latencies are acceptable, but longer delays degrade usability.

### *1.3. Grid of Grids Examples*

From all of these parts we may build comprehensive science application Grids. The examples shown at the top of Figure 1 illustrate several possible Grid systems and Virtual Organizations that we may draw from extensions to our work with SERVOGrid, but these may be applied to other problem domains. Science Grids for earthquake forecasting obviously can incorporate elements of all the Grid families that we have listed but are most strongly tied to Execution and Data Grids that are associated with scientific computing. The role of collaboration in these types of Grids has perhaps been underdeveloped, with the exception of NEESGrid, but data and result sharing, discussion forums, document sharing areas, and real time A/V support are all possible extensions.

On the other end of the spectrum, we have Emergency Preparedness and Response/Critical Infrastructure Protection (EPR/CIP) Grid organizations. These Grid organizations are not used by scientific research communities but instead by user groups such as policy makers, emergency responders, and even the general public. Finally, we may consider the value of "rapid response" Grids to specific, short term issues. Disasters such as the 2004 tsunami tragedy in Southeast Asia require the quick deployment of new Virtual Organizations consisting of diverse, pre-existing Grid family components: today scientists may want the quickest possible access to data for simulation, but hopefully in the future these Grids may be used to mitigate the loss of life and damage to property and the environment.

Members of these different Grid Web Service families may be united into the collection of Grid services that are needed to build application Grid systems. As we will discuss in this article, this unification may be done at two different levels. First, the "core services" substrate level must provide several different core services that constitute what we have termed the Grid messaging substrate [Fox, 2005a; Fox, 2005b]. Our approach, reviewed in the section "Internet-on-Internet" aims to provide a many-to-many messaging software implementation capable of providing the low level routing of SOAP and other messages. Grid service families of all sorts may utilize this messaging substrate layer.

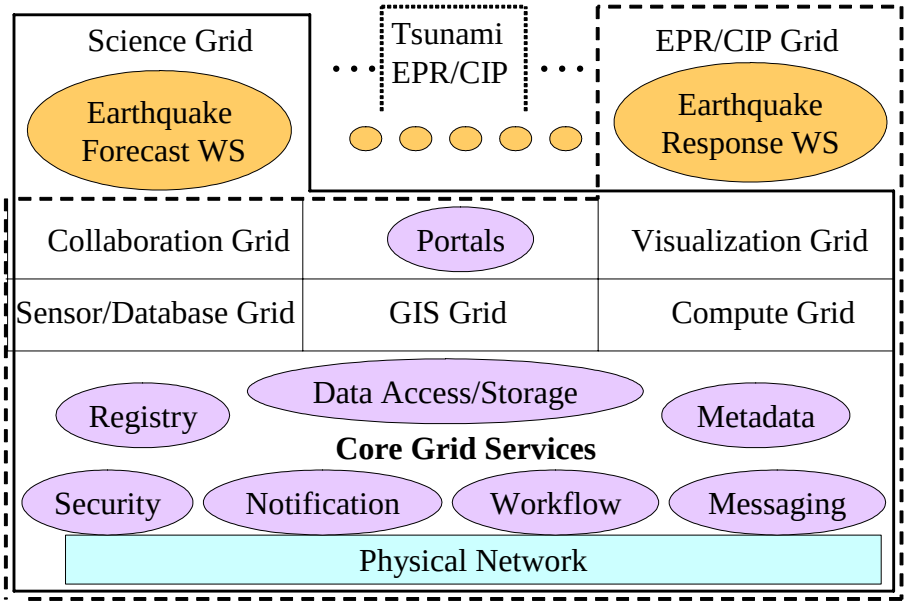


**Figure 1: Science, Critical Infrastructure Protection (CIP) and Emergency Preparedness and Response (EPR) Grids built as a Grid of Web Service (WS) Grids**

The second component of unification is described in the section "Context and Information Environment," or CIE. This layer is responsible for managing the information services and user environments necessary for composing Virtual Organizations out of Grids. In this article, we particularly highlight the role of portals in CIEs. Grid computing portals represent an important role in Web Service Architectures: they manage the lifecycles of requester agents, analogous to the Web Service hosting environment's management of service implementation instances. As we point out below, however, this has not been fully exploited in current portal systems: portals may manage Grid requester agents, but there is a missing family of Grid Web Services for building the portals themselves. We conclude this section by noting the implication that portal standardization efforts will lead to portals themselves acting as full-fledged service provider agents in a Grid of Grid environment.

### *1.4. Web Service Grids and Service Oriented Architectures*

The common feature of the different styles of Grids listed in the previous sections is the growing adoption of both Web Service standards and Web Service Architectures, and more generally Service Oriented Architecture (SOA) principles. In this section we review these subjects. We conclude primarily from this review that the Grid messaging system plays a crucial role in providing the unification of Grid families.

Web Services are essentially built on top of two major specifications: WSDL [Christensen, 2001] and SOAP [Gudgin, 2003]. WSDL describes the remote service's interface (its methods, their arguments, their return types), while SOAP is a messaging format that encapsulates the communications (in XML) between the services.

Technically, the two are decoupled: you can build services described with WSDL that use other messaging protocols besides SOAP.

One of the continuing sources of debate in scalable distributed systems is the proper management of state. State may be either internal to a specific service instance or it may result from the interaction with one or more clients. Distributed object systems allow for "stateful" interactions. That is, the remote object provides methods for modifying internal state data. SOA systems attempt to avoid this: clients may not directly manipulate the state of the remote service. Instead, if required, state may be maintained a conversation between the components. HTTP cookies are a familiar example of stateful conversations.

Distributed object systems have tended to focus on remote method invocations (the object-oriented equivalent of remote procedure calls). The remote object is intended to be used as if it were a local object: developers program their applications using client stubs that can be treated as local programming objects, but which must in reality make remote calls to the "real" object, often blocking until the method returns. This approach is suitable for tightly coupled environments like enterprise intranets, but it does not scale well to the loosely coupled situation seen, for example, in most Grid applications that need to run in several different, autonomous locations. SOAs instead focus on the message itself, rather than its invocation. Since interactions are normally stateless, message traffic between two components is assumed to be decoupled.

Most current approaches to Grid Web Services have adopted at least a partial SOA approach. Grid systems by their nature are not deployed on intranets. They instead involve collections of services offered many different service providers who have temporarily aggregated themselves into a "virtual organization." Message-based, stateless grids meet these requirements for loose coupling.

Message-oriented, loosely coupled systems do require a certain level of tolerance for latency. For Execution Grids, it is worth comparing internet messaging with classic MPI approaches. Classic MPI communications in cluster environments with good networking take place on the order of microseconds. In contrast, internet-based messaging between services has a time scale of milliseconds, so communications are inherently 100-1000 times slower. We therefore may backwardly define "loosely coupled" as meaning any system that does not need for its components to communicate at speeds less than a millisecond. For specific examples from our SERVOGrid efforts, see [Parker, 2004]. To summarize:

1. Couplings that involve autonomous application runs that take minutes to days to complete are good candidates for the messaging approach.
2. Couplings requiring microsecond latencies, such as parallel adaptive mesh refinement are not appropriate for the Grid Service approach. Instead, these should use the more appropriate MPI approach and be deployed on specific clusters.
3. In between these extremes, we have A/V collaboration and interactive visualization services, which require high speed internet messaging performance

(i.e. milliseconds). These may be implemented with the messaging approach but will push the boundaries of performance and Web Service messaging infrastructure. Some efforts in efficient Web Service messaging are described in [Fox, 2004b].

We thus see that message-oriented Grid systems allow for the integration of Grid systems in a scalable fashion. In the next section, we review our efforts to build a messaging substrate for Grid services.

# 2. Building Message-Based Services: Internet-on-Internet

Grid and Web Services that adopt the SOA approach need a messaging infrastructure for exchanging information. In particularly, the SOAs must enable the features that are part of the SOAP 1.1/1.2 specifications, particularly intermediate message header processing. They must also provide support for security, events and notification, reliable messaging, message routing, etc.

## *2.1. NaradaBrokering for Grid Service Messaging*

The Community Grids Lab has for several years been developing a core messaging infrastructure, NaradaBrokering (NB), that can handle sophisticated, many-to-many messaging over numerous transport protocols. Applications of NB have focused on collaborative systems such as Anabas and GlobalMMCS. We have, however, been adapting NB to implement and be compatible with Grid and Web Service specifications [Fox, 2005a; Fox, 2005b; Fox, 2004a].

A detailed description of NB is out of scope for this document; references available from http://www.naradabrokering.org provide full descriptions of the system; the reader should refer particularly to [Pallickara, 2003] for a basic introduction. In brief summary, NB acts as a topic-based, publish/subscribe system that enables communication between distributed components running on different hosts. Publishers and subscribers are autonomous, distributed computing components that have write and read access (respectively) to postings on various topics. Any given entity may be both a publisher and subscriber to a particular topic, and entities may have different roles in different topics. Both publishers and subscribers connect to message brokers, which are responsible for routing messages and maintaining topic lists. On top of this basic system, we may build a number of more sophisticated features: for example, brokers may be distributed, may change the protocols and ports used to transport messages, may guarantee delivery, may enforce security, may ensure once-only delivery and persistent storage of messages, and so on. In terms of SOAP 1.2 messaging, distributed brokers may act as relay nodes and may process SOAP headers.

The following table summarizes the most important (and relevant) NB features.

| | |
|---|---|
| Multiple transport support<br>In publish-subscribe | Transport protocols supported include TCP, Parallel TCP streams, UDP, Multicast, SSL, HTTP and HTTPS.<br>Communications through authenticating proxies/firewalls & |

| | |
|---|---|
| Paradigm with different Protocols on each link | NATs. Network QoS based Routing |
| Subscription Formats | Subscription can be Strings, Integers, **XPath** queries, **Regular Expressions**, **SQL** and tag=value pairs. |
| Reliable delivery | **Robust** and **exactly-once delivery** of messages in presence of failures |
| Ordered delivery | **Producer Order** and **Total Order** over a message type<br>**Time Ordered** delivery using Grid-wide **NTP based absolute time** |
| Recovery and Replay | **Recovery from failures** and disconnects.<br>**Replay** of events/messages at any time. |
| Security | **Message-level WS-Security** compatible **security** |
| Message Payload options | Compression and Decompression of payloads<br>Fragmentation a nd Coalescing of payloads |
| Messaging Related Compliance | Java Message Service (**JMS**) 1.0.2b compliant<br>Support for routing P2P **JXTA** interactions. |
| Grid Application Support | NaradaBrokering enhanced **Grid-FTP**. Bridge to the **Globus TK3**. |
| Web Service reliability | Prototype implementation of **WS-ReliableMessaging** |

**Table 1: NaradaBrokering features.**

One of the main goals of NB has been to virtualize communication connections, building a buffer layer between applications and the lower level networking infrastructure (TCP/IP, UDP, etc). This is motivated by two major concerns:

1. Collaborations and virtual organizations are often constrained or disabled by firewalls, NATs, and other networking features. Both Grid computing and Audio/Video collaboration are prominent examples of technologies crippled by current real networks.
2. The ideal protocol for a given collaboration may change over time, may be different for different recipients participating in the same activity, and may be a mixture of two or more "real" connections in the same "virtual" connection. For example, a participant behind a firewall/NAT in an AV collaboration may be unreachable by UDP, so we may need to tunnel the transmission through the NAT. GridFTP proxied through NB [Fox, 2005a] is an example of dual transmissions in the same virtual connection: control messages are sent back and forth over one connection (which may be re-implemented as SOAP in a hypothetical Web Services version), while transmissions of data go over a separate data channel.

NB-style virtual connections have another potential long term application. For web services requiring high (true millisecond) performance, SOAP over TCP/IP is notoriously inefficient. By virtualizing connection through NB brokers, transmitting messages over high speed UDP connections, and using the NB messaging fabric to provide in the

application layer typical TCP/IP features like reliability, SOAP messaging performance may be increase by 2-3 orders of magnitude [Fox, 2004b].

### 2.2. Integration of NaradaBrokering with Grids and Web Services

We have three potential integration strategies, summarized in Figure 2. First, there is the proxy approach, in which NaradaBrokers masquerade as remote Grid/Web Services. The brokers intercept the incoming messages and route them (unaltered) to the remote service. In this approach, the messages partake in NB Quality of Service features as long as they are within the boundaries between the end proxies. The connections between the proxies and the external endpoints are not covered by NB. One may adopt this approach when the remote web service is normally unreachable (due to firewalls) or when one wishes to hide the actual service's or client's URL. It also allows legacy implementations to be integrated with minimal alteration. We have adopted this approach for GridFTP tunneling and have also tunneled ordinary SOAP messages this way.

Second, we may follow the approach of directly integrating SOAP support into NB (middle of Figure 2). Here the services are also brokers and send SOAP messages directly over NB virtual connections. We may insert any number of brokers in between the two services, which may act as SOAP relay nodes [Fox, 2004a].

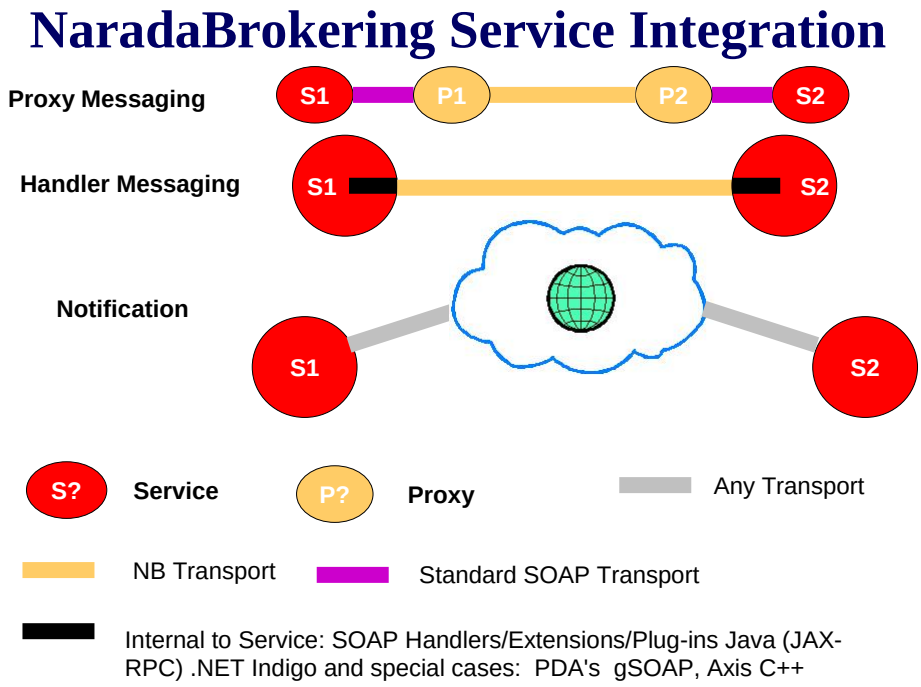


**Figure 2 Approaches to integrating Narada and Services.**

Finally, NB may be used to send events/notifications between services. Relevant specifications include WS-Notification and WS-Eventing. This is somewhat different than the first two approaches, and will coexist with them. In the first two cases, we have

pull-style messages: we are invoking the service's main interface and have an expected behavior from the service. In the last case, a service that is part of a cooperating group of services may need to notify its other partner services of various events (such as "I'm alive" or "I'm going away" in the simplest cases.) It is up to the event recipient to decide what to do with the message. This is an example of push messaging. Proxy messaging and handler messaging are alternatives to each other. Notification can by used by services in either case.

## 2.3. Reliable Messaging

Messaging in SOA-based grids often requires reliable messaging: the message originator usually needs to know if the message was actually received by the designated endpoint. Two competing specifications (WS-ReliableMessaging and WS-Reliability) provide straightforward solutions to this problem through acknowledgement messages. In both cases, the reliability Quality of Service capability is implemented as a SOAP header element that goes along with the normal SOAP message body.

Interestingly, the reliability approaches closely resemble the TCP/IP mechanisms, but in the application layer of the protocol stack. That is, reliability is an example of duplicating (previously considered) core networking functions in the application layer. Thus, we may use application layer reliability (implemented in SOAP messages sent over NB) to send messages over higher-performance UDP, eliminating the redundant TCP/IP features. We have implemented WS-ReliableMessaging and are in the process of integrating it with NB SOAP support.

## 2.4. Fault Tolerance

Reliable messaging is somewhat misnamed, as it does not define what should happen if messages actually fail to arrive; rather, it just is a mechanism for communicating failure or success. We may improve the implementation by providing some additional guarantees of delivery through fault tolerant messaging. Here, messages that partially or completely fail to reach their endpoints may be resent. This requires features such as persistent storage and once-only delivery. This feature is part of the core NaradaBrokering system (see Table 1) and may be applied to Web Service messaging.

## 2.5. Building the Internet-on-Internet (IOI)

In the previous section we have previewed an interesting and important development in Web Services: they are beginning to mimic the capabilities of the lower level network within their messages and messaging implementations. Reliability and fault tolerance are two prominent examples.

We refer to this as the "Service-Internet-on-Bit-Internet," or IOI. IOI is essentially a reimplementation of standard low-level networking capabilities at the higher application level. Typical IOI capabilities include several items listed previously:

1. Support for multiple transport protocols
2. Support for many different message delivery protocols, such as reliable delivery, once-only delivery, ordered delivery, and persistent delivery/delivery replay.
3. Application-level performance optimization through compression/decompression.

4. Fragmentation/coalescence of messages, which may be delivered over separate routes, in parallel. One may use this to do higher performance file transfers and to increase the reliability of large file transfers.
5. Security services, such as message encryption and authorization.
6. Time stamping services to assist with ordered delivery and replay.
7. Congestion control and dynamic best-route determination.
8. Performance monitoring.
9. Ad-hoc network support

All of these are traditional "low-level" networking capabilities that can be re-implemented in the NaradarBrokering messaging substrate layer, on top of traditional networking. That is, we may provide the above enhancements to existing Web Service implementations without requiring any modification to the existing services. We may to further and use broker topologies to mimic network topologies, creating overlay networks, "virtual private grids", firewalls, and demilitarized zones [Fox, 2005a].

# 3. Context and Information Environments (CIE)

In addition to the IOI capabilities, we may identify a number of other requirements needed to manage Grid Web Service organizations. That is, if implemented correctly, the IOI fabric may be invisible to the applications that run in it. Although an application developer may conceivably want to directly touch this layer, this would not be the usual practice. Instead, they would specify the desired Quality of Service and let the IOI fabric implement this.

There are a number of higher level services and capabilities that do not belong in the IOI layer. As a general rule, these are services that extend (rather than mimic) the lower level networking features and are more specifically needed for Web service management. Typical examples include service information and metadata management. We refer to this collection of capabilities as the Context and Information Environment.

The basic problem is the following: which service or sequence of services actually accomplishes my desired result? In our Grid of Grids approach, there are various service collections that provide the basic capabilities. There are execution services for running remote applications and orchestrating cooperating services, there are data grid services that provide access to remote data, there are collaboration services, and so on. These services are used to build "useful" grids and are maintained in an IOI fabric that is responsible for the messaging infrastructure and the related qualities of service. The relationship of the IOI to the CIE is shown in Figure 3.

In the Grid of Grids approach, the services must share higher level information about themselves. This is commonly called metadata. We may extend this to the problem of building Information Grids (and their derived group, Knowledge Grids) that build on the more traditional Execution and Data Grids.

The problem in the Grid and Web Service world is that the metadata/information problem for describing services is very confused.

- The WS-Interoperability (WS-I) consortium has essentially endorsed UDDI, but it has a number of problems (rigid data models that don't describe science Grids very well, no mechanisms for dynamic service discovery, and no way to clean up obsolete information, to name a few examples.)
- The Semantic Web has worked for a number of years on metadata descriptions and more sophisticated knowledge management, but tends to get ignored by the Web Service community, at least in the United States. Recent developments in the UK e-Science program and elsewhere may eventually reverse this trend.
- The Grid community has two competing concepts (the WSRF specification suite and WS-GAF) on managing metadata, particularly when it concerns dynamically evolving resource state information.
- From OASIS we have the WS-DistributedManagement suite of specifications and WS-MetadataExchange.

We thus view this as an important area for future development. Our focus here has been to take a particular domain area (the GIS Grid) and attempt to address its specific needs using general approaches, starting from extensions to UDDI that support leasing and domain-specific extensions. This work is described in [Aktas, 2004].

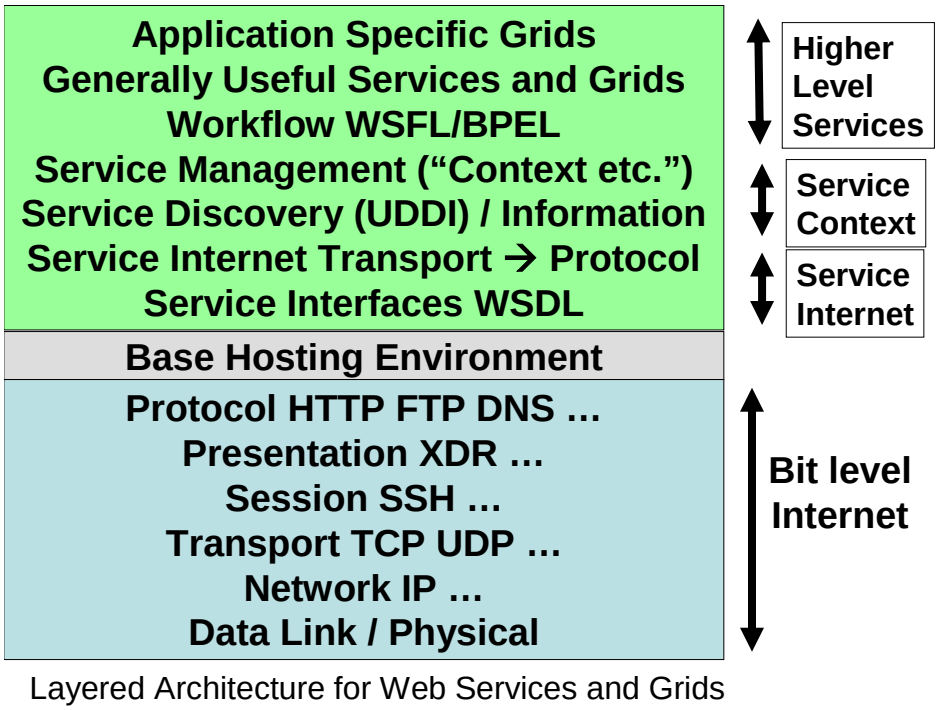


**Figure 3 Layered architecture for Web Services and Grids.**

In our own experience, information services are closely tied to user environments and so have been closely tied to computing portal systems. These types of information systems are often bound directly with the portal container and portal services. As we discuss in the section below, this decision has been workable but leads to design difficulties as portal systems evolve.

## 3.1. Component-Based Grid Portals

Grid Computing Portal development through early 2002 is comprehensively covered in [Hey, 2002]. However, major developments in portal systems, over and above advances and transitions in Grid computing infrastructure, have occurred since the original comprehensive survey of *Currency and Computation.* We may characterize the current state of the field as being at the end of one important revolution (standardized, reusable portlet components) and at the verge of another: service-based containers. It is the purpose of this section to survey the first revolution and hopefully preview the importance (but perhaps not inevitability) of the second.

The Grid Computing Environments (GCE) research group of the Global Grid Forum represented an extensive cross section of Web and client developers for building client environments for computational Grids. These group meetings resulted in papers collected in [Hey, 2002]. Particularly important here is the survey and synthesis of Fox, Gannon, and Thomas [Fox, 2002], which identified portal research classifications and many commonalities between different projects. This formal classification reiterated the more informal conclusions of the GCE community. The key problem was that there was no technically elegant way of sharing portal capabilities between groups. Reinvention was inevitable.

In many ways, the articles of [Hey, 2002] represent a formal closing to an era, as the first step towards solving the interoperability and reusability problems was already at hand. Several projects, notably the Alliance Portal, CHEF, GridPort, and GridSphere, adopted *portlet* approaches to building their (Java-based) portal systems. These generic portal efforts were complemented by application portal efforts, including NEESGrid, CMCS, and SERVOGrid. For an overview of these early efforts to build component-based Grid portals, see [Pierce, 2002] and [Gannon, 2004].

The impact of emerging portlet standards such as JSR 168 [Abdelnur, 2003] and WSRP [Kropp, 2003] has had a sizable impact on the computational Grid portal community. These standards allow portal developers to share and reuse standard components. Portlets are very natural candidates for Web Service clients, and portlet containers can be viewed as the counterpart to Web Service containers. Furthermore, portlet containers are excellent at managing stateful clients. We summarize these two standards below.

- JSR168 defines a standard *local* portlet API in Java. JSR 168 compatible portlet engines can load and run each other's portlet code. Examples of JSR168-compatible portlet containers include WebSphere, Jetspeed2, uPortal, and GridSphere.
- WSRP defines a standard *remote* portlet API in WSDL. That is, portlets run separately and remotely from their container.

These two standards are compatible: JSR168 compatible portlets may act as proxies/web service clients to the remote WSRP portlets. Both standards have shortcomings (some possibly serious) and both have reference implementations that will need improvement, but we expect portlet container developers and vendors to continue to support them nonetheless.

## 3.2. Portlet Standard Shortcomings

Portlet standards such as JSR 168 represent workable but limited means of sharing components between different (Java-based) portlet containers. Criticisms of this standard are numerous, and summarized briefly below.

- It does not support inter-portlet communication. There is no standard way for portlets to share data or to send and receive messages with other portlets.
- It does not define how portlets may access standard services used by their parent container.
- More importantly, and crucially, it does not define patterns and mechanisms for extending portlets to provide these services.

The first criticism has the most immediate consequences for Grid portlets: following login, the portlets need a way to access in-memory proxy credentials, such as may be obtained from a MyProxy server. The difficulty arises from the nature of the standard, in which portlets can be distributed among several different (Java) servlet contexts. In the Java Servlet specification, these contexts have separate classloaders and so do not normally share data. Exceptions are possible: classes and their associated data may be accessed through "common" and "shared" areas (literally, /common/lib and /shared/lib).

The problem with such approaches is that they break the portability and standardization of the portlets. It is possible to build portable versions of these services, as has been done by the Open Grid Computing Environments (OGCE) group, but the interface is not standard and invariably will be reinvented by other groups. Alternatively, portlet container providers may rely upon their richer but more proprietary service APIs to provide the missing service. The key problem is not really missing data exchange services or messaging services, but instead the missing extensibility to provide service definition and instantiation capabilities within the portal (the third criticism, above). This had led some projects (notably, Sakai) to move more heavily toward WSRP.

WSRP is no panacea. WSRP allows portlets to be decoupled from their containers and to be written in other languages besides Java. But from the Grid perspective, it still has shortcomings:

- Identity-related services based on strong authentication are central to Grids, so Web Service security methods must be adopted for establishing distributed session identity.
- Problems in distributed session management must be addressed. External context registries are needed to manage stateful interactions arising between multiple WSRP tools and their remote containers.
- Group definitions and management are not standardized. This is particularly important for Grid portals, since the groups and organizations defined within the portal should correspond to the externally defined Virtual Organizations.

These problems will be eventually addressed by the community, but, as we argue in the next session, extrapolating solutions from current portal environments will lead to specialized solutions.

### 3.3. Portlet Containers as Service Consumers

As we have reviewed in the previous section, portlet reusability and interchangeability are limited by the portlet container. Portlets that comply with standards such as JSR 168 must either have limited capabilities or else must make non-standard extensions. WSRP portlets have a richer interface but must ultimately suffer from similar problems, as XML data objects (such as group identity and roles) must be standardized.

Solutions to the problems raised in the previous section merit some thought. The group access controls of Grid portlets and services will serve as a representative example. Currently, portal containers typically provide their own internal group management subsystems. This must often be mapped to external group services, such as services that provide Virtual Organization definitions. The immediate problem is that this mapping (which also may not be complete) must be done on a one-time basis for every portal container and every VO group service, leading to the usual combinatorial problems. WSRP complicates these matters, as portlet containers must also share their internal group definitions with the remotely running portlets. The individual solutions to this problem are not difficult, but no comprehensive solution emerges from these approaches. Instead, by refactoring the portlet container to be lightweight and to use a simple, Web Service-based interface for group management, we may avoid the combinatorial explosion. Portals in this case will rely on distinct services that may be implemented externally to the container and that may be easily shared with other services in the distributed system.

Portlet containers themselves are thus an excellent candidate for standardization through Web Services. To date, there have been to our knowledge no efforts to standardize common container services as Web Services. Such services include the following:

**Logging**: As we have seen, the distributed nature of portlet applications means that their logs will be scattered, making management of distributed sites difficult. Standard logging services are needed to coalesce the logged information.

**Authentication:** login and authentication are hallmarks of computational portals. Several portal systems provide pluggable, extensible authentication modules but the actual interface is not standardized, requiring one-time solutions for each container if one wants to integrate portal and Grid logins into a singles-sign on system.

**User account management:** Currently, a user identity description is part of the JSR 168 specification, but this must be mapped to the legacy container identity description. Decoupled WSRP-style Grid portal containers will require this.

**Group management:** Typically, portal containers provide their own internal group management system. There is a very obvious bridge to externally defined Virtual Organizations, but currently VOs themselves are somewhat loosely defined. This is a potentially valuable external service as external, federating group services such as Shibboleth/SAML and WS-Federation become more closely associated with the Grid community.

**Authorization and Roles:** Related to group and user management, these services define the privileges and restrictions that may be associated with certain users and groups. They are distinct as, for example, the authorizations associated with a fixed group may change over time.

**Layout and Display Management:** All portal containers provide services for managing user display. This is essentially manages the aggregation of personalized user content. This service controls the portlets that show up on the user's display after login. Display services must provide support for different user agent devices.

**Portlet Information Services:** One of the more straightforward applications of service-oriented portals is to publish the availability of "live" portlet instances of incorporation into existing services.

**Content Management:** These services associate various layout frames with specific pieces of portlet content. Content management must interact with layout/display and authorization services.

Again, we note that these services are not new: all containers must provide them at some level. The problem is that the implementations have stressed tight integration, making it difficult to disentangle highly interdependent services. By adopting clean, service oriented design principles, it is hoped that next-generation portlet containers will be much lighter-weight. We note however that there is no inevitability in this evolution. Portal development groups still focus heavily on container development, rather than service component or portlet development.

### *3.4. Portlet Containers as Services*

We conclude this section with an observation of the implications of Portlet Web Services. The architecture of portlet containers has become relatively standardized, even if individual specifications continue to evolve. We have generally espoused the view in our discussions that portals serve as the container environment for managing client requester agents: portal containers aggregate Web Service clients and content. However, container decoupling standards such as WSRP provide an additional possibility for the containers to act also as service providers: the aggregated content of the portal may be directly accessed through traditional browsers by users, or it may be accessed through Web Service clients. We anticipate this will pose potentially difficult problems in trust and delegation for traditional Grid application portals built on Execution Grid service families, making this a rich problem for further study.

## 4. Summary

In this article we have reviewed Grid Web Service families that may be used to build composite Grid organizations, or "Grid of Grids." As we have discussed, there are two prominent areas for integrating Grid families: at the messaging substrate layer and at the information and context management layer. For the former, we have reviewed the NaradaBrokering framework and our efforts to use it to integrate the messaging systems

of different Grid services. For the latter, we have focused primarily on portal environments for managing Web Service requester agents.

The key role of messaging in Grid systems is often overlooked. While standard activities such as Web Services Resource Framework provide means for services to communicate with each other about changes in the state of the resources they manage, they do not consider the use of the messaging system for transferring directly the communications between requester and provider agents, perhaps through numerous SOAP intermediaries and relay nodes. The SOAP 1.2 specification makes several changes on the earlier model to emphasize the messaging nature of Web Services, and we are attempting to exploit this in our current research work.

Computing portals have undergone a great deal of standardization over the last several years, but much work needs to be done. We may view the current standardization around reusable portlet components as more clearly defining the requirements for future work. Specifically, portlets have more clearly defined the portlet container. We view standardizing the container's service interfaces as the important next step for the Grid portal community: future portal containers will be lightweight, with decoupled service implementations.

## 5. Acknowledgements

This article refers to work that has been funded by the National Science Foundation's National Middleware Initiative, the United Kingdom's Open Middleware Infrastructure Initiative, and the National Air and Space Administration's Advanced Information System Technology office. We gratefully acknowledge numerous collaborations with members of the Open Grid Computing Environments (www.collab-ogce.org) and QuakeSim/SERVOGrid (quakesim.jpl.nasa.gov) teams that have influenced the material presented here.

## 6. References

[Abdelnur, 2003] Abdelnur, A., Chien, E., and Hepper, S., (eds.) (2003), *Portlet Specification 1.0*. Available from http://www.jcp.org/en/jsr/detail?id=168.

[Allcock, 2002] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke "Data Management and Transfer in High Performance Computational Grid Environments. "Parallel Computing Journal, Vol. 28 (5), May 2002, pp. 749-771.

[Aktas, 2004] Mehmet Aktas, Galip Aydin, Andrea Donnellan, Geoffrey Fox, Robert Granat , Lisa Grant, Greg Lyzenga, Dennis McLeod, Shrideep Pallickara, Jay Parker, Marlon Pierce, John Rundle, Ahmet Sayar, and Terry Tullis, "iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services." Submitted for publication in Special Issue of Pure and Applied Geophysics (PAGEOPH ) for Beijing ACES Meeting July 2004.

[Booth, 2004] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D., *Web Services Architecture*, W3C Working Group Note 11 February 2004. Available from http://www.w3.org/TR/ws-arch/.

[Christensen, 2001] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S. (2001), *Web Service Description Language (WSDL) 1.1*. W3C Note 15 March 2001.

[Cox, 2003] Cox, S., Daisey, P., Lake, R., Portele, C., and Whiteside, A. (eds) (2003), *OpenGIS Geography Markup Language (GML) Implementation Specification.* OpenGIS project document reference number OGC 02-023r4, Version 3.0

[Donnellan, 2004a] Andrea Donnellan, Jay Parker, Geoffrey Fox, Marlon Pierce, John Rundle, Dennis McLeod Complexity Computational Environment: Data Assimilation SERVOGrid 2004 Earth Science Technology Conference June 22 - 24 Palo Alto.

[Donnellan, 2004b]Andrea Donnellan, Jay Parker, Greg Lyzenga, Robert Granat, Geoffrey Fox, Marlon Pierce, John Rundle, Dennis McLeod, Lisa Grant, Terry Tullis The QuakeSim Project: Numerical Simulations for Active Tectonic Processes 2004 Earth Science Technology Conference June 22 - 24 Palo Alto.

[Foster, 2003] Foster, I. and Kesselman, C. (eds), The Grid 2: Blueprint for a New Computing Infrastructure. (Morgan Kaufmann, 2003).

[Foster, 2002] I. Foster, C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration." Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002.

[Fox, 2005a] Geoffrey Fox, Sang Lim, Shrideep Pallickara, Marlon Pierce, "Message-Based Cellular Peer-to-Peer Grids: Foundations for Secure Federation and Autonomic Services." Future Generation Computer Systems 21 (2005) 401-415.

[Fox, 2005b] Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, Building Messaging Substrates for Web and Grid Applications to be published in special Issue on Scientific Applications of Grid Computing in Philosophical Transactions of the Royal Society of London 2005.

[Fox, 2004a] Fox, G., Pallickara, S., and Parastatidas, S. (2004), *Towards Flexible Messaging for SOAP Based Services*, in Proceedings of the IEEE/ACM Supercomputing Conference, November 2004. Pittsburgh, PA.

[Fox, 2004b] Geoffrey Fox Harshawardhan Gadgil, Shrideep Pallickara, Marlon Pierce, Robert L. Grossman, Yunhong Gu, David Hanley, Xinwei Hong, "High Performance Data Streaming in Service Architecture," Community Grids

Laboratory Technical Report, July 2004. Available from http://grids.ucs.indiana.edu/ptliupages/publications/HighPerfDataStreaming.pdf.

[Fox, 2003] Geoffrey Fox and David Walker, "e-Science Gap Analysis." Report prepared for the United Kingdom e-Science Program, 2003. Available from http://grids.ucs.indiana.edu/ptliupages/publications/GapAnalysis30June03v2.pdf.

[Fox, 2002] Geoffrey Fox, Dennis Gannon, and Mary Thomas, "A Summary of Grid Computing Environments," Concurrency and Computation: Practice and Experience, Vol. 14, No. 13-15, pp. 1035-1044.

[Gannon, 2004] D. Gannon, J. Alameda, O. Chipara, M. Christie, V. Dukle, L. Fang, M. Farrellee, G. Fox, S. Hampton, G. Kandaswamy, D. Kodeboyina, S. Krishnan, C. Moad, M. Pierce, B. Plale, A. Rossi, Y. Simmhan, A. Sarangi, A. Slominski, S. Shirasuna, T. Thomas Building Grid Portal Applications from a Web-Service Component Architecture to appear in. special issue of IEEE distributed computing on Grid Systems.

[GML] Simon Cox, Paul Daisey, Ron Lake, Clemens Portele, and Arliss Whiteside, *OpenGIS Geography Markup Language (GML) Implementation Specification Version 3.00*. Available from http://www.opengis.org/docs/02-023r4.pdf.

[Goble, 2003a] C.A. Goble *The Grid needs you! Enlist now.* Invited paper ODBASE2003, 2nd International Conference on Ontologies, Databases and Applications of Semantics, 3-7 November 2003, Catania, Sicily (Italy)

[Goble, 2003b] C.A. Goble, S. Pettifer, R. Stevens and C. Greenhalgh *Knowledge Integration: In silico Experiments in Bioinformatics* in The Grid: Blueprint for a New Computing Infrastructure Second Edition eds. Ian Foster and Carl Kesselman, 2003, Morgan Kaufman, November 2003.

[Gudgin, 2003] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.-J., and Nielsen, H. (2003), SOAP Version 1.2 Part 1: Messaging Framework. W3C Recommendation 24 June 2003. Available from http://www.w3c.org/TR/soap12-part1/.

[Hastings, 2004] Shannon L Hastings, Stephen Langella, Scott Oster, Joel H Saltz, "Distributed Data Management and Integration Framework: The Mobius Project", Proceedings of the Global Grid Forum 11 (GGF11) Semantic Grid Applications Workshop, 2004, pp. 20-38.

[Hey, 2002] Anthony Hey and Geoffrey Fox, eds. *Concurrency and Computation: Practice and Experience,* Vol. 14, No. 13-15 (2002).

[Kropp, 2003] Alan Kropp, Carsten Leue, and Rich Thompson, eds., "Web Services for Remote Portlets Specification." Approved OASIS Standard, August 2003.

Available from http://www.oasis-open.org/committees/download.php/3343/oasis-200304-wsrp-specification-1.0.pdf

[Myers, 2004] James D. Myers et al, "A Collaborative Informatics Infrastructure for Multi-Scale Science." Published in the proceedings of the Challenges of Large Applications in Distributed Environments (CLADE) Workshop, June 7, 2004, Honolulu, HI. Available from http://scidac.ca.sandia.gov/Get/File-886/CLADE_2004_3_28.PNNL-SA-40934.pdf.

[Pallickara, 2003] Shrideep Pallickara and Geoffrey Fox NaradaBrokering: A Distributed Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids in Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003, Rio Janeiro, Brazil June 2003

[Pancerella, 2004] Carmen Pancerella, et al, *Metadata in the Collaboratory for Multi-Scale Chemical Science.* Published in the proceedings of the 2003 Dublin Core Conference: Supporting Communities of Discourse and Practice - Metadata Research and Applications in Seattle, WA, 28 September - 2 October 2003. Available from *http://scidac.ca.sandia.gov/Get/File-856/401_Paper67.pdf.*

[Parker, 2004] Jay Parker and Marlon Pierce, "Exploring Coupling Methodologies for the Development of Cross-Scale Tools." Tech Report for AIST CCE Project, 2004.

[Pierce, 2002] Marlon Pierce, Choonhan Youn, Ozgur Balsoy, Geoffrey Fox, Steve Mock, and Kurt Mueller, Interoperable Web Services for Computational Portals. Published in Proceedings of Supercomputing 2002 November 2002.

[Rajasekar, 2003] Arcot Rajasekar, Michael Wan, Reagan Moore, Wayne Schroeder, George Kremenek, Arun Jagatheesan, Charles Cowart, Bing Zhu, Sheau-Yen Chen, Roman Olschanowsky, "Storage Resource Broker - Managing Distributed Data in a Grid," Computer Society of India Journal, Special Issue on SAN, Vol. 33, No. 4, pp. 42-54 Oct 2003.