Evolution of grid computing architecture and grid adoption models

by J. Joseph
M. Ernest
C. Fellenstein

During recent years, we have witnessed a major paradigm shift in distributed computing principles, with a focus towards service orientation, open standards integration, collaboration, and virtualization. One particular area of interest centers on the evolution of grid computing principles into the mainstream of distributed computing and Web services. In this paper, we focus our analysis on this evolution and the significance of achieving some form of standardization of grid-computing architecture principles. This paper presents the technology standards that are driving major grid initiatives and explains in simple terms how these standards and technologies are aligned with the IBM on demand business concepts. In addition, we discuss the recent Web services specifications related to stateful resources (i.e., resources whose behavior is defined with respect to their underlying state) and how these standards relate to grid computing. We conclude with discussions exploring major aspects of grid-computing adoption models and some significant attributes that influence the transformation, collaboration, and virtualization features of these models.

A distributed system consists of a set of software agents that work together to implement some intended functionality. Because the agents in a distributed system do not operate in a uniform processing environment, they must communicate by protocol stacks that are intrinsically less reliable than direct code invocation and shared memory. Grid-computing principles focus on large-scale resource sharing in distributed systems in a flexible, secure, and coordinated fashion. This dynamic coordinated sharing results in innovative applications making use of high-throughput computing for dynamic problem solving.

In this paper, we analyze the core initiatives that will enable grid computing to evolve into a service-oriented computing platform. These initiatives are based on the concepts of merging grid architectures with service-oriented architectures (SOAs). Our approach is based on the emergence of open standards platforms to build resource collaboration models, combined with the ability to simply construct dynamic applications that leverage virtualized resources. This approach allows for the implementation of well-defined grid adoption models and their application in on demand business environments.

One of the basic requirements of a grid system is the ability to provide the high-level quality of service (QoS) needed for a satisfactory user experience. Thus, QoS validation must exist as a basic feature in any grid system, as measured by the available resource metrics. These metrics include response time measurements, aggregated event performance monitoring and measurements, security fulfillment, resource scalability, availability, autonomic features, fail-over mechanisms, and networking services (in-

0018-8670/04/$5.00 © 2004 IBM

cluding network circuit provisioning and event correlation).

There are important architectural implications related to the grid adoption model, in light of the fact that all distributed systems require developers (of both infrastructure and applications) to consider the unpredictable latency of remote access. This means developers must take into account issues of concurrency and the possibility of partial failure.[1] Grid computing aims to resolve these complexities of distributed computing through a well-defined architecture that is aligned with SOA, autonomic-computing principles,[2] and open standards for integration and interoperability.

The remainder of the paper is structured as follows. In the next section, we review grid architecture evolution and its alignment with SOAs. We then detail the standards that are driving this architecture platform, especially as they relate to IBM's on demand business[3] and grid[4] initiatives. In the last section, we discuss the grid adoption model, including the standards and attributes that are driving the capabilities of this adoption model. Some material presented in this paper was taken from Reference 5.

Evolution of grid architecture. In 1998, it was stated that "a computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."[6] This definition was primarily centered on the computational aspects of grids. Later iterations broadened this definition with more focus on coordinated resource sharing and problem solving in multi-institutional virtual organizations.[7]

The grid problem. Grid computing has evolved into an important discipline within the computer industry by differentiating itself from distributed computing through an increased focus on resource sharing, co-ordination, manageability, and high performance. The focus on resource sharing is called *the grid problem*, which can be defined as the set of problems associated with resource sharing among a set of individuals or groups. This sharing of resources, ranging from simple file transfers to complex and collaborative problem solving, is accomplished under controlled and well-defined conditions and policies. In this context, the critical problems are resource discovery, authentication, authorization, and access mechanisms. Resource sharing is further complicated when a grid is introduced as a solution for utility computing,[8] where commercial applications and resources become available as shareable and on-demand resources. This concept of commercial on-demand utility grid services adds new, more difficult challenges to the already complicated grid problem list,[7] including service level features, accounting, usage metering, flexible pricing, federated security, scalability, and open-ended integration.

Virtual organizations. A *virtual organization* (VO) is a dynamic group of individuals, groups, or organizations who define the conditions and rules for sharing resources. The concept of the VO is the key to grid computing. All VOs share some characteristics and issues, including common concerns and requirements that may vary in size, scope, duration, sociology, and structure. The members of any VO negotiate the sharing of resources based upon the rules and conditions defined by the VO, and the members then share the resources in the VO's constructed resource pool.[9]

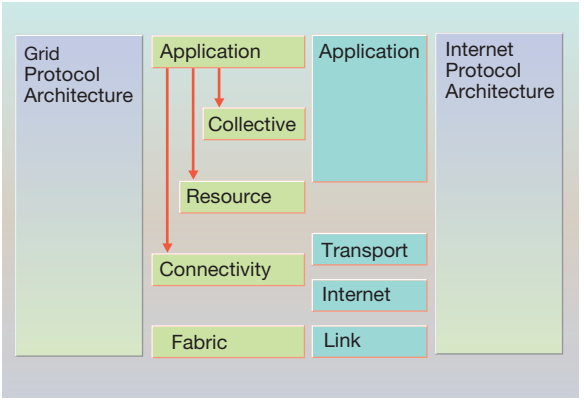
Assigning users, resources, and organizations from different domains to a VO remains one of the key technical challenges in grid computing today. This task includes the determination of a definition of resource discovery mechanisms, such as identification and application of appropriate resource-sharing methods, specification and application of rules and conditions for member assignment, security federation or delegation, and access control among the participants.

Common characteristics typically exist among the competing and sometimes distrustful participants that contributed to the formation of the VO. These may include the following:[7]

1. Concerns and requirements exist concerning resource sharing.
2. Resource sharing is conditional, time-bound, and rules-driven.
3. The collection of participating individuals and/or institutions is dynamic.
4. Sharing relationship among participants is peer-to-peer in nature.
5. Resource sharing is based on an open and well-defined set of interactions and access rules.

The above characteristics and requirements lead to the definition of an architecture for VO establishment and management and for resource sharing among participants. We examine this architecture in the following section, exploring its scope and characteristics.

Figure 1 The layers of the grid architecture and its relationship to the Internet Protocol architecture



Reprinted with permission of Ian Foster

Grid architecture model. A new architecture model and technology has been developed for the establishment and management of cross-organizational resource sharing. This new architecture, called *grid architecture*, identifies the basic components of a grid system. The grid architecture defines the purpose and functions of its components, while indicating how these components interact with one another.[7] The main focus of the architecture is on interoperability among resource providers and users in order to establish the sharing relationships. This interoperability, in turn, necessitates common protocols at each layer of the architectural model, which leads to the definition of a grid protocol architecture as shown in Figure 1.

This protocol architecture defines common mechanisms, interfaces, schema, and protocols at each layer, by which users and resources can negotiate, establish, manage, and share resources. Figure 1 shows the component layers of the grid architecture and the capabilities of each layer. Each layer shares the behavior of the underlying component layers. The following describes the core features of each of these component layers, starting from the bottom of the stack and moving upward.

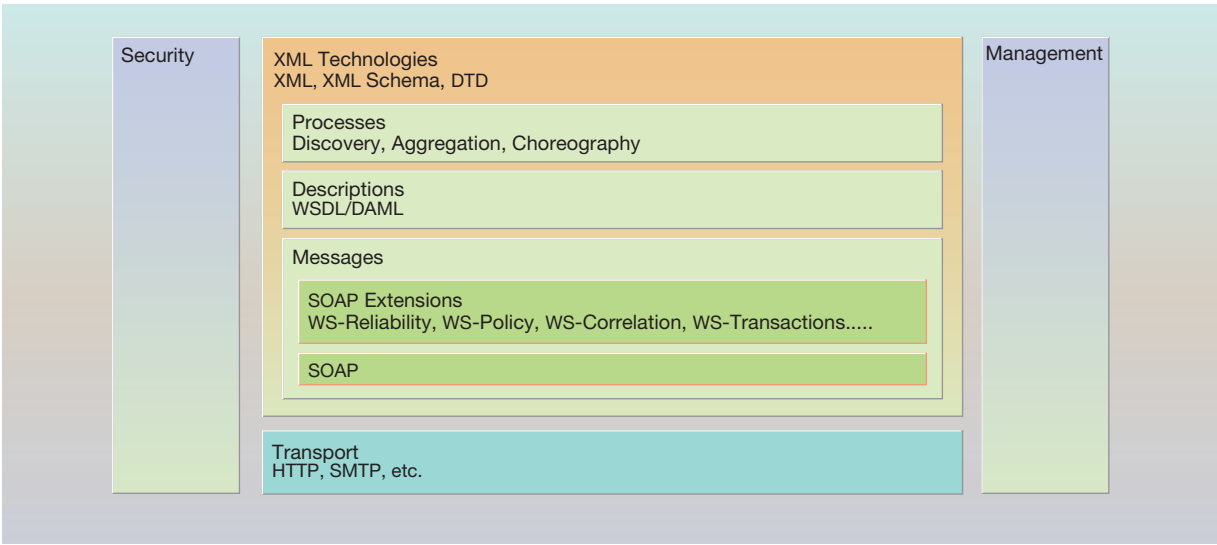
- *Fabric layer*—The fabric layer defines the interface to local resources, which may be shared. This includes computational resources, data storage, networks, catalogs, software modules, and other system resources.
- *Connectivity layer*—The connectivity layer defines the basic communication and authentication protocols required for grid-specific networking-service transactions.
- *Resource layer*—This layer uses the communication and security protocols (defined by the connectivity layer) to control secure negotiation, initiation, monitoring, accounting, and payment for the sharing of functions of individual resources. The resource layer calls the fabric layer functions to access and control local resources. This layer only handles individual resources, ignoring global states and atomic actions across the resource collection pool, which are the responsibility of the collective layer.
- *Collective layer*—While the resource layer manages an individual resource, the collective layer is responsible for all global resource management and interaction with collections of resources. This protocol layer implements a wide variety of sharing behaviors using a small number of resource-layer and connectivity-layer protocols.
- *Application layer*—The application layer enables the use of resources in a grid environment through various collaboration and resource access protocols.

Thus far, our discussions have focused on the grid problem in the context of a virtual organization and the proposed grid computing architecture as a suggested solution to this problem. This architecture is designed for controlled resource sharing with improved interoperability among participants. In contrast, emerging architectures help the earlier-defined grid architecture quickly adapt to a wider (and strategically important) technology domain.

SOA model. A service-oriented architecture (SOA) is a specific type of distributed system framework which maintains agents that act as "software services," performing well-defined operations. We define a SOA as a loosely coupled architecture with a set of abstractions relating to components granular enough for consumption by clients and accessible over the network with well-defined policies as dictated by the components.

Thus, a service acts (in some manner) as a user-facing software component of an application or resource. This paradigm of functionality enables the users of that application (or resource pool) to be concerned only with the operational description of the service. In addition, SOA stresses that all services have a network-addressable interface and communicate via standard protocols and data formats called messages.

Figure 2 The Web Services Architecture (WSA) and the respective layers in the WSA framework



Reprinted with permission of the Pearson Technology Group.

The Web Services Architecture (WSA) helps to enable and define SOA, where services interact by exchanging XML (Extensible Markup Language) messages, as shown Figure 2. The main characteristics of the WSA are:

- It is based on XML technologies such as XML Information Model,[10] XML Base, and XML Schema.[11]
- It is independent of the underlying transport protocols (e.g., HTTP [HyperText Transfer Protocol] or SMTP [Simple Mail Transfer Protocol]) and the transport selected as a runtime binding mechanism.
- It uses XML message exchanges, while providing the extensibility model for this message exchange pattern to adapt to various message interchange requirements (e.g., security, reliability, correlation and privacy.) These interoperable messages could be created using Simple Object Access Protocol[12] (SOAP) and SOAP extensions. SOAP extension mechanisms and XML information models are used to construct Web services extensions.
- Service capabilities are described using description languages such as Web Services Description Language (WSDL).
- It uses the service discovery, workflow choreography, and transaction/state management that are built on the XML capabilities and lower-layer specifications in the architecture layer.

The emergence of the SOA concept helps grid resources to advertise their capabilities through standard interfaces defined as part of their service extensions. This enables grid users to integrate the resources through open-standards interfaces. In addition, the operational functions of each layer in the grid architecture can be abstracted to enable easier integration across all the layers. This service abstraction of each layer of the grid architecture is shown in Figure 3.

In the next section, we discuss the architectural standardization initiatives for the alignment of the SOA and the grid architecture. This discussion focuses on how these architectures complement each other and on the major open standards that assist this coordination and its evolution.

Defining an open-standards platform for grid computing. In order to achieve true distributed resource sharing across heterogeneous and dynamic VOs, grid-computing technologies require several improvements in alignment with other computing technologies. In the early days of grid computing, a number of custom middleware solutions were created to solve the grid problem, but this resulted in non-interoperable solutions and problematic integration among the participants. As stated earlier, the new wave of

Figure 3 Service-oriented grid architecture with service interfaces

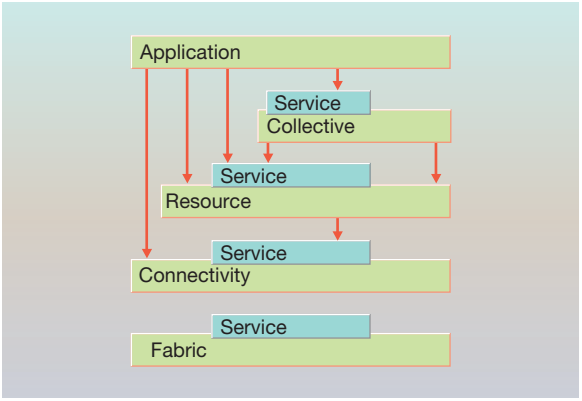
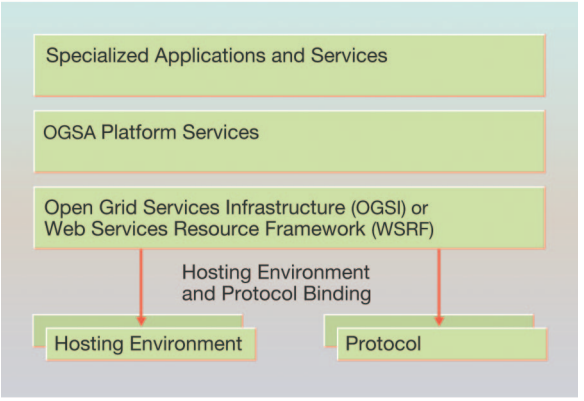


Figure 4 OGSA platform architecture



grid computing focuses on the easier integration, security, and QoS aspects of resource sharing.

Foster et al.[13] described the Open Grid Services Architecture (OGSA) as a solution to the above problem. This architectural concept is a result of the alignment of existing grid standards with emerging SOAs, as well as with the Web. OGSA provides a uniform way to describe grid services and define a common pattern of behavior for these services. It defines grid-service behavior, service description mechanisms, and protocol binding information by using Web services as the technology enabler. This architecture uses the best features from both the grid-computing community and the Web-services community.

OGSA architecture and goal. OGSA is a layered architecture, as shown in Figure 4, with clear separation of the functionalities at each layer. As seen in the figure, the core architecture layers are the Open Grid Services Infrastructure (OGSI) and OGSA platform services. The platform services establish a set of standard services including policy, logging, service level management, and other networking services. High-level applications and services use these lower-layer platform core components to become a part of a resource-sharing grid.

The Global Grid Forum[14] (GGF) has adopted the OGSA platform. There have been many activities in the GGF to define the use cases and core platform services, but there has been little activity related to the platform binding and resource modeling and profiling areas. A detailed discussion of OGSA, infrastructure, and platform services can be found in Reference 5.

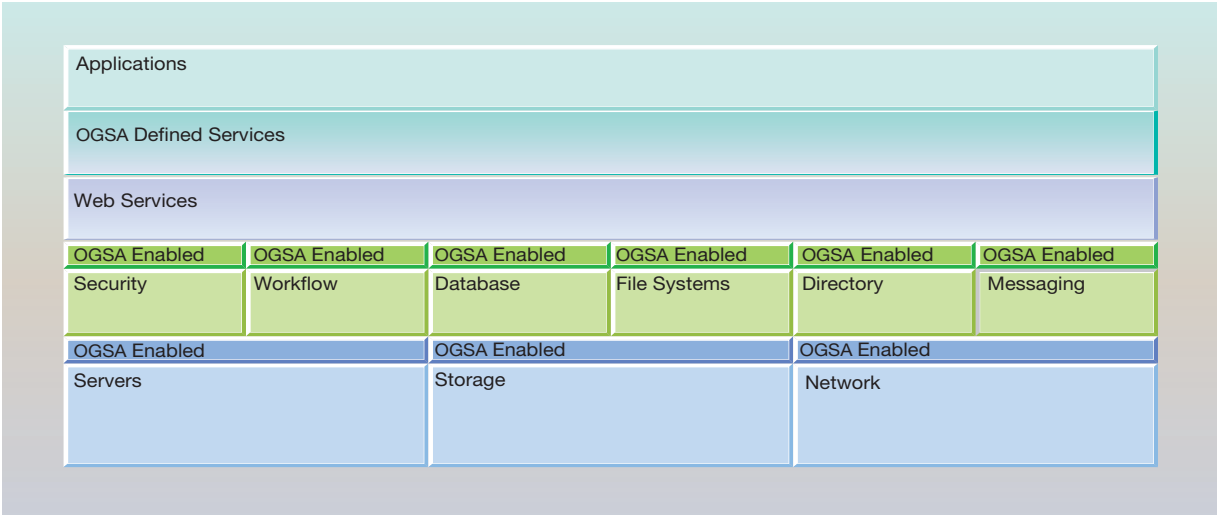
The IBM vision of the OGSA is summarized in Figure 5. This is a layered architecture, with the lowest layer comprising the basic IT resources, such as servers, storage, and network services. This includes the hardware and corresponding software support for operating systems, subsystems, and the components that control them.

The layer above the IT resources handles functions such as security, workflow, databases, file systems, directories, and messaging software. These are typically implemented as general-purpose middleware components. This middleware generally exploits the lower layer of physical resources and provides functions that can be abstracted and "virtualized" as services.

Grid services standardization

A grid service is (in practice) a Web service that conforms to a particular set of coding practices, namely, a set of XML coding conventions in the form of standards. For example, grid services can be defined in terms of standard XML-based WSDL, with perhaps some minor XML language extensions. These grid services can now take advantage of standard Web services binding technologies (for messaging, reliability, transactions, and security), such as SOAP, WS-ReliableMessaging (Web Services Reliable Messaging), WS-Transaction (Web Services Transaction) and WS-Security (Web Services Security). Grid ser-

Figure 5 OGSA architecture with Web services-enabled service interface



vices indeed look like Web services, especially from a programmatic view.

As stated earlier, all of the resources (physical or logical) in OGSA are modeled as services. These services are built on top of the SOA, and more specifically the WSA. This enables a grid service to use the capabilities of the message model, service descriptions, and discovery. Web services standards have evolved to enable these services to extend capabilities for secure and reliable transactions. In the next section, we explore the evolution of Web services standards and the fundamental principles behind these pluggable extended standards.

Evolution of Web services standards. Standards are critical to interoperability within a distributed computing platform, especially an advanced platform that provides a service-oriented, loosely coupled, cross-platform programming model. Standards enable platform services to more simply integrate with the middleware and infrastructure. This, in turn, also helps to reduce the complexity of heterogeneous and cross-enterprise orchestration and integration.

Figure 6 illustrates some of the evolving standards supported by industry leaders such as IBM and Microsoft.[15] These standards share the following elements:

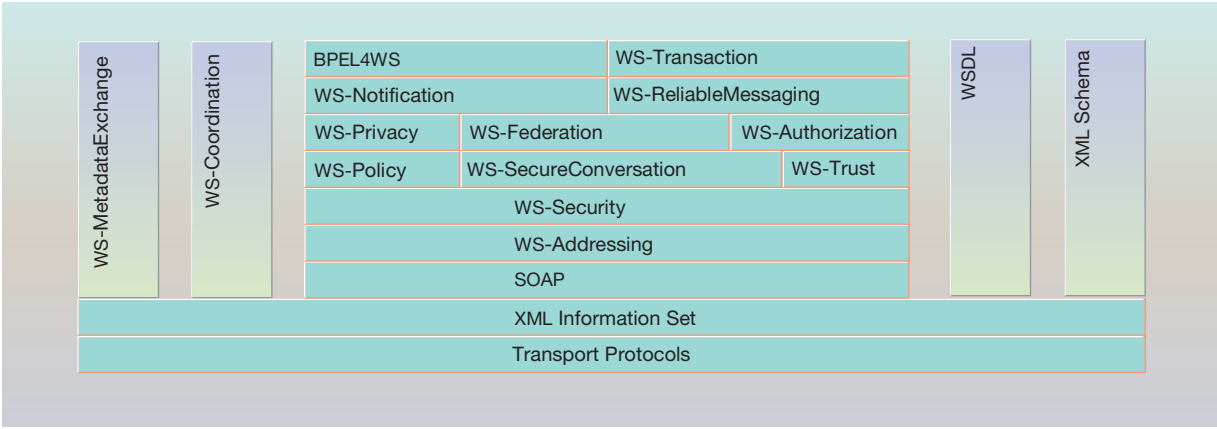
1. *The XML data model*— As defined by W3C** (the World Wide Web Consortium), the XML data model, or XML information set, forms the core of all XML specifications, including SOAP and WSDL. This common base allows more adaptable tools and XML processors to be created.

2. *Modularity*—SOAP provides an enveloping and processing mechanism for XML messages to be exchanged. SOAP provides a transport-independent data format for the transfer of XML messages based on the SOAP encoding format as specified by the SOAP specification or by using XML schema. Messages encoded by using the XML schema are recommended and are often referred to as *document/literal* messages.

3. *Decentralization and federation*—A constraint agreement is a regulatory agreement between a Web services client and a service provider on the format of message to be exchanged. Constraint agreements exist between the parties involved in any Web services integration. The parties agree on the syntax and semantics of the messages being exchanged although they may disagree on the software accepting the messages. They may also disagree on the programming languages used to build the system, the operating systems used, and the databases used. This is a benefit as it does not force issues of configuration conformance on all participants.

   This concept of decentralization allows the parties involved to make their own decisions on

Figure 6 Web services extensions



all aspects of message processing, including security, policy requirements, and processing. At the same time, the concept of federation allows these parties to exchange information in meaningful formats. For example, parties can exchange messages even though they disagree on internal security implementation mechanisms.

4. *Application and transport neutrality*—This is a binding-level decision between the service and the requestor. This binding agreement is independent of the message being exchanged. The application neutrality comes from the fact that the parties are not defining any specific message exchange protocol but instead are defining standards-based XML messages that need to be exchanged.

5. *Open standards*—Most of the specifications and standard protocol definitions mentioned here are being submitted to various standards authorities, including OASIS (Organization for the Advancement of Structured Information Standards) and W3C. Later in this paper, we discuss some of the standards bodies that are playing a major role in the grid standardization process.

As seen in Figure 6, the major building blocks identified by these plug-and-play extension standards include the facilities for message-level security, exchanging transaction-aware messages, message exchange coordination among participants, and reliable message exchange patterns. Other facilities include addressing mechanisms, service and message policies for proper message handling, meta-data information exchange, and a notification framework for consumers and producers exchanging messages.

In the following section, we focus our attention on grid services and explore the standards that play a major role in this area.

Grid services as Web services. There are two core standards that are available in the grid standardization area. These are OGSI and Web Services Resource Framework (WSRF).

The OGSI standard. The base component of the OGSA architecture is OGSI. This is a grid software infrastructure standard based on the emerging Web services standards. The goal of OGSI is to provide maximum interoperability among OGSA software components. Based on the OGSI specification,[17] a grid service instance is a Web service that conforms to a set of conventions expressed by WSDL as service interfaces, extensions, and behaviors. A grid service provides the controlled management of the distributed and often long-lived state that is commonly required in sophisticated distributed applications.

Figure 7 shows the layering of the OGSI components in a Web service with new interfaces and behaviors. The most notable point is the extension of WSDL to provide additional state data description mechanisms. In addition to this, the specification is a set of behaviors and interfaces to support service life-

Figure 7 Open Grid Services Infrastructure (OGSI) components

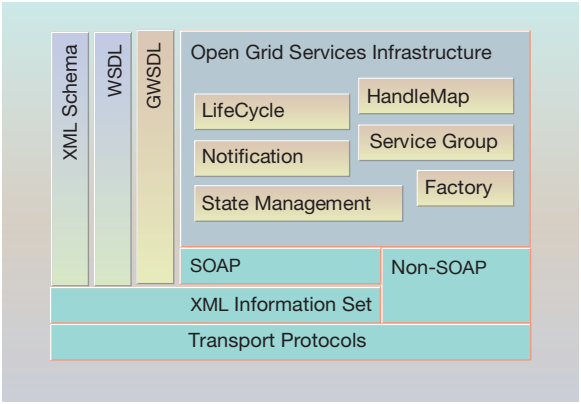
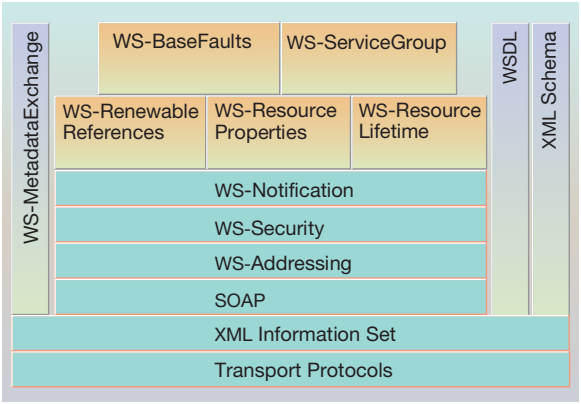


Figure 8 Web Service Resource Framework with WS specifications



cycle stages: for example, service-level management, collection management, state-change notifications, service creation mechanisms, and instance-reference management.

Message-level interoperability is a key feature of this standard, and it is achieved by using XML as the core message format and schema. One of the requirements of the services defined by OGSI is the ability to describe the concepts of state data, life-cycle properties, and instance behaviors using an OGSI description model, which is in fact a combination of WSDL and the OGSI GWSDL[5,16] (Grid Web Services Definition Language).

The WSRF standard. WSRF is a collection of specifications to support grid services or other stateful resources (resources whose behavior is defined with respect to their underlying state) and is comparable to OGSI. There are many motivations behind the WSRF specifications.[17] As seen in Figure 8, the most notable contribution is the intersection of grid computing and Web services standards and their alignment with SOA principles.

This alignment will continue to help define open standards through interoperable and compatible plug-and-play service extensions to the grid architectures, thereby increasing acceptance and facilitating integration. Through this alignment with the Web services stack, grid services can use existing Web services standards, such as WS-Notification,[18] WS-Addressing,[19] and WS-Security,[20] and build extensions for extended capabilities such as service state data, lifetime, grouping, and reference management.

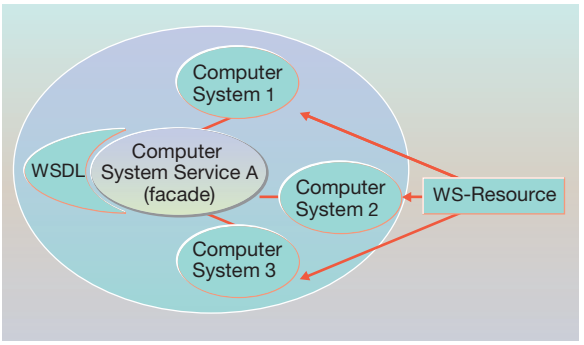
In the previous discussion, we described grid services as service representations of resources. These resources could be stateful or stateless (a resource whose behavior is not affected by stored behavior). Normally, grid services are assumed to represent some resources that have state. In the following sections, we will suggest how to manage the context in the case of a stateful resource. This discussion introduces aspects of the WS-Resource standard and the new modeling concepts relating to these resources.

Introduction to WSRF

There are many motivations behind the WSRF specifications. The most notable contribution of WSRF is to bring grid and Web services standards together. This requires the alignment of WSRF with SOA principles. This alignment helps to further define open-standards, interoperable, and plug-and-play service extensions to the grid architecture. It has been noted that the OGSI specification, as opposed to WSRF, tends to be more object-centric, as a complex set of concepts with an overutilization of XML schemas; some argue that its extensibility features are not well suited for existing Web services tools.[21]

Another motivating factor is the convergence of grid services to align with existing languages, programming models, tools, and technology directions in Web services, systems management, and on demand computing. Some of the best examples of this conver-

Figure 9 A relationship model showing the WS-Resource and associated Web service



gence are in Web Services Distributed Management (WSDM) and in IBM's on demand infrastructure virtualization areas.

The WSRF specifications are built on top of the implied resource pattern, as described in the following section. These extended sets of specifications now enable a stateful resource to manage its lifetime, send and receive notification on state changes, expose state data as XML documents, and manage faults.

Implied resource pattern for stateful resources. The implied resource pattern[22] addresses the relationship between Web services and stateful resources through a set of conventions expressed through composable Web services technologies such as the WS-Addressing standard. (By "composable," we refer to the Web services set of extensibility points, which enables composing multiple Web services specifications to enable more advanced features.) The requestor identifies a stateful resource through the WS-Addressing mechanism called the endpoint reference (EPR). The EPR contains enough information to dispatch the correct endpoint through the EPR properties. This is a pattern of message exchange with the implicit resource identifier in the context of message (using WS-Addressing reference properties) to uniquely identify the resource with which to communicate. A stateful resource that is taking part in this implied resource pattern is called the WS-Resource.

WS-Addressing. This capability provides transport-neutral mechanisms for locating Web services.[19] It is provided by the specification, utilizing the following constructs: a flexible and extendable EPR description model, a set of SOAP message headers (SOAP features), and rules to map these reference elements to the SOAP header elements.

Normally, Web services are invoked by the service endpoint information that is provided by the WSDL. For example, a WSDL service port has a location address, which identifies the endpoint. This information is suitable in most cases, with the exception of stateful Web services and cases that add more dynamic information to the address (instance information, policy, complex binding, and so on). This requires a client or runtime system to uniquely identify a service at runtime, based on this runtime information. This binding-specific information may include a primary key, unique identifier, and so on. Currently, there is no standard way by which this information can be exchanged and then mapped to the runtime engine while the service is accessed. The WS-Addressing specification addresses this problem by providing a lightweight mechanism for identifying and describing the endpoint information and mapping that information into the SOAP message headers.

The WS-Addressing specification standardizes the representation of the address of a Web service deployed at a given network endpoint, provides a WS-Addressing EPR that is an XML serialization of a network-wide pointer to a Web service, and provides an EPR that contains a service address (wsa:Address), meta-data associated with the Web service (such as service description information [WSDL]), policy information related to the usage of the service, and reference properties (wsa:ReferenceProperties), which can be used to identify a specific stateful resource instance associated with the Web service at the endpoint address.

WS-Resource. This is a stateful resource such as disk storage, a computer system, an operating system, or a shopping cart. Resources are identified as Web services resources by participating in the implied resource pattern (i.e., conforming to the defined pattern of interactions for stateful objects). The WS-Resource represents both the stateful resource and an associated Web service.

A WS-Resource has a network-wide EPR to identify the Web service and a set of reference properties to uniquely identify the specific resource through the WS-Addressing reference properties. For example, Figure 9 shows three computer system resources. It shows a service facade Web service with a network-wide endpoint. Each computer system is identified by using the reference properties in WS-Addressing.

The content of the identity information for the WS-Resource is binding-specific, and the service requestor should not interpret that content. This identity information, which is supplied along with the SOAP header, is used by the Web service facade to identify the correct resource. A WS-Resource may implement multiple interfaces and can associate multiple Web services facades with different EPRs.

The state data of the stateful resource can be exposed to the service requestor through resource property documents (XML document views). Each stateful resource has a life cycle with a well-defined semantics association for its creation and destruction. The identity of a resource is attached to it when it is created.

WS-Notification. This is the event-driven interaction pattern that is commonly used for interobject communications.[18] Examples exist in many domains, for example, in publish/subscribe systems provided by message-oriented middleware vendors, or in system and device management domains. This notification pattern is being more widely used in a Web services context.

The WS-Notification standard is a family of related white papers and specifications that define a standard Web services approach for message notification, using a topic-based publish/subscribe pattern. It includes standard message exchanges to be implemented by service providers that wish to participate in notifications, standard message exchanges for a notification-broker service provider (allowing publication of messages from entities that are not themselves service providers), operational requirements expected of service providers and requestors that participate in notifications, and an XML model that describes topics (i.e., items of interest for subscriptions.)

The WS-Notification family of documents includes a white paper (see Reference 18) as well as three normative specifications: WS-BaseNotification,[23] WS-BrokeredNotification,[24] and WS-Topics.[25] *WS-BaseNotification* defines the Web services interfaces for notification producers and notification consumers. This includes standard message exchanges to be implemented by service providers that wish to act in these roles, along with operational requirements expected of them. *WS-BrokeredNotification* defines the Web services interface for the notification broker. A notification broker is an intermediary, which, among other things, allows publication of messages from entities that are not themselves service providers. It includes standard message exchanges to be implemented by notification-broker service providers, along with operational requirements expected of service providers and requestors that participate in brokered notifications. *WS-Topics* defines a mechanism to organize and categorize topics. It defines three topic expression dialects that can be used as subscription expressions in "subscribe request" messages and other parts of the WS-Notification system. It further specifies an XML model for describing meta-data associated with topics.

The implied resource pattern is used to describe a stateful resource interaction. This scenario is shown in Figure 10 and involves the following steps:

1. Create a stateful resource with a specific identity[26] and a service interface.

   As shown in the specific scenario in Figure 10, we have identified three computer systems, which are stateful with codified state elements such as memory, CPU version, bus speed, and so on. These resources are stateful in nature and their identities are assigned by the runtime manager during their creation. In addition, there is a Web service for these resources with unique endpoint information (codified in WS-Addressing). The identity of each resource is unique in the domain of the associated Web service, and there is no need for it to be globally unique.

2. Service requestors interact with the stateful resources through implied resource patterns.

   A service requestor can interact with a stateful resource through the context established by WS-Addressing and its reference properties. The WS-Addressing EPR contains the network address (logical or physical) of the resource Web service, and the EPR reference properties (wsa: ReferenceProperties) contain the additional context information about the resource.

The above pattern of interaction with the resource is implicit. That is, the requestor does not provide the identity of the resource as an explicit parameter in the body of the request message; instead, the context is established through message headers, as indicated by the EPR reference properties. The resulting SOAP message is shown in Figure 11.

Figure 12[27] shows the convergence of grid and Web services standards. The technology gap and the pro-

Figure 10 Scenario showing the implied resource pattern associated with a WS-Resource

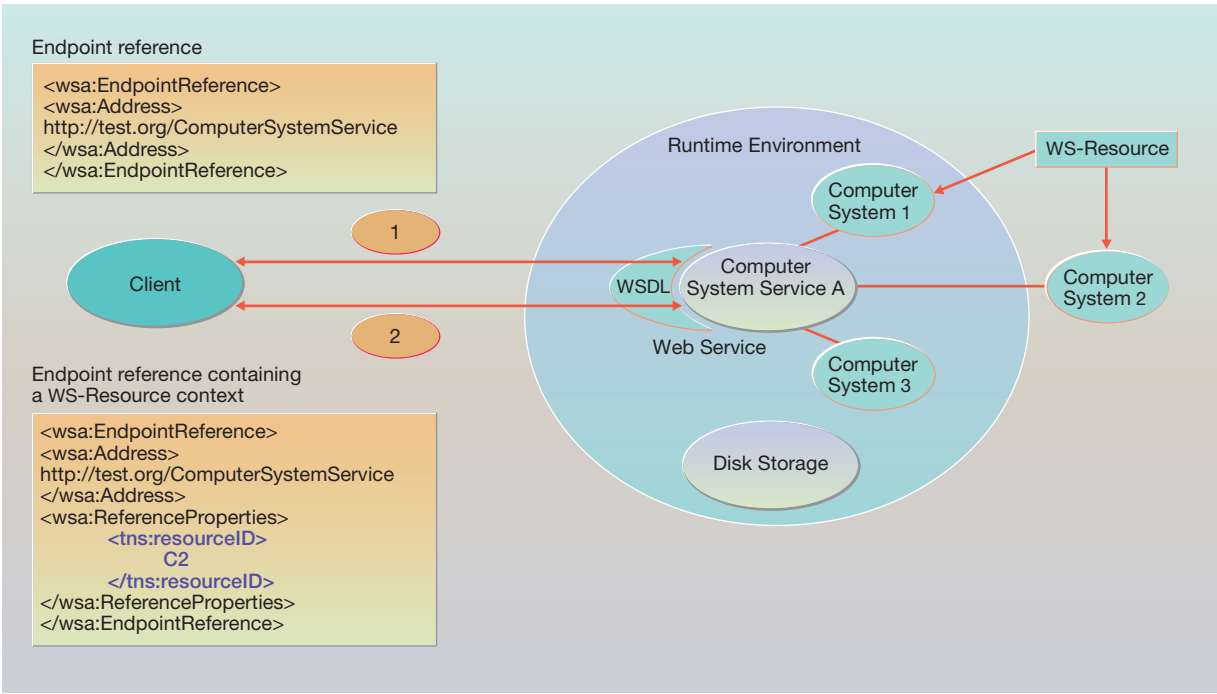


Figure 11 A SOAP message with a WS-Resource identity

```
<soap:Envelope>
        <soap:Header>
        <wsa:Action> …… </wsa:Action>  <!— The action to perform e.g., operation name to invoke - ->
        <wsa:To> http://test.org/ComputerSystemService </wsa:To> <!— Web service  end-point - ->
        <tns:resourceID> C </tns:resourceID> <!—resource identifier - ->
        </soap:Header>
  <soap:Body>
         … some message
  </soap:Body>
</soap:Envelope>
```

gramming model differences are narrowing through a standards convergence model that is moving towards a common framework. This has been noted in key areas such as Web services extension specifications, WSDL, and WSRF.

In the following, we describe the new set of standards that have been recently introduced to describe grid services, using the above implied resource patterns for stateful resources. The WSRF specifications that are of interest for our discussion are WS-Resource Properties, WS-ResourceLifetime, and WS-ServiceGroup. We examine these new standards in detail and discuss their relationships to the concept of grid services.

WS-ResourceProperties. Every grid resource has some form and state associated with it. This form

and state specification constitutes a message schema that defines and exposes these properties of a resource, as part of the Web services interfaces, to other resources. These properties may be a part of the resource's state (both logical and physical). Metadata about the resource and stateful information may be presented upon request.

This specification does not dictate exactly how the values of these properties are constructed. These values may in fact be in the resource, or they may be from external sources. They are exposed to the requestor as XML documents. In addition, this specification defines a set of standard message exchange patterns that allow the requestor to query, insert, and update the properties of a resource. As shown in Figure 13, the computer system resource exposes its state information through a well-defined set of interfaces, as defined by the exposed port types. A client could use these exposed operations to find, access, and modify the resource properties.

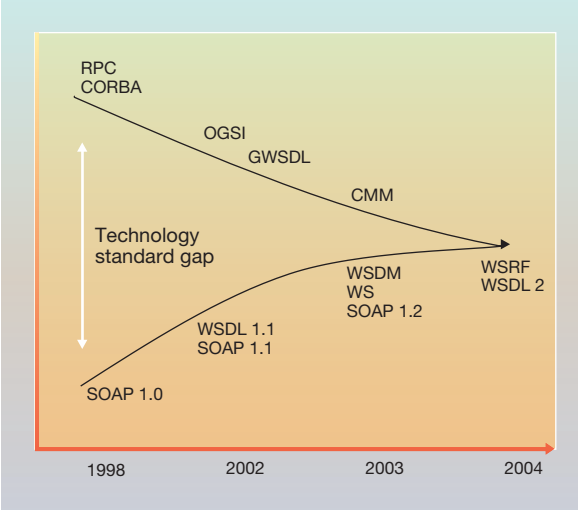
As shown in the XML code fragment of Figure 14, the "ComputerSystem" resource exposes a property called "current_memory." This property is defined as a part of the global XML schema element (e.g., "GetComputerSystemMemory") and referenced as part of the WSDL:portType open attribute content.

A service requestor can retrieve this property from the resource at runtime as an XML document view by using the "GetResourcePropertyRequest" or "GetMultipleResourcePropertiesRequest" operations associated with the Web service. The GetResourcePropertyRequest message is shown in the XML code fragment of Figure 15. This request is interpreted by the Web service, which in turn retrieves the necessary resource property value from the underlying resource. A service requestor can insert, delete, or update a resource property with a new value. The response is returned to the requestor, as shown in the XML code fragment of Figure 16.

Finally, this specification provides facilities for querying, by sending query expressions to retrieve fragments of resource properties. The format of this type of query expression is shown in the XML code fragment of Figure 17. The dialect attribute defines the query expression language of choice, such as XPath, XQuery, or SQL, and the query expression defines the query to be applied to the resource properties.

It is possible for a service requestor to request notification of changes (e.g., updates, deletions, and insertions) made to values of one or more resource property elements of a given WS-Resource. The requestor is able to take part in this notification process through the Web service associated with the WS-Resource.

Figure 12 Standards convergence graph



Reprinted with permission from Ian Foster

WS-ResourceLifetime. A grid service may be transient, created with a specified lifetime. This stateful resource's lifetime could be negotiated and controlled by its clients. This helps to provide a controlled clean-up mechanism for resources. The specification defines a set of standard message exchange patterns for time-based immediate destruction of a service. This service lifetime management offers explicit destruction capabilities to a service requestor or a specific service. In addition, it provides the capability for scheduled destruction at a future time. The semantics of destruction of a service are controlled by the security and service container policies. The interaction model ensures that once the resource is destroyed, the resource EPR is no longer valid and the requestor will not be able to connect to the resource using the same EPR.

This specification defines a set of service properties and two types of service destruction message patterns. The properties of a service that are used to manage its lifetime are InitialTerminationTime, CurrentTime, and TerminationTime. The identified service destruction message exchange patterns for lifetime management capabilities are:

Figure 13 A computer system WS-Resource with WS-ResourceProperties interfaces

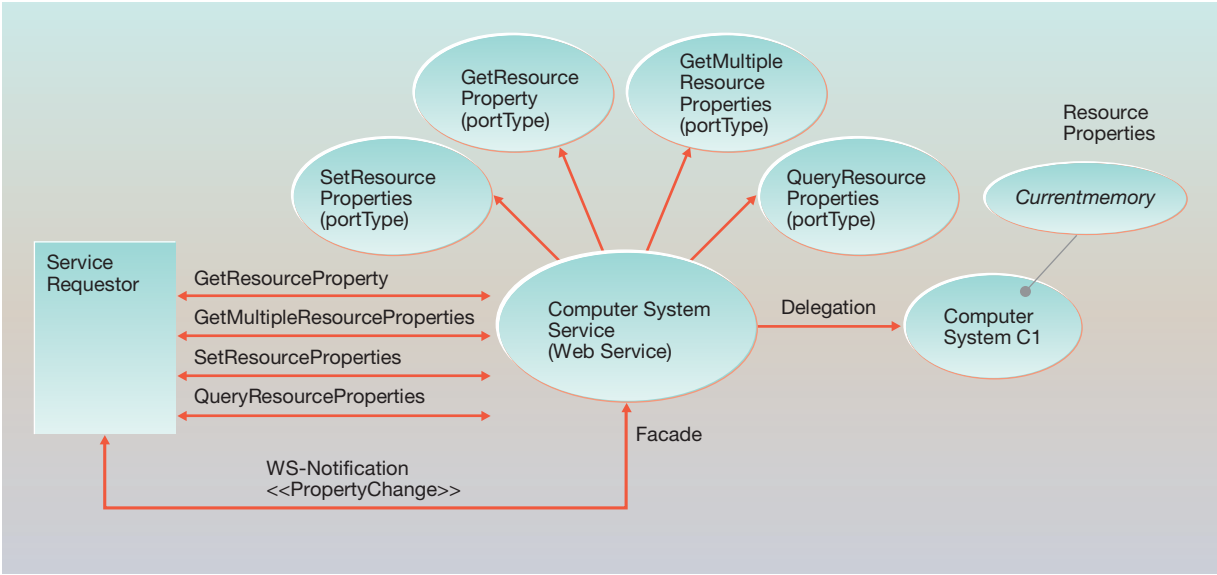


Figure 14 A WSDL message with a portType attribute

```
<xsd:element  name="GetComputerSystemMemory"   xmlns:tns="http://test.org/computersystem" >
            <xsd:element ref="tns:current_memory" />
</xsd:element>
<xsd:element name="current_memory"  value="xsd:integer" />

<wsdl:portType name="ComputerSystem"  wsrp:ResourceProperties="tns:GetComputerSystemMemory">
<operation name="start" .../>
</wsdl:portType>
```

Figure 15 A WS-ResourceProperties message with a GetResourcePropertyRequest operation

```
<wsrp:GetResourcePropertyRequest xmlns:tns="http://test.org/computersystem"  >
            tns:current_memory <!—name of the property to retrieve - ->
</wsrp:GetResourcePropertyRequest>
```

1. *Immediate destruction*—A resource that is taking part in the implied resource pattern could send a message (DestroyRequest) to the service, in order to immediately destroy the resource. This message could be of the format shown in Figure 18. The Web service that accepts this message may destroy the implied resource, or it could return an exception.
2. *Scheduled destruction*—The resource that is taking part in the implied resource pattern (by im-

Figure 16 A WS-ResourceProperties message return with a GetResourcePropertyRequest operation

```
<wsrp:GetResourcePropertyResponse xmlns:tns="http://test.org/computersystem"  >
        <tns:current_memory>  <!—an XML view of the resource property- ->
          512
        </tns:current_memory>
</wsrp:GetResourcePropertyResponse>
```

Figure 17 A WS-ResourceProperties message to query, send query expressions, and retrieve fragments of resource properties

```
<wsrp:QueryResourcePropertiesRequest>
            <wsrp:QueryExpression dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
                    //current_memory
            </wsrp:QueryExpression>
</wsrp:QueryResourcePropertiesRequest>
```

Figure 18 A SOAP message involved in a lifetime *destroy* management function

```
<soap:Envelope>
 <soap:Header>
            <wsa:Action>  http://test.org/ComputerSystemService/Destroy</wsa:Action>
            <tns:resourceID>  <!—the identity of the resource to destroy- ->
               C1
            </tns:resourceID>
 </soap:Header>
 <soap:Body>
            <wsrl:DestroyRequest/>
 </soap:Body>
</soap:Envelope>
```

plementing the scheduledTermination interface) could accept a message (SetTerminationTimeRequest) to destroy the resource after a specified period of time as indicated in the message.

The time synchronization in a distributed application is complex. The WS-ResourceLifetime specification provides a capability through which the service requestor could get the current time of a service (using the CurrentTime resource property).

In addition to the above capabilities, the Web service associated with the WS-Resource could be a notification producer (as defined in the WS-Notification specification). Notification producers provide notification topics to allow service requestors to subscribe for notification about the destruction of a resource.

WS-ServiceGroup. In some cases, it may be required to aggregate a set of Web services. This may be in order to provide a domain-specific solution, or as a simple collection of services for indexing and other discovery enablement scenarios. The WS-ServiceGroup specification describes a "by-reference" collection of Web services, where these Web services

Table 1 Comparison of OGSI and WSRF concepts.

OGSI	WSRF
Grid service reference	WS-Addressing endpoint reference
Grid service handle	WS-Addressing endpoint reference
HandleResolver portType	WS-RenewableReferences
Service data definition and access	WS-ResourceProperties
GridService lifetime management	WS-ResourceLifeCycle
Notification portType	WS-Notification
Factory portType	Factory pattern concept
ServiceGroup portTypes	WS-ServiceGroup
Base fault type	WS-BaseFaults

Reprinted with permission of Ian Foster

may constitute a WS-Resource. It also provides key manageability interfaces to better manage entries in the group (e.g., add, delete, and modify).

Although any Web service can become a part of this collection, the service group manages its individual entries as WS-Resources. It accomplishes this through the use of other standards for Web services, and WS-Resource standards such as those concerning notification, lifetime, properties, and addressing.

In the next section, we compare the existing OGSI specification with the concepts introduced by WSRF.

Comparison of WSRF with OGSI. Table 1[27] shows a one-to-one mapping of OGSI-related concepts to WSRF solutions. The most noticeable aspects of the comparisons shown are:

- The implied resource pattern introduces the concept of a stateful resource (WS-Resource), and this pattern replaces the mandatory interface GridService, as defined by the OGSI specification. This enables all stateful resources, including grid services, to be handled through a common set of Web services tools.
- The Grid Service Handle (GSH) and Grid Service References (GSR) concepts introduced by the OGSI specification are replaced by the Web services standard WS-Addressing. GSH and GSR are encoded in the WS-Addressing specification, using EPR and reference properties. This is the most notable contribution of the WSRF.
- The WSRF introduces a standard notification framework for Web services. This enables all Web services and grid services to share notification patterns (i.e., standard and brokered notifications) as introduced by the specification. This in turn enables the messaging providers to define a common set of interfaces and facilities.

From our observations thus far, the remaining concepts that have been introduced are more specific to the grid services concept of stateful resources and their management than to Web services. We predict that these composable specifications will help service providers to make a selection of the most suitable choices for their services.

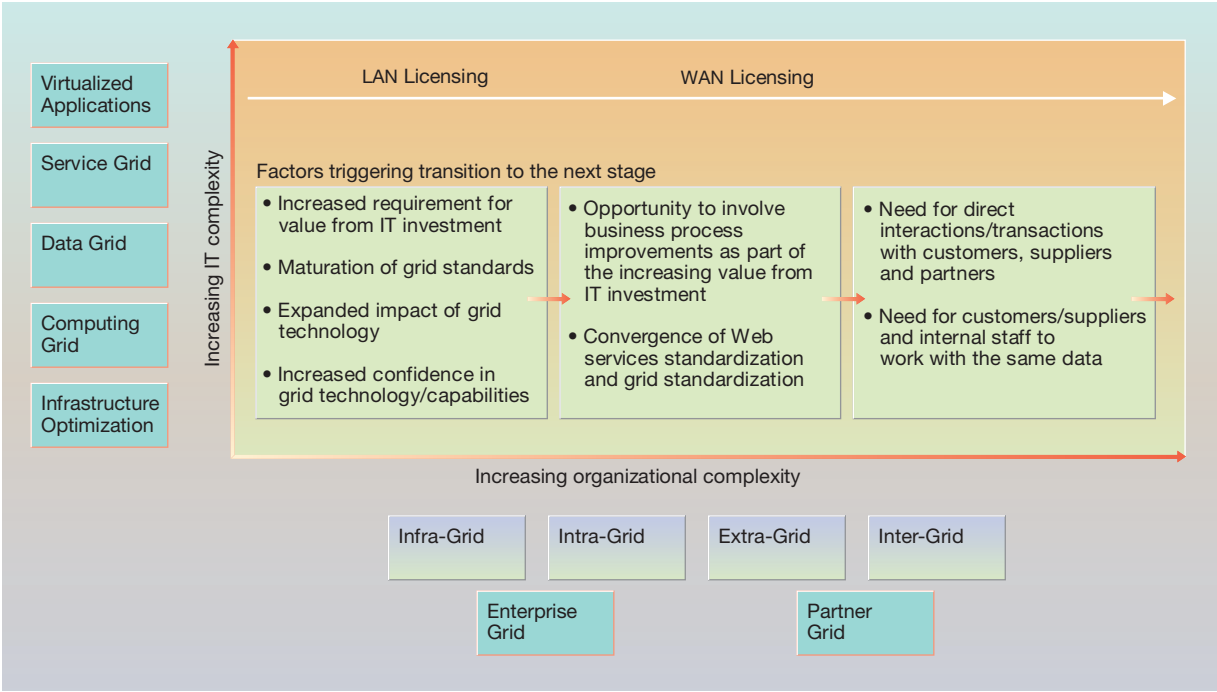
Grid adoption models

Grid computing falls into the domain of several different research communities, depending on how the question, "What is a grid?" is answered. These grid communities include the "compute-centric," peer-to-peer, utility, data and applications, and collaboration areas. Each of these communities may use a different model for the adoption of grid technology (i.e., a different "grid adoption model").

The concept of grid computing is relatively new to commercial industry. A major focus area surrounds infrastructure virtualization while dealing with resources as utilities. Grid computing is one of the strategic technologies for IBM.

Grid adoption (in the commercial industries) depends on the ability of this technology to deliver increased business value. In this section, we focus our attention on the models for grid adoption and describe their basic attributes and the standards that facilitate the grid adoption process. The business issues related to the grid adoption model include key factors, such as leveraging existing hardware investments and resources, reducing operational expenses, creating a scalable and flexible infrastructure, accelerating development time, improving time to market, and increasing customer satisfaction and business productivity. Figure 19 shows the major factors that are influencing grid adoption.

Figure 19 Grid adoption factors



An organization may do business with other organizations within the enterprise or with external organizations. As a result, grids may involve only internal partners or both internal and external partners. The complexity of the business requirements of such an integration depends on the virtualization requirements, business impact, trust relationships, security considerations, globalization, and time-to-market requirements of the enterprises involved. Based on differing levels of complexity, grids for the enterprise can be categorized as one of the following types:

- *Infra-grid*—This grid architecture enables optimizing resource sharing within departments in one division of an organization. This is a very controlled environment with well-defined business policies, integration, and security requirements. Because the management issues are contained within a single management domain, the focus can be kept primarily on gaining technical experience. These types of grids are sometimes called "cluster grids."
- *Intra-grid*—This more complex grid implementation is a scenario for resource integration by using the computing and data/storage resources of various divisions within an organization. These types of grids need well-defined policies for the sharing of resources within the enterprise, and valuable experience can be gained in dealing with the more complex security, data-sharing and resource-sharing policies required. However, because the resources are within the same enterprise, the primary focus can still be on the technical implementation of the policies defined. These types of grids are sometimes called "enterprise grids" or "campus grids."
- *Extra-grid*—These grids deal with sharing of resources, including those belonging to a trusted external partner with whom a business relationship has already been established. These types of grids are also known as "partner grids." Because these grids extend outside the management domain of a single enterprise, mutual or shared partner agreements and service-level objectives on resource utilization are required. Establishing these agreements provides valuable experience in writing, maintaining, and managing access according to grid service-level agreements.
- *Inter-grid*—An inter-grid enables the sharing of computing and data/storage resources across a

public Web in collaboration with other enterprises, potentially selling excess capacity (for example). This is more of an on-demand utility conception of a grid, with inherent complexities in managing service-level requirements, security federation, and integration. Its requirements will, in part, build upon the experiences and critical skills gained from the previous grid implementation types.

The above discussion and Figure 19 accentuate the complexity of considerations potentially encountered within any line of business and the requirements of grids involving complex business patterns such as partner integration. More is known about the left side of the continuum of the abscissa of Figure 19, as opposed to the far right side of the continuum, simply due to industry experience in grid computing to date.

A major factor that determines the adoption rate of grid models is the complexity of the IT infrastructure required to implement a grid. In this section, we address the topic of integration of grid resources, including grid adoption in various technology domains. The complexity of IT integration across heterogeneous environments is a real challenge. There are various factors we need to take into consideration, including enabling grid resources in homogeneous and heterogeneous environments, enabling resources as services to grid partners, and enabling virtualized applications to grid partners. These requirements at the IT level enable us to classify grids into various categories, with varying degrees of complexity at the integration level, such as:

1. Grids designed to optimize infrastructure
2. "Computing grids" with virtualized processing
3. "Data grids" with virtualization of data and storage
4. "Service grids" with virtualized services to enable easier integration
5. Virtualized applications enabled by composing resources from various partner applications through service interfaces

Figure 19 shows the levels of IT and organizational complexity of various grids and the specific factors that prompt moving from one stage to the next in the grid adoption process.

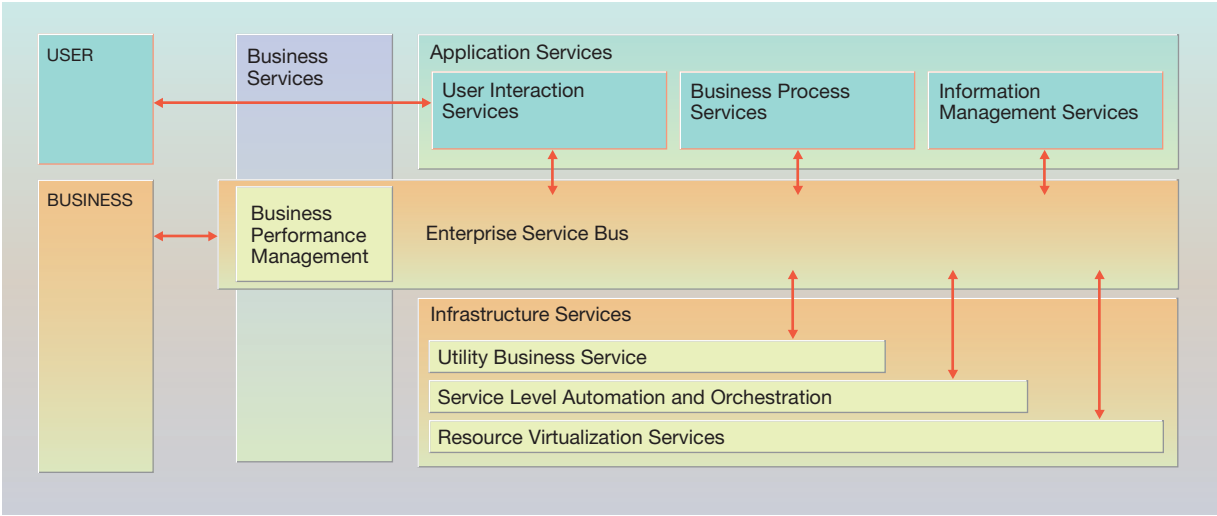
The combination of business and technical aspects mentioned previously gives rise to the grid adoption model framework. By considering both the technical and business requirements within each cell of the framework, clients considering grid implementations can better anticipate and prepare for their challenges. The grid adoption framework reveals two significant transition points. At the outset, projects primarily focus on IT and increasing sophistication of IT management capabilities. However, the transition from internal implementations to those involving external partners represents a significant advance and necessitates the involvement of not only the IT organization, but also the rest of the business.

The grid architecture and global standards have a major role in governing the adoption of the grid in the commercial world. Because these standards are still evolving and are not sufficiently mature to support the latter stages suggested by the adoption framework, the framework itself will also have to evolve. These same standards will also restrict the speed with which these latter stages can be attained. However, grid computing and on demand business environments have been implemented in several vertical industries (e.g., finance, education, life science, telecommunications), and these implementations seem to validate the stages suggested. The primary benefit of the framework is to clearly articulate what capabilities, both in business and IT, are required before an organization can successfully attain its virtualization goal.

Clearly, the success of grid computing depends on integration and services orientation. The ability of applications to decompose their capabilities and then expose themselves as services is also important for strengthening the SOA. In addition, a key to success in grid adoption is creating virtualized applications to solve specific VO problems by choreographing the business functions exposed through services. This is where global standards and architecture frameworks will help.

The standards for Web services and WSRF, aligned with OGSA, assist in this integration. The availability of technology implementations to facilitate such standardization is important. This includes middleware application platforms, resource virtualization engines, workflow models, messaging and correlation systems, and tool frameworks. Figure 20 shows a high-level framework view of an IBM on demand operating environment. The figure shows the required infrastructure services such as systems, applications, and business processes at the lower layer, and application services at the higher level.

Figure 20 A high-level view of the on demand operating environment



Grid standards in the on demand operating environment

An on demand business is an enterprise whose business processes—integrated end-to-end across the company and with key partners, suppliers, and customers—can respond with flexibility and speed to any customer demand, market opportunity, or threat. Our strategic intent is to assist clients in achieving virtualization of on demand services and to increase IBM's market share in this area.

When an enterprise begins to virtualize on demand business services, the complete involvement of the affected business organizations is always required. As a result, systems development is no longer under the complete control of the IT organization, and collaboration of various departments within an organization is critical. Similarly, before the virtualization of specific on demand business services is started, the projects involving business services are largely under the control of the IT organization.

The on demand operating environment is based on the concept of an SOA. The on demand business models are built upon open standards, intended to define business, applications, and systems at various levels within a department, across an enterprise, or spanning an entire industry.

Each element of the SOA is a service, and together, these services implement the operating environment capabilities. These services communicate over a shared enterprise service bus (as shown in Figure 20), which provides messaging, mediation, transformation, integration, and correlation capabilities. This integration fabric of components is shared across departments, enterprises, and partners. The system-level components are integrated through virtualized resources and services.

SOA is an area where grid computing is beginning to play a major role. The grid adoption rate may vary depending on the environment's integration requirements, for example, whether it operates within an enterprise or includes external partners. Also significant is the virtualization complexity, which is proportionate to, and dependent on, the grid application's complexities.

Focus areas for grid standards

This section highlights some of the aspects of grid computing that still require some form of standardization. This discussion also describes some of the existing standards that could be precursors to this standardization process. Table 2 identifies and describes the focus of some of the important organizations that are playing a major role in the Web services standardization area, including those related to the grid and systems management domains. These standardization efforts, as listed in Table 3, should be aligned with aspects of

Table 2 The major grid standards organizations.

Organization	Standards Activities	Web Site
Global Grid Forum (GGF)	Grid computing, distributed computing, and peer-to-peer networking	http://www.ggf.org
World Wide Web Consortium (W3C)	World Wide Web, XML, Web services, Semantic Web, XML, mobile, and voice	http://www.w3c.org
Organization for the Advancement of Structured Information Standards (OASIS)	Electronic commerce, systems management and Web services extensions (WS), BPEL4WS, and portals	http://www.oasis-open.org/
Web Services Interoperability Organization (WS-I)	Interoperable solutions, profiles, best practices, and verification tools	http://www.ws-i.org
Distributed Management Task Force (DMTF)	Systems management	http://www.dmtf.org
Internet Engineering Task Force (IETF)	Network standards	http://www.ietf.org
European Computer Manufacturers Organization (ECMA), International Organization for Standardization (ISO)	Language standards (C++, C#)	http://www.iso.org http://www.ecma-international.org
Object Management Group (OMG)	Model-Driven Architecture (MDA), Unified Modeling Language (UML**), Common Object Resource Broker Architecture (CORBA**), real-time system modeling	http://www.omg.org
Java Community Process (JCP)	Java standards	http://www.jcp.org

SOA and with the Web services standards, as suggested earlier.

Conclusions

There is a natural convergence of grid services and Web services. This convergence is occurring right now, and it is happening in all industries. It can be observed in the evolutionary thinking of those people who are members of VOs and are participating in this transformation.

The grid architecture and global standards serve a major role in determining the adoption rate of grids in the commercial world. These standards are still evolving.

Grid-service conventions are nontrivial in their functions; they solve (in a new way) some of the fundamental issues in distributed computing. These issues relate to the naming, creation, discovery, monitoring, and management of the lifetime of stateful services. More specifically, these conventions support very important distributed computing areas, including named service instances, a two-level naming schema that facilitates traditional distributed system transparencies, a base set of service capabilities, including rich discovery facilities, and explicitly stateful services with lifetime management capabilities.

Will the intersections of services and standards that we have discussed in this paper continue to be accomplished in the way that is currently prevalent? Our approach enables a transformation by applying the full power of traditional distributed systems to grids, including naming and binding techniques, across the widest possible set of Web services. The grid adoption models provide an innovative means for accomplishing this transformation, while shortening the time required to deliver grid computing and other capabilities of grid services.

Clearly, the emergence of grid services is an important milestone in the development of global Web services. Grid services are important because they provide uniformity and consistency for many vital distributed system functions. The ability to apply the

Table 3 Grid standards and requirements.

Area	Requirements	Existing Standards
Policy and service-level agreements	· Policy-based negotiation to establish grid service integration · Service-level agreement (SLA) management across heterogeneous grid providers · Advance reservation mechanisms based on customer requirements for QoS	OGSA Policy, WS-Policy, and WS-Agreements Advance Reservation Application Programming Interfaces
Job management	· Identification of individual schedulable entities · Definition of a common information model for job execution requirements, workflow characteristics · Definition of scalable and extendable mechanisms for job descriptions	Resource Specification Language (RSL), Job Submission Description Language (JSDL), Job Submission Information Model (JSIM), and Business Process Execution Language (BPEL4WS)
Grid scheduling	· Scheduling and executing individual jobs, workflow management · Meta-scheduler interfaces with capabilities for resource discovery, provisioning, resource scheduling, and job execution · Creating a unified scheduler that acts as a provider for all the grid resources in a virtual organization and as a meta-scheduler for execution of jobs	Grid local scheduling, Meta-scheduling, and Unified scheduler
Grid service security	· Sandbox (protected domain that places tight controls around the execution of downloaded code) security model for grid service execution · Federation of security across heterogeneous resource providers · Trust model and certificate management in a federated environment	WS-Security, WS-Trust, WS-Federation, Grid Security Infrastructure (GSI), and Generic Security Service extensions
Grid management model	· A common management model to describe and interact with heterogeneous resources · A standards-based monitoring system with systems management capabilities. · Performance management to meet SLA requirements for resources · Alignment with systems management concepts	Common Management Model (CMM), Grid Monitoring Architecture (GMA), and Web Services Distributed Management (WSDM)
Grid data management	· Access to and integration of structured and semi-structured data across grid · Discovery and storage of multitudes of data · Enhanced data mining for global information exchange · Virtual file system directory service and grid file system · Replica management	DAIS (Grid Data Access and Integration), XQuery, Virtual File System, Directory Service (VFSD), Grid File System (GFS), Local replica catalog services, and GridFTP
Grid and network	· Handling varying load on the network infrastructure due to heterogeneity of resources and policies applied to resources. · Handling the varying type data stream and its impact on the network · Managing dynamic network conditions · Managing the SLAs among customers and service providers · Aligning with emerging network standards such as IP-V6 · Utilizing mobile network standards	Grid High-Performance Network (GHPN) and Open Service Gateway Initiative (OSGi)
Grid architecture and programming models	· Defining an open architecture model for the grid · Aligning the architecture with the existing/emerging programming models and standards · Defining use cases, interoperable solutions and best practices for grid adoption	Open Grid Services Architecture (OGSA), Open Grid Services Infrastructure (OGSI), Web Service Resource Framework (WSRF), Java** Network Initiative (JNI), Semantic Grid, and New Productivity Initiative (NPI)

grid adoption models described in this paper serves a vital need in efforts for on demand business lifecycle planning.

Acknowledgments

The authors wish to thank George Galambos and John Helmbock for their work in the IBM grid adoption model. We also thank Ian Foster of the Argonne National Lab and Steven Tuecke of the Globus** Alliance for their work in the grid and WSRF areas. Finally, the authors would like to thank the reviewers for their thoughtful comments relating to an early draft of this paper.

Cited references and notes

1. J. Waldo, G. Wyant, A. Wollrath, and S. Kendall, *A Note on Distributed Computing,* Sun Microsystems Technical Report SMLI TR-94-29 (November 1994), http://research.sun.com/techrep/1994/smli_tr-94-29.pdf.
2. *IBM Autonomic Computing*, http://www-3.ibm.com/autonomic/index.shtml.
3. *IBM On Demand Business*, http://www-3.ibm.com/e-business/index.html.
4. *IBM Grid Computing*, http://www-1.ibm.com/grid/.
5. J. Joseph and C. Fellenstein, *Grid Computing*, IBM Press, Pearson Education Publishing, Pearson Technology Group (2003).
6. I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, San Francisco, CA (1998).
7. I. Foster, C. Kesselman, and S. Tuecke, *The Anatomy of the Grid—Enabling Scalable Virtual Organizations*, The Globus Alliance, http://www.globus.org/research/papers/anatomy.pdf.
8. Utility computing can be defined as the network delivery (by service providers) of IT and business process services.
9. A resource pool is a collection of dynamically created physical and logical resources such as servers, processors, and networks.
10. *XML Information Set*, WorldWide Web Consortium (February 2004), http://www.w3.org/TR/xml-infoset/.
11. *XML Schema*, WorldWide Web Consortium, http://www.w3.org/XML/Schema.
12. *XML Protocol Working Group*, WorldWide Web Consortium (2004), http://www.w3.org/2000/xp/Group/.
13. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, *The Physiology of the Grid—An Open Grid Services Architecture for Distributed Systems Integration*, The Globus Alliance (June 2002), http://www.globus.org/research/papers/ogsa.pdf.
14. Global Grid Forum, http://www.ggf.org.
15. D. Box, *Global XML Architecture* (2002), http://whitepapers.zdnet.co.uk/0,39025945,60063919p-39000542q,00.htm.
16. Final OGSI Specification V1.0 (July 2003), https://forge.gridforum.org/docman2/ViewProperties.php?group_id=43&category_id=392&document_content_id=347.
17. K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke, and W. Vambenepe, *The WS-Resource Framework* (March 2004), http://www-106.ibm.com/developerworks/library/ws-resource/ws-wsrf.pdf.
18. *Publish-Subscribe Notification for Web Services* (March 2004), http://www-106.ibm.com/developerworks/library/ws-pub-sub/.
19. A. Bosworth, D. Box, E. Christensen, F. Curbera, D. Ferguson, J. Frey, C. Kaler, D. Langworthy, F. Leymann, S. Lucco, S. Millet, N. Mukhi, M. Nottingham, D. Orchard, J. Shewchuk, T. Storey, and S. Weerawarana, *Web Services Addressing (WS-Addressing)* (March 2003), http://msdn.microsoft.com/ws/2003/03/ws-addressing/.
20. B. Atkinson, G. Della-Libera, S. Hada, M. Hondo, P. Hallam-Baker, J. Klein, B. LaMacchia, P. Leach, J. Manferdelli, H. Maruyama, A. Nadalin, N. Nagaratnam, H. Prafullchandra, J. Shewchuk, D. Simon, and C. Kaler, *Web Services Security (WS-Security)* (April 2002), http://www-106.ibm.com/developerworks/library/ws-secure/.
21. K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke, *From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution* (March 2004), http://www-106.ibm.com/developerworks/library/ws-resource/ogsi_to_wsrf_1.0.pdf.
22. I. Foster, J. Frey, S. Graham, S. Tuecke, K. Czajkowski, D. Ferguson, F. Leymann, M. Nally, I. Sedukhin, D. Snelling, T. Storey, W. Vambenepe, and S. Weerawarana, *Modeling Stateful Resources with Web Services* (March 2004), http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf.
23. S. Graham, P. Niblett, D. Chappell, A. Lewis, N. Nagaratnam, J. Parikh, S. Patil, S. Samdarshi, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, and B. Weihl, *Web Service Base Notification (WS-BaseNotification)* (March 2004), ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf.
24. S. Graham, P. Niblett, D. Chappell, A. Lewis, N. Nagaratnam, J. Parikh, S. Patil, S. Samdarshi, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, and B. Weihl, *Web Service Brokered Notification (WS-BrokeredNotification)* (March 2004), ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BrokeredN.pdf.
25. S. Graham, P. Niblett, D. Chappell, A. Lewis, N. Nagaratnam, J. Parikh, S. Patil, S. Samdarshi, I. Sedukhin, D. Snelling, S. Tuecke, W. Vambenepe, and B. Weihl, *Web Service Topics (WS-Topics)* (March 2004), ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-Topics.pdf.
26. The endpoint creates the reference properties in the EPR and uses these properties to express enough information to dispatch the appropriate state of the resource.
27. I. Foster, *WS-Resource Framework—Globus Alliance Perspectives*, http://www.globus.org/wsrf/foster_wsrf.pdf.

Accepted for publication June 4, 2004.

Joshy Joseph *IBM Software Group, A0-3A, Building K, Poughkeepsie, New York 12601 (joshy@us.ibm.com).* Mr. Joseph is currently working with IBM Software Group's On Demand Architecture and Development team, which is responsible for the development of IBM's on demand incubator projects. His main focus is on business integration and business performance management and grid computing. In his current position, he is working as the development lead for the Enterprise Component Business Architecture (ECBA) pilot projects. Prior to joining this group, he was working on the IBM Systems Group grid computing architecture initiatives. Mr. Joseph is actively involved with

IBM's contribution to the OGSI standard and the Globus Grid software programming model. Previously, he contributed to IBM's WebSphere® Catalog Manager and Commerce Integrator teams. He is an architect and programmer with primary skills and experience in the areas of distributed computing, grid computing, workflow models, and advanced Web services. Mr. Joseph is the coauthor of a book on grid computing. He has received a number of performance and publication awards. He has approximately 20 patents pending in the U.S. Patent Office, over 15 inventions with "file" status within IBM, and a number of invention publications. He has also published a number of articles for *IBM developerWorks* in the areas of grid computing, business processes, and Web services. Prior to joining IBM, Mr. Joseph worked for Bell Atlantic Science and Technology and People's Bank and was involved in numerous projects in the areas of mobile computing, distributed computing, performance management, and Internet computing. He is an expert in the .NET and J2EE™ programming models.

Mark Ernest *IBM Global Services/Integrated Technology Services 800 N. Frederick Ave., Gaithersburg, Maryland 20879 (lernest@us.ibm.com).* Mr. Ernest's primary focus over the past four years has been the leadership of the team responsible for the development, evolution, and application of IT optimization methodology. In that role, he helped develop assessment and adoption methods for both grid and autonomic computing. Currently, he holds a position within the Global ITS Organization as the Chief Technology Officer for operational efficiency. In addition, he is a member of the ITS Technology Council and serves on the core teams of the IT optimization, enterprise systems management, IT cost and value, and e-business infrastructure communities of practice. Collectively, these groups are responsible for creating many of IBM's IT consulting-related work products, tools, and techniques. Mr. Ernest joined IBM at the Washington Systems Center 26 years ago and was asked to become a founding member of IBM's IT consulting competency in 1992. He was named a Distinguished Engineer by the Corporate Technology Council in April 2001.

Craig Fellenstein *IBM Global Services/Strategic Outsourcing, 6 Hunting Ridge Road, Brookfield CT 06804-3710 (cfellen@us.ibm.com).* Mr. Fellenstein has served for many years as a Global Chief Architect for the IBM Corporation. His primary skills and experience are in the areas of technical support and large-scale Internet commerce infrastructure design, traditional infrastructures, electronic commerce transformations, and telecommunications, including development and deployment of global enterprise-wide architectures. Mr. Fellenstein is also a leading member of the IBM Global Services Invention Evaluation Team, helping evaluators in complicated evaluations for many IBM patents being submitted to the U.S. Patent Office. Mr. Fellenstein has held chair assignments on the IBM Corporate Technology Council chaired by Lou Gerstner, solving difficult technology challenges within IBM. He is currently IBM's Global Chief Architect and Senior Executive Consultant. During the late 1990s, Mr. Fellenstein helped IBM establish its full line of hosting services, which were instrumental in the IBM Service Delivery Center deployment strategy and worldwide architecture. Mr. Fellenstein has several publications, including four books, in advanced areas of computer science, most illustrating an emphasis in systems design, technical architectures, grid computing, on demand business strategies, patents, and artificial intelligence. He currently has over 80 patents pending for IBM, plus other awards for patents on file with the U.S. Patent Office. He is a disabled veteran of the Vietnam era and a retired member of the U.S. Air Force Tactical Air Command, where he was an aerospace weapons delivery specialist on the A7-D fighter-bomber aircraft.