Introduction to Web services architecture

by K. Gottschalk
S. Graham
H. Kreger
J. Snell

This paper introduces the major components of, and standards associated with, the Web services architecture. The different roles associated with the Web services architecture and the programming stack for Web services are described. The architectural elements of Web services are then related to a real-world business scenario in order to illustrate how the Web services approach helps solve real business problems.

Web services technology is changing the Internet, augmenting the *eyeball web* with capabilities to produce the *transactional web*. The eyeball web is dominated by *program-to-user* business-to-consumer (B2C) interactions. The transactional web will be dominated by *program-to-program* business-to-business (B2B) interactions. This transformation is being fueled by the program-to-program communication model of Web services built on existing and emerging standards such as HyperText Transfer Protocol (HTTP), Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and the Universal Description, Discovery, and Integration (UDDI) project.

Web services technologies provide a language-neutral, environment-neutral programming model that accelerates application integration inside and outside the enterprise. Application integration through Web services yields flexible loosely coupled business systems. Because Web services are easily applied as a *wrappering* technology around existing applications and information technology assets, new solutions can be deployed quickly and recomposed to address new opportunities. As adoption of Web services accelerates, the pool of services will grow, fostering development of more dynamic models of just-in-time application and business integration over the Internet.

This paper presents a business scenario showing how Web services standards are used to solve problems in a business situation. Preceding this scenario is a brief overview of the major Web services concepts and standards.

Web services overview

A *Web service* is an interface that describes a collection of operations that are network-accessible through standardized XML messaging. A Web service performs a specific task or a set of tasks. A Web service is described using a standard, formal XML notation, called its *service description*, that provides all of the details necessary to interact with the service, including message formats (that detail the operations), transport protocols, and location. Web service descriptions are expressed in WSDL.

This paper describes Web services in terms of a service-oriented architecture. As depicted in Figure 1, this architecture sets forth three roles and three operations. The three roles are the service provider,

0018-8670/02/$5.00 © 2002 IBM

the service requester, and the service registry. The objects acted upon are the service and the service description, and the operations performed by the actors on these objects are publish, find, and bind.

A service provider creates a Web service and its service definition and then publishes the service with a service registry based on a standard called the Universal Description, Discovery, and Integration (UDDI) specification.

Once a Web service is published, a service requester may find the service via the UDDI interface. The UDDI registry provides the service requester with a WSDL service description and a URL (uniform resource locator) pointing to the service itself. The service requester may then use this information to directly bind to the service and invoke it.

For a more complete description of these Web services components, see the architecture document.[1]

The Web services programming stack. We now give a brief introduction to the Web services programming stack. This stack is a collection of standardized protocols and application programming interfaces (APIs) that lets individuals and applications locate and utilize Web services. After introducing the stack itself, we illustrate how each of its layers facilitates the use of Web services.

Prominent at each layer in the Web services programming stack is the standardization of simple, open protocols and APIs. This standardization is the key to the ubiquitous deployment of Web services architectures, and the ubiquitous deployment of the infrastructure is the key to the network effect of Web services adoption.

The network is the foundation layer for the Web services programming stack (Figure 2). All Web services must be available over some network. The network is often based on an HTTP protocol, but other kinds of network protocols, such as the Internet Inter-ORB** Protocol (IIOP**) or the IBM MQSeries*, are also used.[2]

On top of the networking layer is an XML-based messaging layer that facilitates communications between Web services and their clients. The messaging layer is based on SOAP. SOAP is an XML protocol that facilitates the publish, find, bind, and invoke operations described previously.[3]

Figure 1 Web services actors, objects, and operations

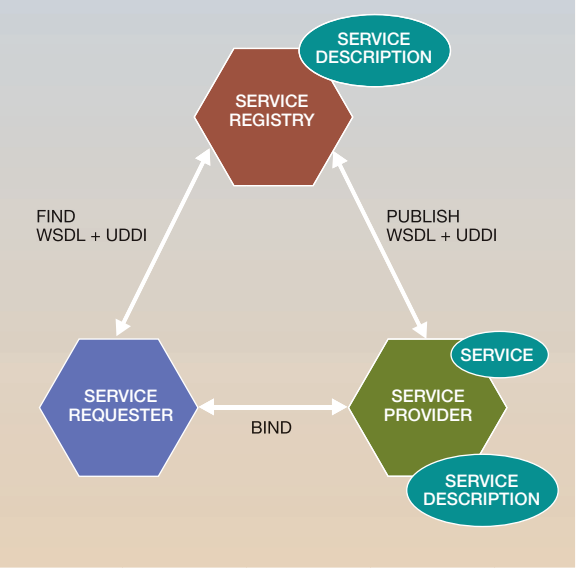
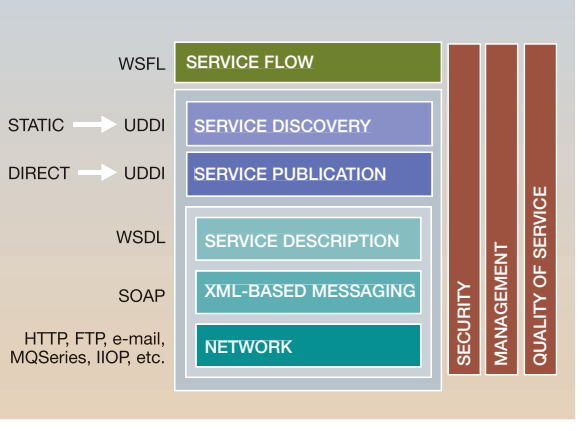


Figure 2 Web services programming stack



WSDL is a specification that describes available Web services to clients. These descriptions take the form of XML documents for the programming interface and location of Web services.[4]

The three layers described thus far are required in order to have interoperable Web services. These layers also create a low-cost entry for leveraging Web services by allowing these services to be deployed

over the Internet. The remaining layers in the programming stack are optional and will be used as business needs require them.

Publication of a service is really any action by the service provider that makes the WSDL document available to a potential service requester. Sending the WSDL (or a URL pointer to the WSDL) as an e-mail to a developer is considered to be publishing. Publishing is also advertising the WSDL in a UDDI registry for many developers or executing services to find.[5]

Likewise, discovery of a service is any action that gives the service requester access to the WSDL for a service. The action may be as simple as accessing a file or URL containing the WSDL or as complex as querying a UDDI registry and using the WSDL file(s) to select one of many potential services. The service flow layer of the stack facilitates the composition of Web services into workflows and the representation of this aggregation of Web services as a higher-level Web service. Standardization activity at this level is ongoing, but IBM has produced the Web Services Flow Language (WSFL) as its input to the standardization process.[6]

In order for a Web services application to meet the stringent demands of today's e-businesses, enterprise-class infrastructure must be supplied, including security, management, and quality-of-service management. These infrastructure items represented by the vertical towers on the right side of Figure 2 must be addressed at each layer of the stack. The solutions at each layer may be independent of one another. More of these vertical towers will emerge as the Web services paradigm is adopted throughout the industry.

We have briefly summarized the layers and standards in the Web services programming stack. In the next section, we present a scenario describing how these standards apply in the real world.

Applying Web services standards to a business scenario

In this section we show how Web services standards apply to a business situation.

The ClearSailing Corporation and its customers. Consider a hypothetical corporation called ClearSailing that provides a set of Web services oriented toward facilitating parts purchasing within a particular industry, let us say the sailboat manufacturing industry. ClearSailing has three sets of customers:

1. Merchants who sell sailboat parts to sailboat manufacturers
2. Buyers employed by sailboat manufacturers to procure sailboat parts
3. Procurement managers employed by sailboat manufacturers to establish purchasing policies for their companies

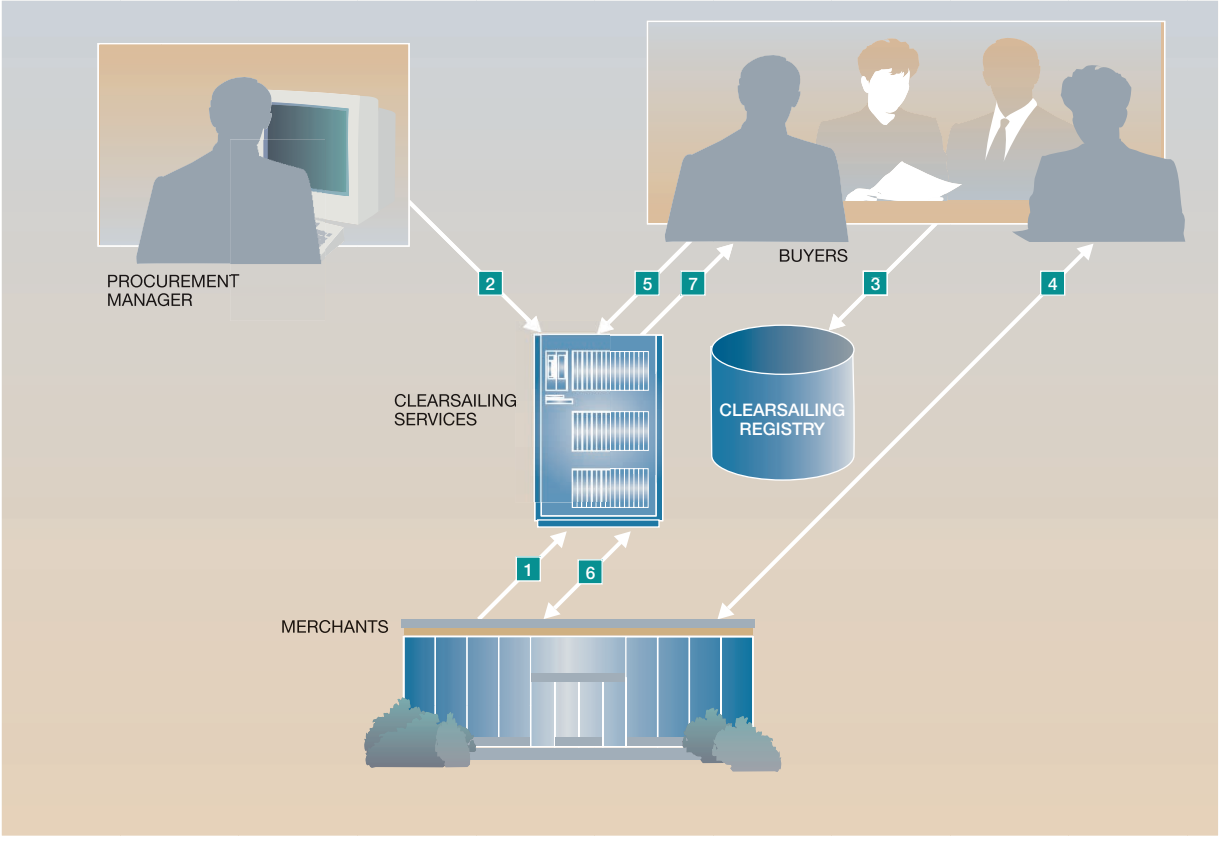
ClearSailing serves as a broker among these three sets of customers. Relationships among ClearSailing and its customers are illustrated in Figure 3. These relationships are indicated by the corresponding boxed numbers in the figure, as follows:

1. Merchants who wish to sell sailboat parts to manufacturers through ClearSailing register with ClearSailing, describing their goods and prices.
2. When a sailboat manufacturer wants to order goods through ClearSailing, its procurement manager sets up user profiles for each of the company's buyers, establishing purchase areas and purchase limits for each user.
3. Once each buyer's profile has been set up, the buyer can access the ClearSailing registry of merchants.
4. The buyer can then browse individual merchant's catalogs to determine which merchants have appropriate parts at the best combination of price and quality for that buyer's company.
5. The buyer can place items from multiple merchants into a single shopping cart and then submit his or her order to ClearSailing for execution.
6. When it receives the order, ClearSailing initiates a business process, validating the purchase against the company procurement policies defined by the procurement manager, submitting appropriate orders to the individual merchants, and updating order status.
7. Finally, ClearSailing reports the order status back to the buyer.

Success factors for ClearSailing Corporation. In order to be successful with its set of Web service applications, ClearSailing must devise and utilize software that meets a number of requirements. Among them are the following:

- ClearSailing must be able to exchange information programmatically with its users over a network.
- ClearSailing and its users must share a common

Figure 3 ClearSailing and its customers



set of data and message formats for things such as orders and catalog information.
- ClearSailing and its users must have a common understanding of the meaning of the contents of the messages they exchange. For example, a buy order must be commonly understood by the buyers, the merchants, and ClearSailing itself.
- ClearSailing must provide a mechanism to allow merchants to inform potential buyers that they have appropriate merchandise.
- ClearSailing must provide a mechanism to allow potential buyers to discover available merchandise that meets their needs.
- ClearSailing must have a way to combine its own service applications with those of its merchants in workflows that are appropriate for its business environment.
- In order to be commercially viable, ClearSailing must provide its customers with a way to ensure that their transactions are conducted in a secure environment, with appropriate quality of service (which may include guaranteed availability levels, transaction support, etc.).

How Web services standards help solve e-business problems. The Web services environment differs from previous programming environments in one respect only: Web services components (service applications and clients) use generally accepted standardized APIs to communicate at each level of the programming stack. In this subsection we describe how the Web services standards in the Web services conceptual stack help solve the problems mentioned above.

Networking standards for Web services. In order for ClearSailing to communicate with its users in an automated fashion, both must either share a common networking protocol or use a protocol converter to convert between the networking protocols each uses.

Until relatively recently, no single networking protocol has been pervasive. Such early networking protocols as the IBM Systems Network Architecture (SNA) did not generally extend across enterprise boundaries, so communications over these networks tended to be limited to users and servers within a single corporation. The same is true of distributed remote procedure call (RPC) -based protocols such as IIOP for CORBA** (Common Request Broker Architecture**) and the Microsoft Common Object Model (COM) protocol. The growth of the public network known as the Internet, based on Transmission Control Protocol/Internet Protocol (TCP/IP) and related protocols such as HTTP, has greatly extended the ability of companies such as ClearSailing to easily and automatically communicate with its users, whether these users are computers or human beings.

Although HTTP use within an enterprise is growing, multiple networking protocols are still often found. The basic communications requirement stated above still holds—there must either be one common protocol or a protocol converter—but the growth of TCP/IP and HTTP has greatly facilitated communications between a service provider and service requesters, and has greatly extended the potential number of users available.

In the example illustrated in Figure 3, procurement manager, buyers, and merchants are all communicating with ClearSailing services and the ClearSailing registry via HTTP. Within the ClearSailing Corporation itself, multiple networking protocols may be used, but in this case adapters will convert between these protocols and HTTP for messages that pass through the enterprise boundary to and from users.

Messaging standards for Web services. The common network described above circulates messages among service providers and requesters. However, in order for successful communications to occur, both the service providers and the requesters must agree to a common format for the messages being delivered so that they can be properly interpreted at each end. An order flowing from ClearSailing to a merchant must have an organization or message format that is understood at both ends.

Messages delivered over a network tend to fall into two general categories:

1. Messages that are composed primarily of a document that is to be processed remotely
2. Messages that contain commands and parameters that are used to directly invoke a remote procedure (i.e., remote procedure calls)

Until relatively recently, there was no common protocol for handling both types of messages; indeed, multiple protocols have been used to handle messages of each type. For example, products such as the IBM MQSeries have proprietary protocols for formatting and delivering messages of the first type listed above, whereas protocols such as Remote Message Invocation (RMI) for the Java** programming language and the Microsoft COM protocol have been used to format messages of the second type.

In the last few years, XML has gained widespread acceptance as a standard specification for data markup, validity checking, and tagging. XML greatly aids in the generation, validation, and machine interpretation of complex data structures or documents.

Built on top of XML, SOAP is the standardized mechanism for communicating document-centric messages and remote procedure calls using XML. In other words, SOAP, a standard owned by the World Wide Web Consortium (W3C) (as XML protocol), accommodates both types of messages described above. As such, it greatly enhances the ability of service providers and users to properly interpret the messages they are exchanging with one another.

In the example set forth in Figure 3, messages flowing between ClearSailing and its users (procurement managers, buyers, and merchants) flow in SOAP format in XML documents.

Service description standards for Web services. Given a common network for communicating and a common set of conventions for formatting and interpreting messages, what is the next requirement to facilitate communications between service provider and requester? They must have a common semantic understanding of the content of these messages—of what they mean to accomplish in their transactions over the network. A potential requester must know what services are available from the service provider, what message format is required to invoke them, what costs are involved, etc. A merchant who wants to use the service provider to sell goods must be able to describe them in such a way that the service provider can understand the descriptions and convey them to potential buyers.

Standardization of service descriptions to support Web services is achieved via WSDL. This language

defines the interface required for interaction between a requester and a service provider and also defines the location of the service provider. A service provider publishes a service by making its WSDL description document available to potential requesters. This can be done in a variety of ways, but one standardized way is for the service provider to register the service with a registry and for the service requester to discover the service by searching the registry. The specification used for the registry is the UDDI specification.

The area of workflow and business process definition is an important one to Web services.

In the example given above, a merchant who sells sails to sailboat manufacturers would describe wares in terms specified by ClearSailing and register them via a registry of preferred merchants for the sailing industry that might be maintained by ClearSailing or by an industry consortium. When a buyer for a sailboat manufacturer wanted to buy sails, the buyer would use this registry to discover and consider the wares of each registered seller of sails. A WSDL document would describe each merchant's offerings at a basic programming-interface level.

As Web services mature, industry groups will probably standardize WSDL descriptions of the services that are important for them and their customers.

Some business context descriptions for services have already been specified by UDDI, including categorization information on the type of business, geographic location, and contact information. In order to facilitate discovery and usage of appropriate services, further standardization is needed of descriptive material for specific industries and across industries; this work is ongoing in many industry groups.[7]

Service publication and service discovery standards for Web services. Service publication and service discovery go hand in hand. In the case of the ClearSailing Corporation, merchants must publish their services to the ClearSailing registry, and buyers must find these services by searching the registry for appropriate services. Obviously, for this operation to be successful, ClearSailing must provide standard APIs to its merchants and its buyers for publishing and finding services.

ClearSailing could have devised its own APIs for finding and publishing services and then told merchants and buyers that they needed to use these services. This route may create a burden on the merchants and suppliers who are dealing with multiple service providers, because they would have to customize their applications to work with each different service provider. In fact it may mean that some merchants and buyers do not use ClearSailing's registry. A standard is needed for publishing and finding Web services to make ClearSailing, the merchants, and the buyers successful. This standard is the UDDI standard mentioned previously, which provides standard sets of APIs for publishing and finding services. Thus, the registry illustrated in Figure 3 is assumed to be a UDDI registry.

There are two types of UDDI registry—public and private. Public registries are located at http://www.uddi.org and are maintained and synchronized by companies such as IBM and Microsoft. The registries at this URL are available, at no charge, to all users. Individual enterprises or industry consortiums maintain private UDDI registries and control what service data are registered and who can access the data. The registry illustrated in Figure 3 is an example of a private UDDI registry. ClearSailing controls this registry to ensure that only merchants and buyers from companies that meet its strict standards are allowed to publish and discover information there. In this way, ClearSailing performs a quality-control operation that facilitates trust among its users.

Service flow standards for Web services. When dealing with Web services, it is often desirable to automatically compose Web services into a workflow, aggregating several simpler services into a higher-level service. In the ClearSailing example, the procurement manager for a particular sailboat manufacturer may wish to specify that when a buyer enters an order over a certain amount, the system must obtain the approval of the procurement manager before proceeding. Another example of a workflow transaction is when ClearSailing fulfills an order, as described in steps 6 and 7 of the process set forth earlier for Figure 3. Here several Web services, some at ClearSailing and some at individual registered merchants' locations, are composed into a workflow.

The area of workflow and business process definition is an important one to Web services, and it is still under definition. IBM has put together a proposal for a Web services flow language that will serve as IBM's input into the standards process in this area.[6]

Infrastructure services. In order to be viable for e-business, Web services must possess the same characteristics that business applications in an enterprise must possess—reliability, availability, manageability, security, etc. Figure 2 shows some of these characteristics as vertical towers that apply to each of the horizontal layers in the stack. A company such as ClearSailing will want to provide its merchants, buyers, and procurement managers with good security characteristics (for example, authentication of buyers), good quality-of-service characteristics (for example, processing a transaction in a reasonable time and being available 24 hours a day), good management characteristics for its Web services, etc.

In traditional information processing environments, these requirements are typically met by middleware products such as the IBM WebSphere* Application Server and DB2* (DATABASE 2*) manager. To support Web services, these products and those from other vendors must be extended to support Web services standards. This work has begun and continues to mature. For interoperable Web services running on platforms supplied by multiple vendors, standards are once again essential.

Utility services. In order to be effective, infrastructure services often require tailoring to the Web services they support. In the ClearSailing example, a security service would need to know security characteristics for the buying service and for each of its potential clients, for example. A management service would need to know which management policy applies to each Web service being managed. In general, the standards specified for each level in the Web services conceptual stack set forth in Figure 2 will need to be extended to accommodate information associated with infrastructure services.

Infrastructure services are an example of utility services—services that exist to help business services achieve their goals. Utility services are used by business services to help them perform their business processes.

In the ClearSailing example, when ClearSailing puts together its ordering service for buyers, it might use a buyer validation security service, a shopping cart service, a catalog display service, and similar services provided by other vendors, and tie these utility services in with the services it codes and provides itself by means of a workflow utility service.

Conclusion

In this paper, we have presented a scenario illustrating how Web services may be applied to solve a real business problem, and we discussed both the benefits of Web services for business applications and the state of standards development for these services. Though the basic standards for Web services are now available, higher-level standards are still under development. Although valuable Web services are now being developed and deployed, widespread commercial exploitation of Web services across the public Internet awaits development and acceptance of higher-level standards in such areas as security, reliable messaging, transaction support, and workflow. Fortunately, IBM and other software vendors are working hard to help develop such standards and to have them widely accepted.

Cited references

1. The document *Web Services Conceptual Architecture* is located at http://www.ibm.com/software/solutions/webservices/documentation.html.
2. For more information on HTTP from the World Wide Web Consortium, see http://www.w3.org/Protocols/.
3. For more information on SOAP from the World Wide Web Consortium, see http://www.w3.org/TR/SOAP/.
4. For more information on WSDL from the World Wide Web Consortium, see http://www.w3.org/TR/wsdl.
5. The UDDI project is a cross-industry initiative to create an open framework for describing, discovering, and integrating Web services across the Internet. For more information on UDDI, see http://www.uddi.org.
6. For more information on WSFL, see the document titled *Web Services Flow Language Guide* at http://www.ibm.com/software/solutions/webservices/documentation.html.
7. See, for example, the standardization work being done by OASIS and XML.org (at http://www.oasis-open.org/) and by the ebXML work on an electronic business framework (at http://www.ebxml.org/).

Accepted for publication November 21, 2001.

Karl Gottschalk *IBM Software Group, P.O. Box 12195, 3039 Cornwallis Road, Research Triangle Park, North Carolina 27709 (electronic mail: karlgott@us.ibm.com).* Mr. Gottschalk is a Web Services Technical Strategist in the Emerging e-Business Tech-

nologies area of the IBM Software Group. He has helped formulate IBM's Java, XML, and Web services strategies for the past four years. He joined IBM in 1968 and has held positions in the areas of program design, program development, program maintenance, and information development. Mr. Gottschalk has published several articles in the *IBM Systems Journal* on the topics of usability, system and network management, and enterprise Java. He holds an M.A. degree in English literature from the University of Mississippi, an M.S. degree in computer science from the University of North Carolina at Chapel Hill, an M.B.A. degree from Duke University, and an M.A. degree in liberal studies from Duke University.

Stephen Graham *IBM Software Group, P.O. Box 12195, 3039 Cornwallis Road, Research Triangle Park, North Carolina 27709 (electronic mail: sggraham@us.ibm.com).* Mr. Graham is an architect in the Emerging Technologies organization in the IBM Software Group. He has spent the last several years working on service-oriented architectures, most recently as part of the IBM Web Services Initiative. Prior to this work, he was involved as a technologist and consultant with various emerging technologies such as Java and XML, and before that he was an architect and consultant with the IBM Smalltalk consulting organization. Before joining IBM, he was a developer with Sybase, a consultant, and a faculty member in the Department of Computer Science at the University of Waterloo. Mr. Graham holds a B.Math degree and an M.Math degree in computer science from the University of Waterloo.

Heather Kreger *IBM Software Group, P.O. Box 12195, 3039 Cornwallis Road, Research Triangle Park, North Carolina 27709 (electronic mail: kreger@us.ibm.com).* Ms. Kreger is a senior architect in the Emerging Technologies area of the IBM Software Group. She represented IBM as a member of the Java Management eXtensions (JSR0009) Expert Group. Her years in lead positions in network management, combined with her experience on the IBM Web server and WebSphere Application Server products, gives her unique insight into the problems and solutions for managing applications and e-business. She has contributed to the specification, reference implementation, and compatibility test suites for the JMX Reference Implementation and authored "Java Management Extensions for application management" in Vol. 40, No. 1, 2001, of the *IBM Systems Journal*. She was also involved in management issues with other standards bodies, including: The Open Group Management Program, the DMTF (Distributed Management Task Force) Application Management Work Group Chair, and WBEM (Web-Based Enterprise Management) JSR (Java Specification Request) Expert Group. Ms. Kreger is currently the lead architect for Web Services in Emerging Technologies, and recently authored the document, "Web Services Conceptual Architecture." She chairs the IBM Web services architecture team, holds an associate position on the IBM AIM Architecture Board, and is the Specification Editor for JSR109: *Implementing Web Services in the Enterprise* being led by IBM.

James Snell *IBM Software Group, 4400 Silicon Drive, Raleigh, North Carolina 27713 (electronic mail: jasnell@us.ibm.com).* Mr. Snell is an architect and strategist in the Emerging e-Business Technologies area of the IBM Software Group. He is the voice of the Web Services Insider on IBM's developerWorks Web Services Zone and a coauthor of a new book on SOAP-based Web services. He joined IBM in March 2001 and has been focused primarily on the area of Web services development. Prior to joining IBM, Mr. Snell was an enterprise solution architect working for a variety of independent software companies. He has been actively involved in Web-based development since 1994.