Editor: Siobhán Clarke • siobhan.clarke@cs.tcd.ie

Web Services Management: A Survey

Michael P. Papazoglou and Willem-Jan van den Heuvel • *Tilburg University*

Solutions based on service-oriented architectures are promising in that they leverage common services and enable collaborative business processes that cross organizational boundaries. However, because Web services applications can span multiple hosts, operating systems, languages, and enterprises, it's problematic to measure, control, and manage application availability and performance. In addition to discussing the relationship of Web services management to traditional distributed systems management, this survey explores various Web services management approaches and their underlying architectural concepts.

Web services is now the most popular paradigm for distributed computing. By adopting service-oriented architectures (SOAs) using Web services technologies, enterprises can flexibly solve enterprise-wide and cross-enterprise integration challenges and thereby quickly address ever-changing business challenges and opportunities. Because Web services increasingly play mission-critical roles, the need to monitor and measure them has never been greater. Once services and business processes become operational, their progress must be managed and monitored to offer a clear view of how services perform within their operational environments; to enable management decisions; and to perform control actions to modify and adjust the behavior of Web-services-enabled applications. These capabilities require distributed management solutions for Web services.

In distributed systems management, tool suites and utilities monitor distributed system activities to facilitate management decisions and system behavior modifications. Traditionally, distributed systems and network management have focused on managing successive layers of new technologies and devices as they were deployed into distributed computing environments. However, such techniques fail to meet enterprise requirements in a service-centric world because they lack the business-awareness functions required by services management products. Also, many existing system management infrastructures don't support service-level agreement (SLA) reporting or collect specific service information from Web services applications for troubleshooting purposes.

In service-oriented solutions, administrators should continuously collect and analyze service usage patterns, SLA criteria and metrics, and failure data. Without such information, it's often quite challenging to understand the root cause of an SOA-based application's performance or stability issues. In particular, enterprises must monitor and measure service levels for distributed and federated processes. By watching over business operations, analysts and administrators can identify opportunities and diagnose problems as they occur, and thus ensure that the Web services supporting a given business task are performing in accordance with service-level objectives.

Currently, management vendors offer only instrumentation at SOAP endpoints and intermediaries or at UDDI servers. Although this provides information about Web services while the applications use them, the view is clearly incomplete and lacks critical information regarding the Web services' state and characteristics. To derive such information, Web services must become measurable and much more manageable. This can be achieved only through a standard approach, which

 Published by the IEEE Computer Society 1089-7801/05/$20.00 © 2005 IEEE

can provide, among other things, end-to-end visibility and control over all parts of a long-lived, multistep transaction spanning multiple applications, human actors, and enterprises. Web services management is therefore expected to become a critical component of production-quality Web services applications in the near future. Here, we discuss distributed services and Web services management issues, exploring SOA concepts and how they address key requirements.

Distributed Services Management

A distributed application-management system controls and monitors an application throughout its life cycle, from installing and configuring to collecting metrics and tuning the application to ensure responsive execution. The system's functionality must cover all operational activities — including starting and stopping processes, rerouting operations, and even rebooting servers — as well as problem-detection functions, such as application tracing and message editing.

Managing a distributed computing environment requires reliable and efficient processes for both systems administration and network management. Such processes provide programs, tools, and utilities that give systems administrators and network mangers insights into system and network health, as well as into application status and behavior patterns. Although toolsets and utilities have existed in traditional distributed computing environments for many years, we've yet to develop equivalent technologies for the complex world of loosely coupled, Web-services-based applications.

Using Web services to expose and extend legacy applications offers businesses clear technology advantages. Configuring services into discrete logical components aligned directly with business functions can help achieve advantages that are even more significant. Businesses can "orchestrate" such services into many different configurations to support multiple business processes. An orchestrated service implementation forms a logical network of services layered over the infrastructure (including software environments, servers, legacy applications, and back-end systems) and the physical connectivity network. As such a Web services network grows, its existence and performance become as relevant to the business's core activities as other critical IT resources.

Quality Requirements

For a business to perform its intended task, the source provider and customer must agree on mutual expectations about the offered service. In applications that integrate geographically dispersed Web services, such a contract constitutes an SLA.

The enterprise SLA's goal is ultimately to improve the service's quality of experience (QoE) to its internal or external clients. QoE entails a shared foundation for measuring service or product quality, including performance, overall customer satisfaction, pre- and post-sale service, and product and service delivery. To achieve an effective contract, the provider often must establish and monitor several SLAs, whether internally (for business or end users) or externally (for service providers, partners, outsourcers, suppliers, and so on). Generally, a business process involves multiple SLAs governing different services that cooperate to fulfill an end user's request. For every tier or peer SLA, one party can be considered a supplier or provider and the other a customer.

Ensuring quality, service-focused management requires measurable key performance indicators (KPIs) for applications and services. A service provider can first consider a KPI generically before applying it to particular applications. Generic KPIs for business applications include service availability, response time, transaction rate, service throughput, service idle time, and security. The service provider must know whether the services it has contracted to deliver are meeting its KPI objectives and issue warnings and alarms if they fail to do so. Service providers must therefore set the service KPIs and their objectives, and be able to measure, analyze, and report on the KPI values achieved against these objectives.

Web services, on their own, can't fulfill these service-focused management functions. There is therefore a critical need for a Web services measurement and management infrastructure and toolset.

Conceptual Management Architecture

A system's management technologies monitor and respond to managed-resource-related situations in the environment, basically functioning as a control loop. They collect details regarding a managed resource and act accordingly. As Figure 1 shows, we can view the management infrastructure as a conceptual architecture that fuses various components, including resources, resource managers, and a management console.

The management console displays information through an interface that lets administrators control and supervise manageable resources and

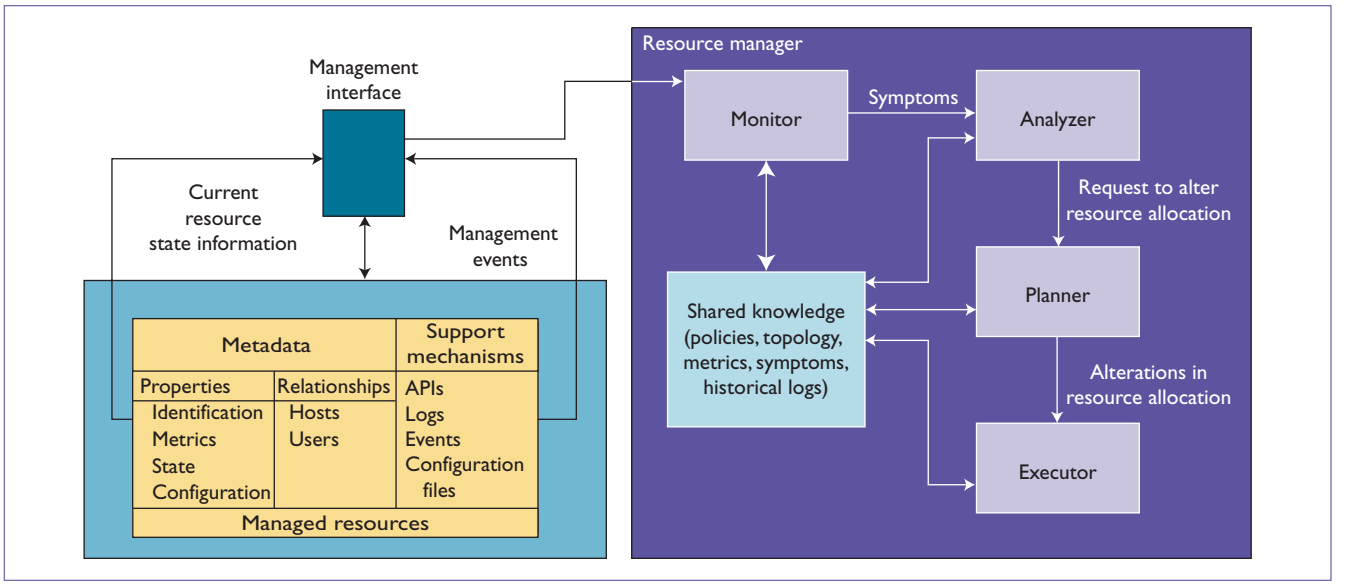

Figure 1. Conceptual management architecture. The architecture's components include resources, resource managers, and a management console.

change their behaviors. The management interface uses mechanisms such as log files, events, commands, APIs, and configuration files.[1] As the lower left box in Figure 1 shows, the interface also uses metadata about the resource's properties, including

- *identification*, which denotes an instance of the managed resource;
- *metrics*, which provide information and operations for resource measurements, such as throughput, utilization, and so on; and
- *configuration*, which provides information and operations for the configurable attributes of a managed resource.

Management interfaces also require information about a resource's relationships with its immediate environment, including its users and hosts. Finally, the interface retrieves information about the current state of a managed resource and the management events (notifications) that might occur when the resource undergoes significant state changes.

Distributed Management Frameworks

Several standard distributed management frameworks are currently in wide use; the most prominent include the Simple Network Management Protocol (SNMP; see www.ietf.org/rfc/rfc1157.txt); the Common Information Model (CIM; see www.dmtf.org/standards/cim/); Web-Based Enterprise Management (WBEM; see www.dmtf.org/standards/wbem/); and the Open Management Interface (OMI; see www.openview.hp.com/downloads/try_omi_0001.html).

The IETF's SNMP is an application-layer protocol that facilitates the exchange of management information between network devices.[2] SNMP's network elements are devices — such as hosts, gateways, and terminal servers — with agents that perform the functions requested by network management stations. The SNMP framework lets network administrators manage network performance, find and solve network problems, and plan for network growth. SNMP is today's de facto industry standard for network management.

CIM, developed by the Distributed Management Task Force (DMTF), describes managed elements across the enterprise, including systems, networks, and applications. CIM is implementation-independent, letting different management applications collect required data from various sources. The DMTF also developed WBEM, a set of management and Internet standard technologies that aims to unify enterprise computing environment management. With WBEM, the industry can deliver a well-integrated set of standards-based tools that leverage emerging Web technologies.

Jointly authored by Hewlett-Packard and webMethods,[3] OMI's goal is to provide an easy, open way for systems management vendors and other interested parties to access and manage the resources associated with an integration platform and its associated business processes. OMI lets

management frameworks access a business integration platform's management capabilities by providing a programmatic interface into the platform's management aspects. Through this interface, consumers can manipulate a set of OMI objects that represent the available resources.

Web Services Management

As traditional, centralized management applications transition to highly distributed and dynamic SOAs, they can more flexibly deploy essential management functions. With SOAs, the Web services management framework (WSMF) can provide support for discovering, introspecting, securing, and invoking resources, management functions, and management infrastructure services and toolsets. As such, the same technologies that define and execute business processes and higher-level IT management processes can now interact directly with resource managers, infrastructure services, and resources.

Service Management Approaches

All essential communication between service providers and consumers is carried out using standardized SOAP messages, which facilitate Web services management in several ways. First, developers can use SOAP messages to mark the beginning and endpoints of service transactions. Also, because a SOAP message's source and destination are readily identifiable, it's easy to obtain detailed performance measures for applications that span platforms and enterprises.

Although it's not always possible to manage and control the Web services infrastructure, organizations can monitor the stream of SOAP messages that applications exchange. Indeed, sniffing and altering SOAP messages — which can be intercepted at many points in the network of applications connected via Web services — is the main underlying principle of Web services management. Service containers facilitate the request–response message-flow environment. Like J2EE containers, the Web service containers entail the physical manifestation of an abstract service endpoint and provide the service interface implementation.

Administrators can inject service management capabilities into the SOAP pipeline using either an agent-based or intermediary approach. Agent-based service management reflects a purely container-based style of Web services management. Here, the platform's native Web service container hosts the management capabilities. From an architectural perspective, the model is beneficial in that the agent can leverage the infrastructure configuration of the business application container (the application server), which administrators can cluster and secure for required service levels.

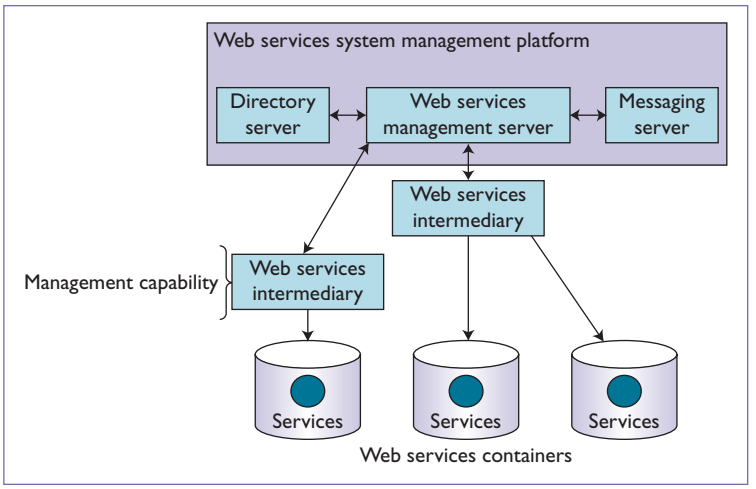

Figure 2. Intermediary-based management scenario. Management capabilities are deployed on the intermediary, which acts as a client endpoint. Following monitoring and control, the intermediary passes the request to the service endpoint.

Unlike the agent-based scenario, which directly links the Web services management servers to containers, in the intermediary approach, an intermediary or broker acts as the client endpoint, then passes the request to the actual service endpoint (see Figure 2). Administrators deploy management capabilities on the intermediary, allowing for monitoring and control. The intermediary model is completely decoupled from the Web services nodes, and can be used in any environment. Services from multiple platforms can be routed through the intermediary, which also facilitates central management and configuration.

The intermediary model is particularly useful in simplifying administration and technical support without requiring that an agent be installed for each individual Web services node. However, intermediaries do require network address reconfiguration. Also, if there's no infrastructure to cluster the Web services intermediaries, they can constitute single points of failure and thus create problems in environments that require high reliability.

Management Infrastructure Services

To manage services, the WSMF uses the *manageability capabilities* of various architecture resources. These capabilities define standard schemas, metadata, and Web Services Description

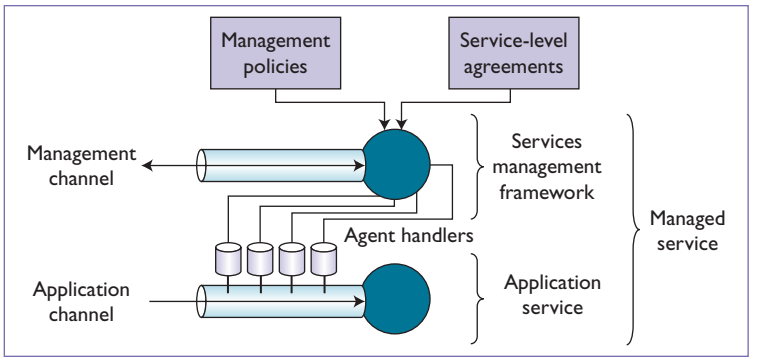

Figure 3. Integrating application and management-based services. The Web services management framework is positioned between Web services consumers and providers and offers a centralized management and policy framework.

Language (WSDL) interfaces that describe resource-specific behaviors that resources use to advertise — and provide access to — manageability information.[4] Typically, resources implement only applicable selections rather than all of their standard manageability capabilities, which include operational status, metrics, and relationships.

Manageability capabilities are realized through a *manageability information model* that represents resource manageability and related information, such as state, configuration, and relationships.[5] The service management framework uses this model's information to retrieve information and exert control. Manageability information might, for example, indicate that a SOAP message header contained a digital certificate of the client's identity. The WSMF would then extract that information and translate it into the client's identification for later use (such as when counting a particular client's SOAP messages to a particular Web service).

The WSMF uses management infrastructure services to define standard interfaces for commonly available functions in the infrastructure's information.[4] Such services include metering, metric and event mediation, monitoring, system scanning, policy enforcement, and model bridges. The standard management interfaces enable higher-value utility functions, processes, or applications, such as availability and performance management, optimization, capacity planning, billing, configuration management, asset protection, problem determination, and business analytics.

Service management systems manage resources using their interfaces, infrastructure services, or other available tools. These mechanisms provide a range of functions, including simple monitoring, control, autonomic tuning and recovery, sophisticated quality management, end-to-end business processes management, and policy-driven system and service provisioning. The mechanisms also offer interfaces and content for management consoles.

A Web service is manageable if its manageability capabilities are exposed via standard interfaces, which are similar to a Web service's functional interfaces.[5] The management interfaces differ only in that they convey management-related semantics and that a Web services management system, rather than a client, uses them.

Figure 3 shows management interfaces exposed by a manageable Web service and accessible by the WSMF. A WSDL document describes the management interfaces and the endpoints to which the WSMF can send messages with management-related payloads. The WSMF is logically positioned between Web services' providers and consumers.

To manage event-driven processes, the management service might support an event mechanism for notification push. Because they're in the same memory space, the agent handlers and management service interact via direct message calls. In the memory space, information from the management channel controls the agent handlers. Any information the handlers collect is passed to the management service, which then makes it available on the management channel.

As Figure 3 shows, along with distributed monitoring and policy enforcement, the WSMF offers a centralized management and policy framework. This framework works in concert with service directories and identity management solutions both in receiving information (such as service performance statistics, alerts, and service interdependency information) and sending information. Policy, in the form of rule sets, is sent to the service-management components. These rules define service-management component behavior — that is, when to raise alerts, how to process transiting message traffic, where to route messages, how to enforce security policies and SLAs, and so on.

Connecting Service Management and Application Channels

In addition to traditional application service development facilities, a Web-services-based SOA requires two key features to effectively manage systems and applications:

- a management framework that is consistent across an increasingly heterogeneous set of participating component systems; and

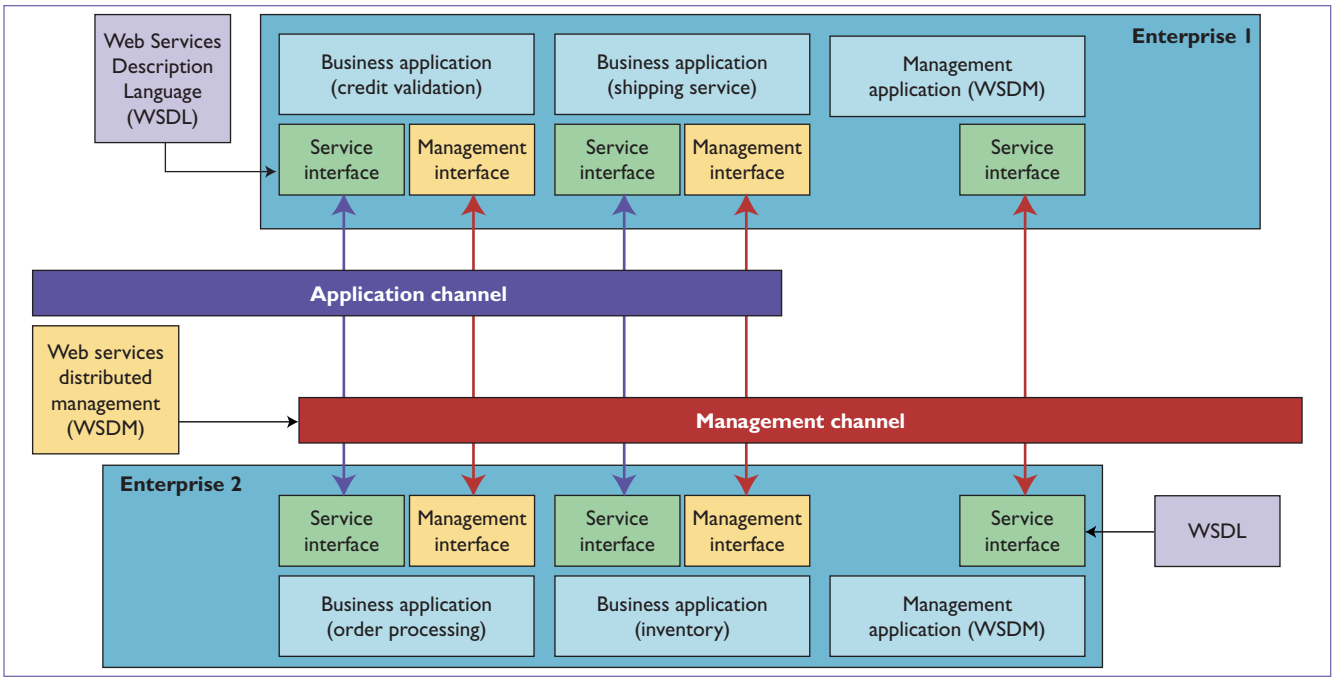

Figure 4. Developing and managing Web-services-based applications. The architecture combines service management and application channels in accordance with service-oriented architecture principles, and keeps the two channels continuously connected.

- support for complex, cross-component management scenarios, such as SLA enforcement and dynamic resource provisioning.

Figure 4 shows the conceptual architecture components that combine service management and application channels developed in accordance with SOA principles. This architecture continuously connects to the Web services application channel and directs it into the management channel. Among the management applications it can serve are availability and performance management, configuration management, capacity planning, asset protection, job control, and problem determination.

In this architecture, service management involves a collection of services that communicate with each other — passing data or coordinating some activity — to facilitate the delivery of one or more business services. Rather than prescribe a particular management protocol or instrumentation technology, the architecture can work with Simple Network Management Protocol, Java management extensions, Web-Based Enterprise Management, and other existing and future technologies and standards.

In Figure 4, manageable resources include physical and logical hardware and software resources. These resources expose their management capabilities as Web services that implement various interfaces, such as those defined in the Web Services Distributed Management Initiative (see the sidebar, "A Web Services Management Standard"). A resource's management interface is described by a WSDL document, resource properties schema, metadata documents, and (potentially) a set of management-related policies.

Resource managers can directly access resources as part of a business or management process. As Figure 4 shows, a business process integrates basic services such as credit validation, shipping, order processing, and inventory services originating from two collaborating enterprises. The architecture's management applications manage resources through their interfaces or infrastructure services.[4] These managers provide a range of functions, from simple monitoring to control, autonomic tuning and recovery, sophisticated quality management, end-to-end business processes management, and policy-driven provisioning of systems and services. They typically provide interfaces and content for management consoles and display management-related information. Managers interact with resources and infrastructure services using the Web services interfaces. In addition, service managers leverage Web services technologies, such as Web Services Business Process Execution Language (WS-BPEL), to

A Web Services Management Standard

Consistent end-to-end Web services management is impossible without industry-wide standards development. To address this, the Organization for the Advancement of Structured Information Standards (Oasis) is actively developing the Web Services Distributed Management specification (www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm). WSDM defines a protocol for the interoperability of management information and capabilities via Web services. To resolve distributed system management problems, WSDM focuses on two distinct tasks: management using Web services (MUWS) and management of Web services (MOWS).[1]

MUWS addresses the use of Web services technologies as the foundation of a modern distributed systems management framework. This includes using Web services to facilitate interactions between managed resources and management applications. In particular, MUWS defines how to describe managed resources' manageability capabilities using WSDL documents. Expressing these capabilities enables more efficient discovery and introspection of resources: because managers typically focus on a particular management task or domain, they must be able to easily and efficiently determine a manageable resource's relevant capabilities.

With MOWS, the WSDM addresses the specific requirements for managing Web services themselves. In WSDM, Web services are the platform for providing essential distributed computing functionality, interoperability, loose coupling, and implementation independence. The MOWS specification is based on the MUWS specification's concepts and definitions. As with MUWS, MOWS aims to build on existing model frameworks to define a Web service's management model, rather than reinvent a general managed-resource object model scheme.

Reference

1. W. Vambenepe, ed., *Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 1 and 2*, Organization for the Advancement of Structured Information Standards, Mar. 2005; www.oasis-open.org/committees/download.php/11819/wsdm-muws-part1-1.0.pdf.

describe and execute management processes that perform "scripted" management functions.

SOA-based management and application-development solutions encourage implementers to support multiple binding and marshalling techniques and leverage common management services, such as

- SLA and quality-of-service management, including performance and availability measurement and alerting services;
- visibility and control capabilities, including interactive monitoring, administration, and reporting;
- service adaptability, including versioning, routing, differentiated services, and message transformation; and
- Web-services and XML-based security mechanisms.

Progressive enterprises building service-delivery structured based on this architecture can extend business applications with special management capabilities as required. Because the architecture lets businesses mix and match proven, standards-based service management capabilities, it also ensures well-controlled enterprise agility.

References

1. "An Architectural Blueprint for Autonomic Computing," white paper, IBM, Oct. 2004; www-3.ibm.com/autonomic/pdfs/ACwpFinal.pdf.
2. J.D. Case et al., *Introduction to Version 3 of the Internet-Standard Network Management Framework*, IETF RFC 2570, Apr. 1999; www.rfc-editor.org/rfc/rfc2570.txt.
3. G. Bullen et al., *Open Management Interface Specification*, v. 1.0, revision 1, Organization for the Advancement of Structured Information Standards, 2001; www1.webmethods.com/PDF/OMI_Spec.pdf.
4. H. Kreger et al., *Management Using Web Services: A Proposed Architecture and Roadmap*, tech report, IBM, Hewlitt-Packard, and Computer Assoc., June 2005; www-128.ibm.com/developerworks/library/specification/ws-mroadmap.
5. M. Potts, I. Sedukhin, and H. Kreger, *Web Services Manageability — Concepts (WS-Manageability)*, tech. report, IBM, Computer Assoc., and Talking Blocks, Sept. 2003; www3.ca.com/Files/SupportingPieces/ web_service_manageability_concepts.pdf.

Michael P. Papazoglou is a professor of computer science and director of the Infolab at the University of Tilburg, the Netherlands. His research interests include distributed systems, service-oriented computing and Web services, enterprise application integration, and e-business technologies and applications. Papazoglou has a PhD in computer systems engineering from the University of Edinburgh. Contact him at mikep@uvt.nl.

Willem-Jan van den Heuvel is an associate professor of information systems at the University of Tilburg in the Netherlands. His research interests include service-oriented computing, alignment of new enterprise system with legacy systems, and system evolution. Van den Heuvel has a PhD in computer science from the University of Tilburg. Contact him at wjheuvel@uvt.nl.