This article appeared in the
February 2005 issue of

business integration
JOURNAL

Extending the Service-Oriented Architecture

By Michael P. Papazoglou

Service-Oriented Architectures (SOAs) provide major advantages by presenting the interfaces that loosely coupled connections require. Service-oriented computing is based on the premise that logic (including business, computational, data access, etc.) can be organized in ways that make it independent of the context in which it's being used. Services in SOA can be combined in ways not initially envisioned at design or implementation time.

Web service technologies offer building blocks that are highly decoupled from each other. The current deployment of Web services technology is a promising early initiative. More work is required, however, for this technology to deliver on all its promises and fully support strategic, long-lived Web service transactions. Specifically, the work required is development of more robust standards and protocols for SOAs relating to security, coordination, and management of processes.

Currently, the basic SOA doesn't address overarching concerns such as management, service choreography and orchestration, service transaction management and coordination, security, and other concerns that apply to all components in a services-based architecture. Such concerns are addressed by the Extended SOA (xSOA) shown in Figure 1.

The architectural layers in the xSOA describe a logical separation of functionality in such a way that each layer defines a set of roles and responsibilities and leans on constructs of its predecessor layer to accomplish its mission. The logical separation of functionality is based on the need to separate basic service capabilities provided by the conventional SOA (e.g., building simple applications) from more advanced service functionality needed for composing services and the need to distinguish between the functionality for composing services from the management of services. As shown in Figure 1, the xSOA uses the basic SOA constructs as its bottom layer and layers service composition and management on top of it.

The basic services layer in the xSOA is identical to SOA in that it defines an interaction between software agents as an exchange of messages between service requestors (clients) and service providers. Providers are responsible for publishing a description of the service(s) they provide. Clients must be able to find the description(s) of the services they require and must be able to bind to them. The interactions involve the publishing, finding, and binding of operations. For reasons of conceptual simplicity, we assume in Figure 1 that service clients, providers, and aggregators can act as service brokers or a service discovery agency and publish the services they deploy.

In a typical service-based scenario, employing the basic services layer in the xSOA, a service provider hosts a network-accessible software module (an implementation of a given service). The service provider defines a service description of the service and publishes it to a client (or service discovery agency) through which a service description is published and made discoverable. The service client (requestor) uses a find operation to retrieve the service description typically from a registry or repository, such as Universal Description, Discovery and Integration (UDDI), and uses the service description to bind with the service provider and invoke the service or interact with service implementation. Service provider and service client roles are logical constructs and a service may exhibit characteristics of both.

The service composition layer in the

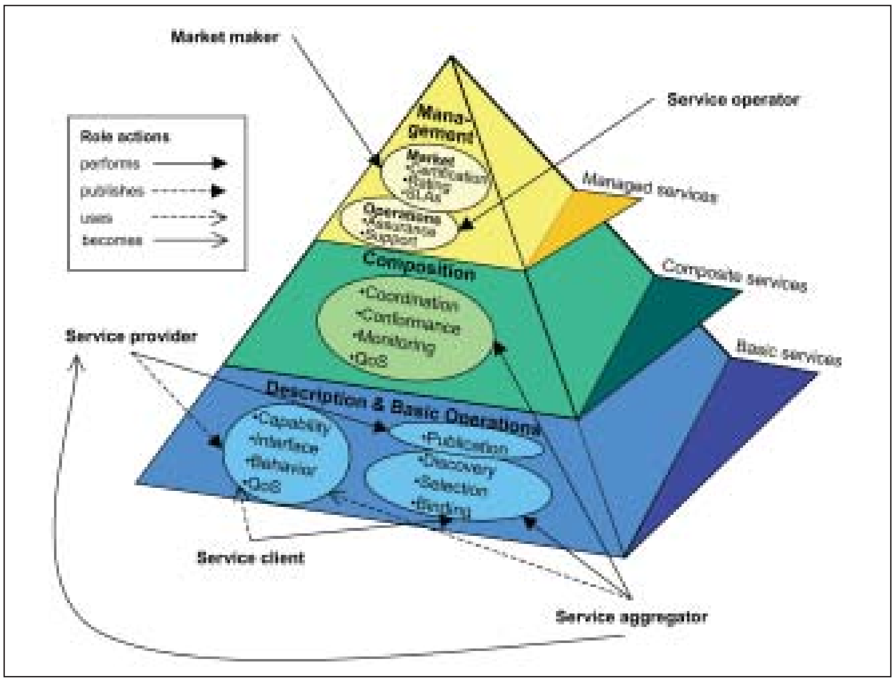

Figure 1: The Extended Service-Oriented Architecture

xSOA encompasses necessary roles and functionality for the consolidation of multiple services into a single composite service. Resulting composite services may be used by service aggregators as components (i.e., basic services) in further service compositions or may be used as applications by service clients. Service aggregators become service providers by publishing the service descriptions of the composite service they create. A service aggregator is a service provider that consolidates services provided by other service providers into distinct, value-added services. Service aggregators develop specifications or code that permit the composite service to perform functions that include the following (in addition to Web services interoperation support provided by the WS-Interoperability standard):

- **Coordination:** controls the execution of the component services and manages dataflow among them and to the output of the component service (e.g., by specifying workflow processes and using a workflow engine for run-time control of service execution)
- **Monitoring:** allows subscribing to events or information produced by the component services, and publishing of higher-level composite events (e.g., by filtering, summarizing, and correlating component events)
- **Conformance:** ensures the integrity of the composite service by matching its parameter types with those of its components, imposes constraints on the component services (e.g., to ensure enforcement of rules), and performs data fusion activities
- **Quality of Service (QoS) composition:** leverages, aggregates, and bundles the component's QoS to derive the composite QoS, including the composite service's overall cost, performance, security, authentication, privacy, (transactional) integrity, reliability, scalability, and availability
- **Policy enforcement:** Web service capabilities and requirements can be expressed in terms of policies. For example, knowing that a service supports a Web services security standard, such as WS-Security, isn't enough information to enable successful composition. The client needs to know if the service actually requires WS-Security, what kind of security tokens it's capable of processing, and which one it prefers. The client must also determine if the service requires signed messages. If so, it must determine what token type must be used for the digital signatures. Finally, the client must determine when to encrypt the messages, which algorithm to use, and how to exchange a shared key with the service. Trying to orchestrate with a service without understanding these details will lead to erroneous results.

Managing critical, Web service-based applications requires in-depth administration capabilities and integration across a diverse, distributed environment. For instance, any downtime of key e-business systems has a negative impact and can't be tolerated. To counter such a situation, enterprises need to constantly monitor the health of their applications. The performance should be continuously in tune under all load conditions. Web service-based application management is an indispensable element of the xSOA that includes performance management and business- or application-specific management. This requires that a critical characteristic be realized: that services be managed. Service management includes many interrelated functions; the most typical include:

- **Deployment:** The Web services support environment should let the service be redeployed (moved) around the network for performance, redundancy for availability, or other reasons.
- **Metrics:** The Web services support environment should expose key operational metrics of a Web service, at the operation level, including such metrics as response time and throughput. In addition, it should allow auditing of Web services.
- **Dynamic rerouting:** The Web services support environment should support dynamic rerouting for failover or load balancing.
- **Lifecycle/state management:** The Web services support environment should expose the current state of a service and permit lifecycle management, including the ability to start and stop a service.
- **Configuration:** The Web services support environment should support the ability to make specific configuration changes to a deployed Web service.
- **Change management and notification:** The Web services support environment should support the description of versions of Web services and notification of a change or impending change to the service interface or implementation.
- **Extensibility:** The Web services support environment should be extensible and must permit discovery of supported management functionality.
- **Maintenance:** The Web services support environment should allow for the management and correlation of new versions of the service.

Web services manageability could be defined as the functionality required for discovering the existence, availability, performance, health, patterns of usage, extensibility, as well as the control and configuration, lifecycle support and maintenance of a Web service or process in the context of the extended services architecture. This definition implies that Web services can be managed using Web services technologies. In particular, it sug-

business integration journal takeaways

BUSINESS

- The Extended Service-Oriented Architecture (xSOA) provides a design-oriented solution that reuses proven resources to deliver integrated business processes that meet ever-evolving business requirements.
- The xSOA has the potential to make it faster and easier for enterprises to create value-added applications that repurpose existing functionality, allowing them to deliver more relevant solutions to business problems—on demand—as new situations may arise.

TECHNOLOGY

- The xSOA provides a logical separation of functionality based on the need to separate basic service capabilities from more advanced service functionality such as the need for composing and managing services.
- The xSOA introduces a new philosophy for building distributed applications where services can be published, discovered, orchestrated, and managed to deliver complex, value-added service solutions and composite application assemblies.
- The xSOA is an attempt to increase the value of integrated business process solutions by coordinating and managing the interaction of significant numbers of applications across enterprise boundaries.

gests a manageability model that applies to both Web services and processes in terms of manageability topics (identification, configuration, state, metrics, and relationships), and the aspects (properties, operations, and events) used to define them. These abstract concepts apply to understanding and describing the behavior of any resource, including processes and Web services.

To manage critical applications or solutions and specific markets, xSOA provides managed services in the service management layer at the top of the xSOA pyramid. The xSOA-managed services are divided into two complementary categories:

- Service operations management that can be used to manage the service platform, the deployment of services and the applications and, in particular, monitor the correctness and overall functionality of aggregated services.
- Open service marketplace management that supports typical supply chain functions and by providing a comprehensive range of services supporting industry-trade, including services that provide transaction negotiation and facilitation, financial settlement, service certification and quality assurance, rating services, service metrics, and so on.

The xSOA's service operations management functionality is aimed at supporting critical applications that require enterprises to manage the service platform, the deployment of services, and the applications. xSOA's service operations management typically gathers information about the managed service platform, Web services and processes, managed resource status and performance, and supporting specific management tasks. These tasks include root cause failure analysis, Service Level Agreement (SLA) monitoring and reporting, service deployment, and lifecycle management and capacity planning.

Operations management functionality may provide detailed application performance statistics that support assessment of the application effectiveness, permit complete visibility into individual processes and transactions, guarantee consistency of service compositions, and deliver application status notifications when a particular activity is completed or when a decision condition is reached. The organization responsible for performing such operation management functions is the service operator. Depending on the application requirements, a service operator may be a service client or aggregator.

In service operations management, it's increasingly important for management to define and support active capabilities vs. traditional passive capabilities. For example, rather than merely raising an alert when a given Web service is unable to meet the performance requirements of an SLA, the management framework should be able to take corrective action. This action could involve rerouting requests to a backup service that's less heavily loaded, or securing a new application server with an instance of the software providing the service if no backup is currently running and available.

Service operations management should also provide global visibility of running processes, comparable to that provided by Business Process Management (BPM). BPM promises the ability to monitor both the state of any single process instance and all instances in the aggregate, using present, real-time metrics that translate actual process activity into Key Performance Indicators (KPIs). Management visibility is expressed in the form of real-time and historical reports, and in triggered actions. For example, deviations from KPI target values, such as the percent of requests fulfilled within the limits specified by an SLA, might trigger an alert and an escalation procedure.

Considerations need also be made for modeling the scope in which a given service is being leveraged—individual, composite, part of a long-running process, and so on. So, to successfully compose Web services processes, one must fully understand the service's Web Services Description Language (WSDL) contract, along with any additional requirements, capabilities, and preferences. For example, knowing that a service supports a Web services security standard such as WS-Security isn't enough information to enable successful composition. The client needs to know if the service actually requires WS-Security, what kind of security tokens it's capable of processing, and which one it prefers. The client must also determine if the service requires signed messages; if so, it must determine what token type must be used for the digital signatures. Finally, the client must determine when to encrypt the messages, which algorithm to use, and how to exchange a shared key with the service.

Trying to orchestrate with a service without understanding these details will lead to erroneous results. Service operations management addresses such concerns. It's a critical function that can be used to monitor the correctness and overall functionality of orchestrated services, avoiding a severe risk of service errors. In this way, one can avoid typical errors that may occur when individual SLAs aren't properly matched. Proper management and monitoring alleviate this type of risk, since the operations management level lets managers check the correctness, consistency, and adequacy of the mappings between the input and output service operations and aggregate services in a service composition.

Another aim of xSOA's service management layer is to support open service marketplaces. Open service marketplaces operate similarly to vertical marketplaces (such as those for the semiconductor, automotive, travel, and financial services industries) but they're open. Their purpose is to create opportunities for buyers and sellers to meet and conduct business electronically, or aggregate service supply and demand by offering value-added services and grouping buying power (like a cooperative). The scope of such a service marketplace would be limited only by the ability of enterprises to make their offerings visible to other enterprises and establish industry-specific protocols by which to conduct business.

Open service marketplaces typically support supply chain management by providing their members a unified view of products and services, standard terminology, and detailed process descriptions. In addition, service marketplaces must offer a comprehensive range of services supporting industry trade, including services that provide transaction negotiation and facilitation, financial settlement, service certification and quality assurance, rating services, service metrics, such as number of current service requestors, average turnaround time, and manage the negotiation and enforcement of SLAs.

xSOA's service management layer includes market management functionality (as shown in Figure 1) that's aimed to support these marketplace functions. The marketplace is created and maintained by a market maker (a consortium of organizations) that brings the suppliers and vendors together. The market maker assumes the responsibility of marketplace administration and performs maintenance tasks to ensure the administration is open for business, and provides facilities for the design and delivery of an integrated service that meets specific needs and conforms to industry standards.

The xSOA service management functions can benefit from grid computing as it targets manageability. Service grids constitute a key component of the distributed services management as the scope of services expands beyond the boundaries of a single enterprise to encompass a broad range of partners, as is the case in open marketplaces. For this purpose, grid services can be used to provide the functionality of the xSOA's service management layer.

Grid services are stateful services that provide a set of well-defined interfaces and follow specific conventions to facilitate coordinating and managing collections of Web service providers or aggregators. The grid service indicates how a client can interact with it and is defined in WSDL. The state of the service is exposed to its clients as a standard interface that addresses Web service filtering, discovery, routing, aggregation, selection, data and context sharing, notification, and lifetime management.

The principal strengths of Web and grid services are complementary, with Web services focusing on platform-neutral description, discovery and invocation, and grid services focusing on the dynamic discovery and efficient use of distributed computational resources. The compatibility of Web and grid services has given rise to the proposed Open Grid Services Architecture (OGSA), which makes the functionality of grid services available through Web service interfaces.

Grid services are used in the xSOA's service management layer to provide an enabling infrastructure for systems and applications that require the integration and management of services in dynamic virtual marketplaces. Grid services provide the possibility to achieve end-to-end QoS and address critical application and system management concerns. **bij**

References

I. Foster, C. Kesselman, J.M. Nick, S. Tuecke, "Grid Services for Distributed System Integration," *IEEE Computer*, 35(6), 2002.

M. Papazoglou, D. Georgakopoulos, "Service-Oriented Computing," *Communications of the ACM*, Vol. 46, No. 10, October 2003, pp. 25-28.

S. Tuecke, et al., "Grid Service Specification," technical report, Open Grid Service Infrastructure WG, *Global Grid Forum*, 2002. Draft 5, November 5, 2002.

About the Author

Michael P. Papazoglou holds the Computer Science chair and is director of the INFOLAB at the University of Tilburg in the Netherlands. He specializes in distributed computing systems, interoperable database solutions, service-oriented computing, and architectures. He serves on several international bodies and has chaired several major scientific conferences. He has (co-)authored/edited 14 books and approximately 150 scientific journal articles and refereed conference papers. He is a Golden Core Member and a distinguished visitor of the Institute of Electrical & Electronics Engineers (IEEE) Computer Science section.
e-Mail: mikep@uvt.nl