



Fundamentals of Service Orientation

CONTENTS

Aligning Business and IT.....	2
Has IT Become a Commodity?	3
Service Orientation	4
Web Service Technology Stack and Specifications.....	7
Service-Oriented Computing Model	10
Adopting a Service-Oriented Architecture.....	12
The Impact of SOA on Your Organization	13
Solving Modern Problems with Modern Technologies	13
Appendix: SOA-Related Organizations.....	14
About the Author.....	14
About Attachmate.....	15

Fundamentals of Service Orientation

As the high-tech world evolves, IT is confronted with a level of complexity that was unthinkable even a few years ago. Security, centralized management, visibility, integration, and overlapping silos of information are just a few examples of the struggles most IT organizations deal with on a daily basis.

This technical complexity, added to a constant demand for new applications and computing resources, tight schedules, and limited budgets, has placed IT in a new, unenviable position: the bottleneck. In reality, the costs and risks associated with IT projects have increased significantly over the past 10 years while productivity gains have been increasingly questioned.

In response, the software industry is changing the way we consume and assemble computing resources, thanks to connectivity improvements and web services technologies. Although the industry has named this evolution service orientation, there are a lot of questions about the meaning of service orientation and its derivations, such as service-oriented architecture (SOA) or service-oriented computing.

This white paper explores the background, technology, and models behind service orientation. As a technology resource, it can help you make informed decisions about services and their place in your enterprise.

Aligning Business and IT

“I believe our industry has a responsibility, and an opportunity, to dramatically simplify the computing environment by seamlessly weaving together all of the devices, services, and multiple layers of software into a coherent, efficiently managed technology framework,” said Steve Ballmer, CEO of Microsoft. Most enterprises have organized their internal functions to operate as services and compositions of services with a clear interface and a formalized exchange of information. However,

this organization is rarely reflected by the systems supporting the operations that hard-wire services and compositions of services within monolithic information systems. As a result, services need to be duplicated as different systems require identical services to operate, but without the ability to leverage them outside system boundaries.

Consider a sales-tax calculation service. Within your organization, it is likely that many systems require this capability (for quotes, orders, invoices, e.g.), and the management of this service is duplicated each time sales-tax calculation rules change. Wouldn't it be more efficient to centrally manage a sales-tax calculation service that can be invoked by any system that needs it? All updates applied to this service would become transparent, regardless of the system invoking it.

Enterprise architecture

Figure 1 represents a traditional enterprise architecture framework with its different perspectives:

- **Business** – describes the way the organization operates.
- **Information** – captures all the data that support the business.
- **Information systems** – capture the logical organization of applications that manage the information.
- **Technology** – describes the physical organization of information systems.

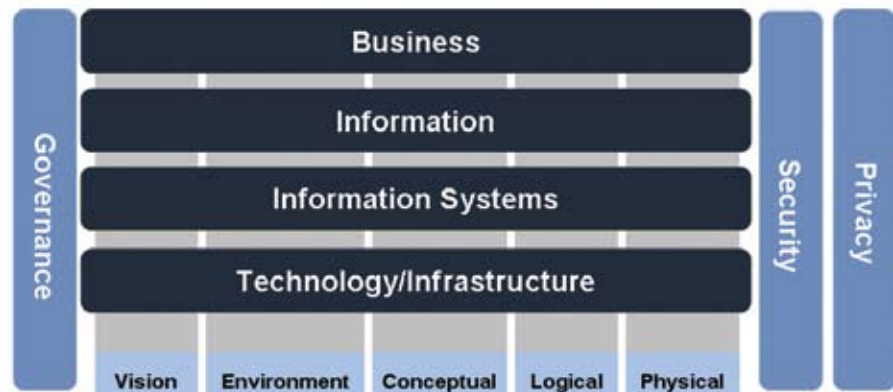


Figure 1: Enterprise Architecture Framework (Adapted from the Institute for Enterprise Architecture Developments)

One of the goals of service orientation is to create a better alignment between information systems and the structure of the enterprise. To this end, service orientation introduces a new layer in the enterprise architecture: the services layer.

Services should be designed to represent an abstraction of the information systems from a business perspective. For example, a customer service could provide a single point of interaction with customer information scattered across multiple systems. This type of service simplifies integration between systems by abstracting the intricacies of several systems of record. For example, an *Update Customer Information* operation of a customer service can be developed as a composition of individual operations, each updating their respective systems of record. Any authorized application can invoke this service without knowing the details of all the systems that must be updated.

These types of services represent packaged integration components that eliminate the need for integration points between new applications and existing systems. Service orientation also makes it more cost effective to expose this type of functionality to partner applications. So, with the appropriate credentials, a partner's retail system could update customer information captured at the point of sale.

A well-designed SOA goes beyond providing a business abstraction of information systems; it provides technology and location abstraction. For example, services can be consumed or coordinated with each other regardless of their implementation technology or their current hosting location. These abstractions address many of today's common IT difficulties while providing new opportunities to serve the extended enterprise, customers, channel, partners, and suppliers.

Service orientation is also about better governance because the messages exchanged by services can be monitored. That means the enterprise can gain real-time visibility.

Has IT Become a Commodity?

Given that businesses have roughly equivalent infrastructures, some argue IT has become a commodity (much like electricity, or machine

tools) that does not offer any specific competitive advantage. But Figure 2 shows a different interpretation, in terms of cost and value for adding new IT systems to an organization.

When the number of information systems is low, it's true that most companies enjoy a positive return on investment as new systems are added. For example, many small and medium enterprises have already had profitable implementations of ERP or CRM systems.

However, larger organizations can encounter steep costs for integrating new systems while having to maintain secure and available sets of information. The challenge is especially daunting for large and diverse organizations in these situations:

- A broad and complex technology landscape with requirements for interoperability between disparate technologies.
- Customers or partners who need to access some of your IT resources.
- Information that is often replicated (either manually or through integration) from one application or group to another.
- Redundant applications, data, software, and hardware resulting from mergers and acquisitions.
- Operations in multiple countries, accommodating the needs of customers in each country.

The good news is that services – aligned with business functions (and not with information systems) – can help reduce the need for integration, evolution, and management in these types of organizations.

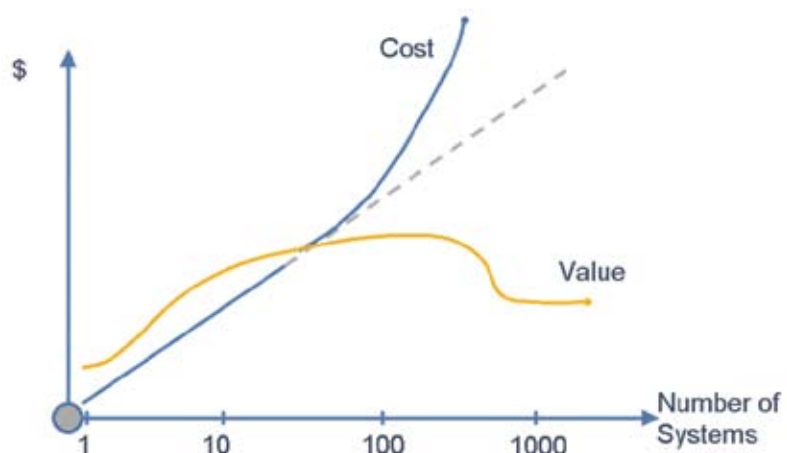


Figure 2: Cost and Value of Adding New Systems

In parallel, it's expected that the value of an additional system could potentially decrease the value of other systems. That's because information workers often use many applications to perform their day-to-day activities. The ongoing need to switch context can increase inconsistencies between similar data inputs, increase training costs, and decrease morale as employees cope.

A compelling reason for the perceived decline in IT's competitive advantage is that most organizations have reached the cross-over point where margins are rarely improved by IT projects. While businesses are constantly finding innovative ways to improve their bottom line, costs and risks – coupled with the complexity of the existing IT landscape – prevent most projects from going forward.

By itself, service orientation might not be enough to return IT to a point of positive ROI. We also need a technology that consolidates user tasks into a coherent interface to maintain a level of productivity, regardless of back-end sources. That's where composite applications come into play.

The web has already opened up new ways to think about applications (such as kiosks or self-service applications) that aggregate functionality from different back-end systems. Composite applications represent a generalization of these concepts on top of a service-oriented architecture that provides the services to access the systems of record.

Service orientation and composite applications represent an answer for organizations that need to:

- Deliver cross-functional systems.
- Exploit existing assets to the greatest degree possible.
- Adapt to changing business conditions or solve new problems.
- Develop new business opportunities by exposing existing assets to customers or partners.
- Reduce the number of unstructured activities that cross departmental boundaries.

Service Orientation

“The real secret sauce of service-oriented architecture is enabling better uses and standardization of business application logic,”

according to Kevin Pollari of *InfoWorld*. The reasons for this will become obvious upon further reading, but let's start with some background.

Origins

Service orientation is not a new idea. For thousands of years, societies have prospered by specializing activities and skills such as the blacksmith, the miller, and the doctor. Generally speaking, a service has a well-defined interface that is optimized for consumption and has an explicit quality of service. It is self-contained and autonomous with no visible dependencies on other services. In fact, its operations are most often hidden. A service can be used when needed, but remains idle until a request arrives. This type of service is readily available with little or no need for integration. New services can be offered by combining existing services, such as the post office's adoption transportation services.

In the past, software and hardware have not been available as services. IT departments are usually cluttered with monolithic technologies, running on dedicated hardware, that do not exhibit any of the characteristics of service orientation. They include different components with redundant, but incompatible implementations that require complex state replication mechanisms. Most often, a simple change in a system has a ripple effect across department and enterprise boundaries. Applications cannot be used as needed with little or no preparation, and nothing can be combined without complex integration projects.

The web, with its ability to provide global access to rich content and computing resources, has already revolutionized many aspects of computing. The web reaches over a billion users, obliterating distances, geographical boundaries, time zones, and technology barriers.

Service orientation started as an experiment in the mid-1990s, with a simple idea: Use web technologies to exchange information packaged as XML documents over HTTP. The goal behind the experiment was to enable servers to communicate with each other by leveraging the same infrastructure users had grown accustomed to. A decade later we can weave computing resources into connected systems that are easier to manage and change, thanks to new concepts, new technologies, and a new application model (composite applications).

Key enablers

Beyond this vision of the massive and seamless interconnection of cooperating computing resources, there is the natural evolution of all human activities, which has transformed what we value most into a utility. Utilities such as energy, water, telephone, mail, and cable are always available, easy to tap, billed on a per-usage basis, and unaware of what customers are doing with them. Why couldn't we do that with computing resources?

What seemed impossible

just a few years ago is now within reach because of advances in bandwidth, system scalability, security, automated connectivity, and information switches.

We can use the simple example of a tracking service. Many business operations involve the movement of goods from one place to another. With technologies such as Radio Frequency Identification, it's possible to scan the location of goods with minimal human intervention. If you are a shipper, these goods could be your customers' parcels. If you are in the retail business, the goods move from inventory to the shelves and then out the door. However, different people are interested in this movement of goods, including employees, suppliers, and customers. They may not want to use a single web site to access tracking information, because user productivity usually declines as the number of applications in play goes up. In a call center, this kind of context switching can introduce errors and reduce response rates.

Service definition

A software service is autonomous and exposes an interface to multiple operations for a single purpose. An operation is invoked through the exchange of messages. Our tracking service example exposes three simple operations: *Post new status*, *Get status*, and *Get status history*. (See Figure 3.)

Ideally, there should be no physical limits to the number of network endpoints that can invoke its operations; otherwise, this could translate into lost

business opportunities. A service is technology-neutral and should support invocations from another application or service, a web browser, a rich client, or an emulator.

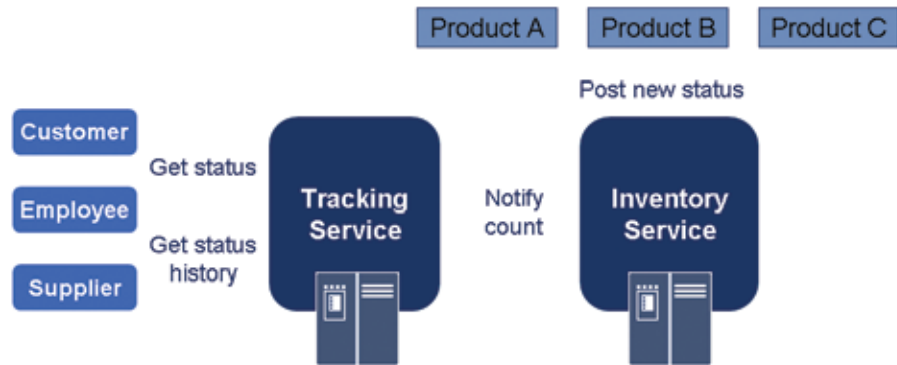


Figure 3: Product-Tracking Service Example

The service does not need to know the context of its use. Potentially, the service could support the operations of a wide variety of customers from a simple retail store to a global carrier, tying together millions of products and customers that need to find each other. The same type of service could be used to track utility usage and send monthly reports to a billing service. This notion of context independence is a fundamental property of a service as well as a key service-orientation design principle.

Triggering events

The tracking service should enable other services to register with it when they need to be notified of a particular event. For example, an inventory service must be notified after a certain number of items have left the warehouse. Events are a type of message exchanged via an operation invocation of a service. Concepts such as publish/subscribe, which are inherent to events, are currently being added in the technology stack by several standards working groups.

Composite services

Services should be designed so they can be assembled to form higher-level services. For instance, a business customer might create a composite service that invokes different tracking services used to ship orders. This service provides a single view into the status of all shipped orders.

Services can also be consumed by a system that itself is not a service. You can create these services from scratch or update existing systems to become service producers. Service orientation enables you to combine services into composite services that invoke other services to perform work, as shown in Figure 4.

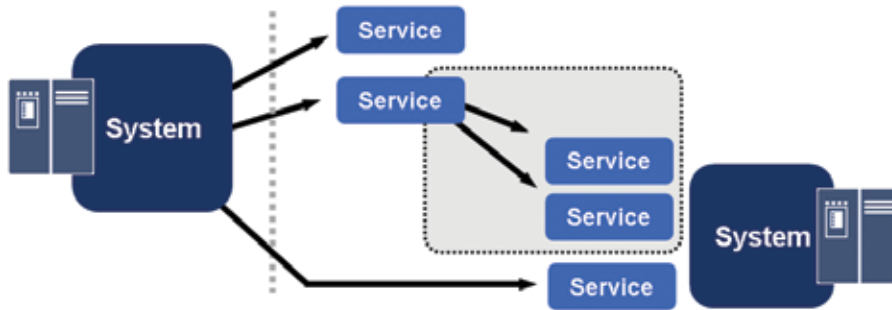


Figure 4: Composite Services

Service classifications

Services are classified using factors that include the cost of searching, selecting a suitable service, and the cost of switching from one service to a similar service, yielding three main categories: Commodity, Standard, and Integrated. (See Figure 5.)

For example, a weather report service might be found and qualified for usage. It might be possible to dynamically search and select such a service for each request because it is critical to get the correct information. In that case, the service does not hold any proprietary data from the service requestor and the lifecycle of the service invocation is limited to the report request. This type of service is poised to become commoditized.

In other cases, the selection process could be long and complex, such as the process for services involving financial information or transactions. However, the cost of switching from these types of services could remain fairly low because they use industry-standard interfaces and were designed not to hold any data from the service consumer.

Standard services represent the first level of mission-critical services and are common to many if not all businesses. They are often transactional in nature, such as a payment service. For

instance, you could set up some of your end-user applications to use a payment service that enables your employees to initiate payments on orders or past due balances. The selection process could be based on quality of service, cost, performance, and accuracy. Switching from one payment service to

another might be as simple as changing the service description file URL because this service does not hold any critical data from the requestor.

Integrated services are usually core to your business functions. They can still be standard enough to be shared by multiple businesses,

but they are involved in long-running activities, cooperating with other services, and supporting the business processes of your organizations. Standard and commodity services are found supporting some of the user activities of these processes not involved directly in changing the state of your business, but rather contributing to it. Integrated services will always manage critical operational data and act as a system of record.

At Attachmate, we consider service orientation the ideal way to organize and provide access to your information systems. Combined with technology advances such as web services, it fundamentally changes the way you consume and assemble computing resources.

	Cost of Searching	Cost of Switching	Types & Examples	Applications
Commodity	Low	Low	Informational <i>Credit Report</i>	Composite Applications
Standard	High	Low	Transactional <i>Update Cust. Information</i>	Composite Applications, EAI
Integrated	High	High	Process Oriented <i>Sourcing</i>	Composite Applications, BPM, B2B, EAI

Figure 5: Service Categories

Service-Oriented Architecture

SOA is a way to organize your information systems so that connected systems and composite applications can be created easily and changed as needed, following the principles of service orientation. In such an architecture, services exchange messages as their primary way to notify each other of events, request information, or demand that an action be done on their behalf. Services within connected systems perform units of work cooperatively.

SOA is poised to change the relationships we have created between systems. It will open up new possibilities with composite applications that leverage services and service orientation to provide a single point of usage into multiple systems of record.

The World Wide Web Consortium (W3C) architecture group has identified that SOA and related technologies are particularly well suited where:

- Systems must be designed and maintained independently of each other.
- Components of the distributed system run on different platforms and vendor products.
- Application data or logic cannot be replicated or run locally.

For example, a Do Not Call List prevents outbound marketing contact centers from calling certain customers. This type of information is impractical to maintain locally because it has a large volume of information that changes on a daily basis. Not having up-to-date information might have drastic consequences for the outbound caller. However, this kind of information can be wrapped in a service that is invoked from any application prior to dialing a phone number. At this point, the application becomes a composite application.

Web Service Technology Stack and Specifications

Connected systems create the need for a fundamental change in the computing model. We used to look for better ways to create code to manipulate data. Structured programming, object-oriented programming, and components were helping developers build complex applications. In a connected system, the information flow, via the exchange of messages between services, becomes the focus of the technologies and architecture.

Unlike preceding messaging technologies that required a centralized infrastructure to manage message exchanges, service orientation provides an infrastructure to establish, manage, and process information flows between services.

Another difference is that service-orientation technologies have been directed by standards groups rather than by infrastructure vendors. The result is that these technologies have been built on a set of open, royalty-free standards that are widely supported by vendors to achieve and promote interoperability. Organizations such as W3C, OASIS, WS-I, and the OMG contribute to the web services technology stack. (See *Appendix* for a list of these organizations.)

Figure 6 categorizes the web services technology stack into three layers: Message, Service, and SOA. The concept of composite applications is not yet supported by specific standards, but could reside in a layer above the Service-Oriented Computing Stack (SOC).

SOC provides the foundation for services to cooperatively perform units of work. The upper layers rely on the lower layers, but lower layers can be used independently. For example, systems can exchange SOAP-formatted messages regardless of whether these systems can be described with a WSDL definition. Similarly, any system can invoke a service without being a service itself. (See explanations for SOAP and WSDL below.)

Figure 6 shows the main technologies of the web services technology stack that are already in place. Here are two recently completed, important specifications:

- WS-Reliable Messaging offers some level of guarantee that a message was received by a service, even on top of non-reliable transports (HTTP). In April 2005 BEA, IBM, Microsoft, and TIBCO Software presented the latest version of this specification, comprising both protocol and policy assertions, to OASIS for further refinement and finalization as a web services standard.
- WS-Transaction with its accompanying specification, WS-Coordination, was published by BEA, IBM, and Microsoft in 2002. The Atomic Transaction part of this specification allows web service operation invocations to participate in a transaction, be it ACID (Atomic, Consistent, Isolated, Durable), long-running, or any other form.

Another set of incomplete specifications relate to business-to-business electronic commerce. Starting in 1999, and now in its third release, the ebXML (e-Business XML) standard is aligning its specifications for e-commerce applications to complement the web services technology stack.

Simple Object Access Protocol

The original meaning of SOAP, Simple Object Access Protocol, was a bit of a misnomer.

SOAP is no longer simple or object oriented, so W3C recommends using only the acronym. SOAP is a lightweight and extensible protocol for exchanging structured information in a decentralized, platform-neutral, distributed environment. SOAP is a connectionless protocol that is transport-independent, although it can be bound to an existing transport, such as HTTP. Unlike most protocols that are point-to-point, SOAP is an end-to-end protocol that supports intermediaries.

A SOAP message is an envelope that contains a header and a body. The body usually contains application-specific data, which can be encrypted. The header contains processing instructions such as a destination address, a reply-to address, message-correlation tokens, or security credentials. SOAP was designed with an extensible header to add different kinds of message-processing instructions and parameters. Many of the specifications (WS-Security) are providing extensions to the protocol as part of their definition.

Web Service Description Language

WSDL is an XML language for describing the message-driven interface of services. These messages can be grouped into operations that follow specific Message Exchange Patterns such

as Request / Response and Notification / Fault. The content of the message body may be specified within the WSDL definition using the XML Schema specification.

WSDL has been designed to separate the abstract definition of an interface from its invocation details such as how and where a given service is offered. Service consumers can use WSDL to introspect all the information necessary to invoke a particular service by sending the appropriate SOAP message to it.

Universal Description, Discovery, and Integration

UDDI focuses on the definition of a service directory via a set of interfaces and operations supporting specific information:

- Description and discovery of businesses, organizations, and other web service providers.
- Web services they make available.
- Technical interfaces used to access those services.

A UDDI-compliant directory operates like a search engine that allows service consumers to find services based on various search criteria. Though this type

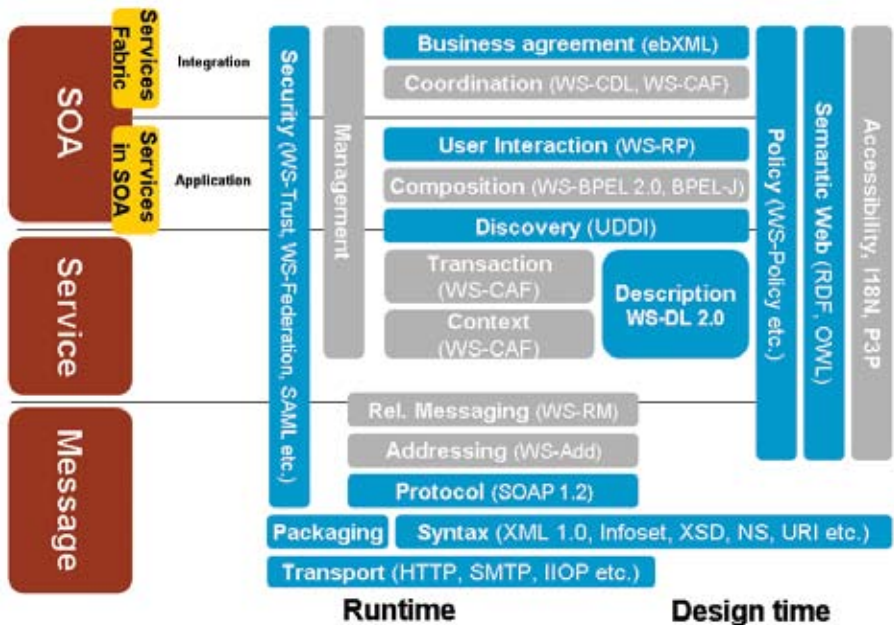


Figure 6: Service-Oriented Computing Stack (Dark boxes indicate published specifications. Light boxes indicate unfinished specifications.)

of registry could be used for services other than web services, the registry contains references to the WSDL files associated with web services.

A UDDI directory has a range of abilities:

- Providing secure access to the information if required.
- Notifying service consumers of any changes to the service usage mode.
- Being used privately within the boundaries of an organization.
- Publishing service definitions to public registries if they are available to all consumers.

Web Service Business Process Execution Language

BPEL is one of the key specifications of the web services stack and offers a variety of capabilities, although its association with business processes is a bit misleading. BPEL cannot model and execute enterprise business processes per se, and it does not have enough semantics to support the definition of such processes. However, WS-BPEL can be used to define an orchestration of service operation invocations; for example, an executable series of operation invocations (in sequence or in parallel). In particular, you can use WS-BPEL to define web service compositions at the operation and interface levels.

WS-BPEL definitions can also be abstract definitions. When associated with a web service interface definition, they represent that behavior; for example, the sequence in which the operation of a particular interface can be invoked.

The Java community is developing BPEL-J, which integrates the Java language and runtime environment with WS-BPEL. Microsoft is working on similar concepts for the .NET platform.

WS-Security

Just as there are secure web sites, there are secure web services, which respond to requests from authorized users and serve encrypted and digitally signed content.

The difficulty around the web services security model is that any incoming message must prove a certain number of claims from the sender. It cannot make any assumption about the transport

such as SSL, which opens a secure connection and lets the two parties interact freely until the connection is closed. Web services establish a secure conversation rather than a connection. The WS-Security standard offers a number of SOAP header extensions to send security tokens as part of a message and to implement message-content integrity and confidentiality.

The core of WS-Security relies on the notion of a security token, which in the physical world can be likened to a driver's license or a national ID card. A token provides a universally verifiable mechanism that associates a series of claims to a subject such as name, age, and address.

In addition to Username security tokens, WS-Security supports signed security tokens (X.509 and Kerberos) that can be validated by an authority and offer revocation mechanisms. X.509 is a recommendation for creating and validating digital certificates. Kerberos is a general authentication and authorization framework based on the notion of tickets.

Many commercial-grade implementations of both technologies are available, as well as implementations of the WS-Security specification using security tokens. Recently, Security Assertion Markup Language (SAML) and Rights Expression Language (REL) tokens were added.

WS-Security is not designed to be the security solution for web services and SOA, but rather building blocks. Other specifications such as SAML, WS-Federation, and ID-WSF from Liberty Alliance are currently being developed to complement the security infrastructure of web services.

WS-Policy

The concept of policy is central to SOA as it provides a framework to programmatically form an agreement between a service requester and a service provider by comparing respective policies. A policy conveys all conditions to interact with a given service such as security, quality of service, cost, response time, or preferred transport.

A WS-Policy may assert that a web service requires WS-Security using Kerberos tokens. This kind of information cannot be expressed in WSDL, so you must use a Policy Attachment to specify a policy for a web service definition.

WS-Remote Portlets

WS-RP defines web service interfaces that allow the plug-and-play of content sources (portlets) with portals and other aggregating applications. WS-RP provides a standard way to consume web services in a portal that everyone can adopt to write content-rich services.

Service-Oriented Computing Model

It's hard to think of an application that can work in complete isolation from other systems, or one that could not benefit from being connected to other systems. When you factor in the value of service orientation, it's not surprising that infrastructure and desktop-application vendors are rethinking their product lines to harness the benefits of producing and consuming services.

In the past, IT focused on increasing productivity, scalability, robustness, security, and user-access technologies for point applications. Today, IT has shifted toward augmenting the value of the whole vs. the parts through integration, federation, and composition. Service orientation enables the creation of autonomous assets that can be composed and coordinated at will to perform complex units of work, building systems that can accommodate change.

Figure 7 shows the layers of the service-oriented computing model to position service-oriented vendors.

Services layer

The services layer is at the bottom of the stack. Just as the World Wide Web created a new breed of businesses, service orientation is creating a new breed of service providers. New pure play vendors are already providing hosted services such as a Do Not Call List service or a service that matches U.S. addresses with phone numbers. Going forward, the majority of services will be created and hosted

internally using existing systems. Companies like Amazon and eBay are racing to service-enable their commerce platforms and let other businesses consume their commerce engines.

Solution vendors are creating a web service interface to their APIs. In the case of host-based systems, services have to be identified and sometimes created by hand. To help standardize this effort, the Open Applications Group (OAGi) has published standard service interfaces for their integration components, including billing, inventory, requisition, order entry, quotes, customers, bills of materials, and project services.

Service-enablement layer

Web services standards have created a market for service-enablement solutions that allow new code or host-based applications to be exposed as web services.

The service-enablement layer is responsible for dealing with the message interchange of service invocations and directing the content of these messages to and from the appropriate process such as a host system, EJB™, or other process. Dedicated technical services such as transformation or orchestration can adapt the service interface to the underlying service implementation.

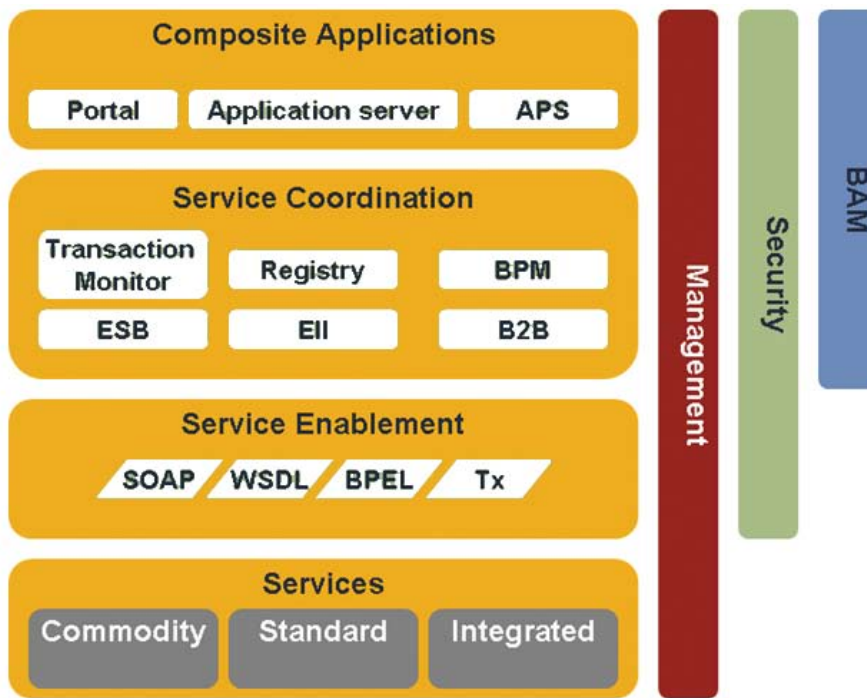


Figure 7: Service-Oriented Computing Model

Service coordination layer

The service coordination layer is sometimes referred to as the service fabric. Just as an application server provides technical services (such as transaction coordination, security, connection pooling, and persistence) to its components, the service fabric provides technical services that are needed to support a unit of work.

This layer provides the following technical services:

- Service registry (UDDI).
- Transaction monitor that maintains transaction context and rollback.
- Message routing engine.
- Federation engine that can dispatch requests to multiple services and aggregate their responses.

Business Process Engines, B2B integration servers, and Enterprise Service Buses provide some of these technical services.

Enterprise Service Buses

An ESB is messaging middleware based on open standards. It supports enterprise application integration via the exchange of SOAP messages, either directly or in the context of web services operations. Like any EAI infrastructure, an ESB provides technical services such as:

- Transformation.
- Content-based routing of documents.
- Reliable message exchanges.
- Security.
- Orchestration.

Business Process Management

BPM refers to a field and a category of products that enables the modeling, automation, execution, and activity reporting of enterprise business processes. Two standards groups (BPML and OASIS) have developed an execution language that is directly tied to web services technologies (BPML and WS-BPEL).

Composite application layer

The composite application layer is the user interface to service orientation. If you think about

service orientation only in the context of simple service invocation or application integration, a service-oriented application model presents some interesting properties:

- A single point of usage independent of the underlying services.
- No physical or technical boundaries (only logical ones).
- Ease of deployment and adaptability to new uses.
- Focus on federation (not integration).
- Scalability and availability

Portal and application servers represent a primitive composite application framework. An application platform suite provides a framework that can support enterprise-wide composite applications and scale with the number of underlying services and systems of record.

Management layer

Management becomes a bigger concern when the number of available services, the size of connected systems, and the need for availability increase. Dependencies between services and their respective versions can become a critical problem as your organization deploys SOA.

Security layer

As a federation of services, connected systems introduce authentication and authorization issues, and require a trust federation to be in place. Similarly, since data travels freely across services outside a connection or the boundary of a process, you must develop a strategy to maintain the privacy of data and the non-repudiation of sending and receiving messages.

Business activity monitoring layer

Because SOA relies on the exchange of XML messages, it is particularly well-suited to increase the visibility of information. Messages can be routed to an information collector that extracts specific data elements from messages and creates reports with them. Business intelligence, BPM, and EAI vendors now provide solutions in this space.

Adopting a Service-Oriented Architecture

The combination of a compelling value proposition and a low barrier to entry portends great success for service orientation. It is not a question of *if* your organization will adopt an SOA, but *when*.

Transforming your IT to a service-oriented architecture is an enterprise-wide project, though with every step, service orientation can help you see a positive return on the effort. To help your enterprise involve everyone in an SOA project, consider these preliminary steps:

- Get a clear mandate from senior management.
- Establish a program office to drive the vision and do the planning.
- Engage all business units and key stakeholders, including users.
- Establish a strategy that focuses on solving recognized enterprise problems, with a clear projection of ROI.
- Be realistic about the value proposition, impact, and schedule.

- **Execution** – By using the knowledge gained from establishing the architecture, applying lessons learned, and understanding the cost benefits to mitigate risk, IT can execute a strategy and be confident in its ability to predict outcomes.
- **Optimization** – Having successfully executed a strategy, the technology is embraced by IT and business units. Benefits are realized, risks are managed, and SOA is part of your problem-solving toolbox. At this point, senior management should be well equipped to respond to change, unconstrained by technology.

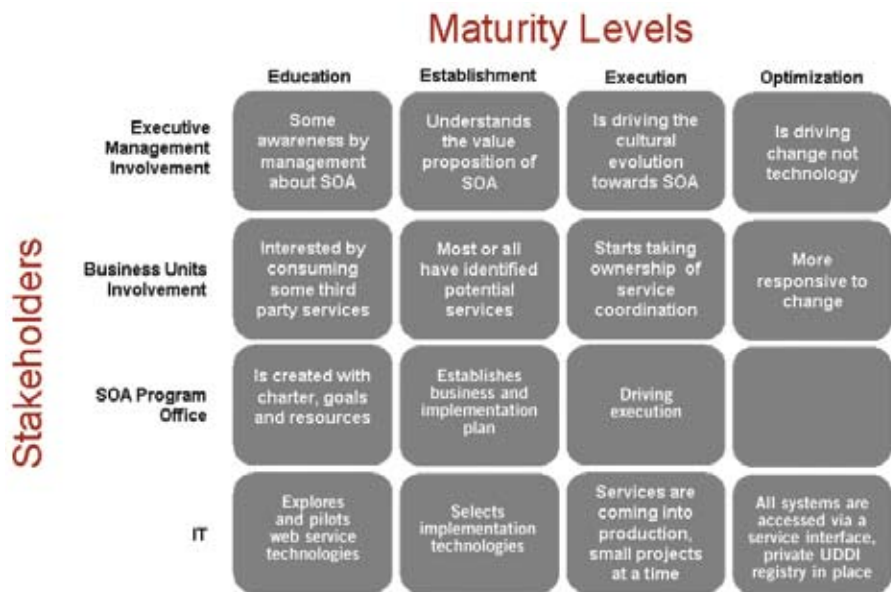


Figure 8: SOA Maturity Model

Figure 8 represents the SOA maturity model. It features the four levels of maturity with respect to the roles and responsibilities in your organization:

- **Education** – IT is learning the foundational principles and technologies of SOA. Learning can include experimental projects ranging from prototypes to small-scale service deployments. Note that ROI won't be recognized if you deploy during this phase; the target is not clearly defined and benefits may not be understood until after the fact.
- **Establishment** – IT collaborates with business units and senior management. Having gathered the necessary knowledge, IT defines and builds the architecture to support current and future service-oriented strategies.

The Impact of SOA on Your Organization

Through better alignment with your business operations, SOA might move some decision making closer to your business analysts and end users. The decisions around service coordination and consumption might even move to a different department in your organization. For example, a particular service might be selected and bound to a composite application by an end user instead of by IT.

SOA promises to share computing resources more efficiently across all organizational units. However, control, design, and provisioning of these resources might not be shared or centralized. Sometimes a decentralized approach works better in terms of another department more effectively doing the work without duplication.

The very nature of composite applications and their ability to federate service consumption at a single point of usage should help streamline your business processes by breaking dependencies on solutions and by quickly responding to business changes. This is a fundamental change for IT organizations accustomed to single-purpose applications that are implemented and maintained over long periods of time.

Solving Modern Problems with Modern Technologies

“Business is complicated,” says Jon Bosak of Sun Microsystems. “Any solution that doesn’t reflect that complexity is not a real solution.” Information systems have grown to a point of extreme complexity, making them difficult to manage, evolve, or replace – while the demand for access and connectivity continues unabated. So, having served as the engine behind productivity gains and new business models, information systems, with their modern complexity, have now become an obstacle to innovation and change. And if web applications have simplified the way users, customers, and partners access data to perform complex tasks in self-service mode, their monolithic architecture has compounded the problem.

SOA and its model of decentralized resources is enabling the development of connected systems, services, and composite applications to reduce the need for integration. SOA can create new opportunities to innovate at the business level with fewer technological constraints. Open standards mitigate the risks of SOA technologies, so you can build your own customized SOA in small, incremental steps, delivering value along the way.

How Attachmate can help

Attachmate is committed to helping you find a successful service-orientation strategy. Attachmate Verastream integration suite can transform all your legacy applications into SOA assets by exposing business processes as web services, XML, JavaBeans, or .NET components. Verastream-generated services can be mixed, matched, and reused selectively to extend legacy functionality to new composite applications or SOA.

Unlike other legacy integration products, the Verastream suite includes solutions for mainframe, web, and desktop integration. No code changes to your legacy applications are required. That means you avoid risk while speeding up SOA implementation, application development, and workflow enhancement.

Appendix

SOA-Related Organizations

IETF – The Internet Engineering Task Force is a large, open, international community concerned with the smooth operation of the Internet and the evolution its architecture.

OASIS – The Organization for the Advancement of Structured Information Standards is a not-for-profit global consortium that drives the development, convergence, and adoption of e-business standards.

OAGi – The Open Applications Group is a consortium of solution and infrastructure vendors focused on publishing common document formats exchanged between integration components for EAI and B2B. OAGi recently published web services abstract interface definitions for all its integration components.

W3C – The World Wide Web Consortium develops interoperable technologies (specifications, guidelines, software, and tools) to lead the web to its full potential. The Web Services group of the W3C is defining the architecture and the core technologies for web services.

WS-I – The Web Services Interoperability Organization is an open industry effort to promote web services interoperability across platforms, applications, and programming languages. The organization brings together a diverse community of leaders that provide guidance, recommended practices, and resources.

OMG – The Object Management Group is an open membership, not-for-profit consortium that produces and maintains computer-industry specifications for interoperable enterprise applications.

About the Author

A recognized industry expert, Jean-Jacques Dubray was instrumental in the design of the OAGi, ebXML Business Process Specification Schema (1.01, 1.1), BPML 0.4, and Web Services Choreography (1.0). He has worked with BPM-related technologies since 1997, pioneering NEC eBusiness Framework and the first generation of Process Engine at NEC Boston Technology Center.

Dubray also led the design of two generations of eXcelon's B2B integration server, the first one with a business process engine dedicated to manage the routing of B2B messages between business partners and business applications. A board member of the Center for XML and Web Services CUNY, Dubray is also co-author of the book *Professional ebXML Foundations* as well as many published articles and conference presentations.

About Attachmate

Attachmate, owned by an investment group led by Francisco Partners, Golden Gate Capital, and Thoma Cressey Equity Partners, enables IT organizations to extend mission-critical services and assures they are managed, secure, and compliant. Attachmate's leading solutions include host connectivity, systems and security management, and PC lifecycle management. Our goal is to empower IT organizations to deliver trusted applications, manage service levels, and ensure compliance by leveraging knowledge, automation, and secured connectivity. For more information, visit www.attachmate.com.



Corporate Headquarters
1500 Dexter Avenue North
Seattle, Washington 98109
TEL 206 217 7500
800 872 2829
FAX 206 217 7515

EMEA Headquarters
The Netherlands
TEL +31 71 368 1100
FAX +31 71 368 1181

Asia Pacific Headquarters
Australia
TEL +61 3 9825 2300
FAX +61 3 9825 2399

Latin America Headquarters
Mexico
TEL +52 55 9178 4970
FAX +52 55 5540 4886

WEB attachmate.com
E-MAIL info@attachmate.com

For regional office information, visit www.attachmate.com.