



Intentional SOA for Real-World SOA Builders

Organizations are deploying Service Oriented Architecture (SOA) to lower the cost of Enterprise Applications and increase business agility. This white paper introduces the basic concepts and standards behind SOA and enumerates the business benefits organizations are seeking from SOA. Intentional SOA is the resulting design methodology which connects business benefits with SOA best practices and principles focused on ensuring that implementations achieve the business benefits of SOA.

► *Infravio, Inc.*
1601 South De Anza Blvd.
Suite 165
Cupertino, CA 95014
Tel: 408.861.3000
Fax: 408.873.3800
www.infravio.com

IFWP_ISOIA_0305 – V1.0 – March 2005

© 2005 Infravio, Inc. All rights reserved. The Infravio name and logo are trademarks of Infravio, Inc. All other trademarks are the property of their respective owners.

Executive Summary.....	5
Section One: Fundamentals of Service Oriented Architecture.....	6
Purpose of Section One.....	6
What is SOA?.....	6
What are the Stages of SOA Web Services Deployment?.....	7
1. <i>Implementation of Individual Web Services.</i>	7
2. <i>SOA Integration of Web Services.</i>	8
3. <i>SOA Integration within the Enterprise.</i>	8
4. <i>On-demand Computing through SOA:</i>	8
When Should I Use Intentional SOA?.....	9
What are the Business Benefits of SOA?.....	9
1. <i>Reduce Integration Expense</i>	9
2. <i>Increase Asset Reuse</i>	10
3. <i>Increase Business Agility</i>	10
4. <i>Reduce Business Risk</i>	10
What are the Technology Standards Used in SOA?	10
<i>SOA and Web Services</i>	10
<i>UDDI, WSDL and SOAP</i>	11
<i>WS-I Basic Profile</i>	11
<i>UDDI (Universal Description, Discovery and Integration)</i>	11
<i>The Web Services Registry</i>	13
<i>WSDL (Web Service Definition Language)</i>	13
<i>SOAP (SOAP is just SOAP)</i>	14
Enterprise Service Bus vs Web Services Fabric.....	15
<i>Extending SOAP with WS-* (Web Services Standards)</i>	16
Section Two: Introducing the Intentional SOA (iSOA) Methodology.....	18
Purpose of Section Two.....	18
Intentional SOA (iSOA).....	19
Intentional SOA Readiness Questions	19

Capturing the Four Business Value Points of SOA.....	20
1. Reducing Integration Expense.....	20
Use Ubiquitous Standards Whenever Possible.....	20
<i>Extending SOAP with Web Services Standards (WS-*)</i>	21
Intentional SOA Recommended Standards.....	21
Intentional SOA Candidate Standards.....	22
Intentional SOA Non-Standard Requirements.....	23
<i>Extending WSDL with Contracts</i>	24
<i>Extending UDDI with Identities, Contracts and Other SOA entities</i>	24
Intentional SOA Integration Cost Recommendations.....	25
2. Increase Asset Reuse	25
<i>Non-Web Services Interfaces in Your SOA System of Record</i>	26
Intentional SOA Reuse Recommendations.....	27
3. Support Business Agility with SOA Metadata	27
A Word About Metadata.....	27
The Ability to See Changes in the Business: Enterprise Systems Management.....	28
The Ability to Rapidly Respond: Processes and Declarative Programming?.....	28
<i>Business Processes Programming</i>	28
<i>Declarative Programming</i>	29
SOA Metadata Enables Declarative Programming.....	29
<i>SOA Metadata Platform Features</i>	30
Intentional SOA Business Agility Recommendations.....	31
4. Reduce Business Risk with Runtime Policy Management.....	31
Intentional SOA Business Risk Recommendations.....	33
Intentional SOA Recommended Next Steps.....	33
About the Author.....	34
Appendix A: A Peek into an SOA Repository.....	34
Appendix B: Infracrio X-Registry Metadata Platform.....	36
X-Registry Metadata Entities.....	36
<i>Service Consumer Identities</i>	36
<i>Contracts</i>	36

<i>Processes</i>	37
<i>Dependencies</i>	37
<i>Access Rights</i>	37
<i>Flexible Metadata Attributes</i>	37
X-Registry Functional Model	38
<i>Registry Applications(SOA Management)</i>	38
<i>Governance</i>	38
<i>Contracts (Service Configuration)</i>	39
<i>Repository</i>	39
<i>Verticalization</i>	39

Executive Summary

With the emergence of robust Web Services standards, expensive and customized point-to-point integration has become a thing of the past. Web Service enabled applications are becoming ubiquitous whether provided by ERP vendors or in house application builders. Organizations are increasingly turning to Service Oriented Architecture (SOA) in order to reduce integration expense, increase asset reuse, increase business agility, and reduce business risk. Some organizations are looking to SOA to standardize runtime policy management for compliance and corporate governance initiatives. This paper describes some of these architectural insights and key standards used in building out an SOA, including SOA design-time and runtime components.

The design of Web Services enabled applications is distinct from the design of an SOA. While Web Services development is iterative and organizations can enable their applications one at a time, organizations need to deliberately design their SOA from an architectural perspective, in order to reach the broader benefits of SOA. Ad hoc collections of services strung together as a set of point solutions go back to the same problems SOA was created to address—they create expensive, customized, IT-intensive point solutions with reduced likelihood for reuse and no coherent platform from which to build composite applications. An important step to developing a coherent architecture is the management of metadata which is used for improving the discovery, management and configurability of Web Service applications.

The process of SOA design centered around business requirements is referred to here as *intentional SOA* or iSOA. By focusing on, documenting and measuring business value, organizations can ensure that they will reap the promised advantages of SOA of reuse, agility, and runtime governance.

Section One: Fundamentals of Service Oriented Architecture

Purpose of Section One

This section of the white paper is intended to answer the following questions

What is SOA?

What are the Stages of SOA Web Services Deployment?

When should I use Intentional SOA?

What are the Business Benefits of SOA?

What are the Technology Standards used in SOA?

This section of the paper is targeted at an introductory level. It also introduces a set of benefits of organizational adoption of SOA.

Readers familiar with these concepts may choose to skim or skip this section and advance directly to Section Two, the introduction of *Intentional SOA*.

What is SOA?

A concise functional definition is provided below :

“SOA is the architectural style that supports loosely coupled services to enable business flexibility in an interoperable, technology-agnostic manner. SOA consists of a composite set of business-aligned services that support a flexible and dynamically re-configurable end-to-end business processes realization using interface-based service descriptions.”

--Arsanjani, Borges and Holley¹

A number of best practices have evolved around SOA. This includes separating the software interfaces from the implementation. The idea of interface-based service descriptions dates back to CORBA Interface Definition Language (IDL) and even older systems. These remote interfaces and standards-based communication protocol form a communication “Bus,” a unified channel for integration. By having each service connect to a standards-based “bus”, the number of integrations decreases dramatically over doing large numbers of point-to-point projects.

¹ <http://www.webservices.org/index.php/ws/content/view/full/44989>

The goal of loose coupling is a style of integration where service endpoints assume that arbitrary consumers will use their interfaces. In addition, loose coupling is supported by the opacity of services; consumers should have no knowledge of the implementation details of the underlying service. The addition of a standards based communication style enables a much more flexible style of integration where IT infrastructure can be interconnected on an as-needed basis. Instead of a model where every single point-to-point integration must be hand coded in an unmaintainable and proprietary fashion, loose coupling consists of many generic software components all offering their services to any and all takers. In addition, metadata such as deployment parameters and contract (which will be described later as an Intentional SOA best practice) can increase the ability for a service to be dynamically discovered and the flexibility that it can be bound to arbitrary endpoints. This can make loose coupling even more flexible and dynamic.

Within SOA three fundamental entities emerge: service providers, service consumers, and intermediaries (often referred to as brokers). This approach is a shift from Client-Server thinking—primarily because in SOA, servers can consume services provided by other servers. Clearly, the language of client/server breaks down when a server becomes a “Client” for another server; SOA provides more flexibility and a powerful pattern for reusable software services. The idea of services and SOA is a sufficiently old architectural concept that some ironically refer to it as “Same Old Architecture.” Although the ideas in SOA are not new, the richness and maturity of Web Service Standards and middleware infrastructure (particularly J2EE and .NET Application Platform Middleware) enable this architecture to provide powerful technical and business benefits.

SOA is an evolutionary (not revolutionary) architecture which merges the vision of distributed computing with the practicality of enterprise application integration. It interconnects heterogeneous systems from mainframes to .NET to J2EE. It reduces the cost of integrating software and enables composite application development. A successful SOA project captures the benefits that will be described in this first half of the white paper.

What are the Stages of SOA Web Services Deployment?

According to an article entitled “SOA Is Coming Fast. Are You Ready?” on an IBM developer Web site provided by DevX², the incremental progression towards SOA often entails four stages:

1. Implementation of Individual Web Services.

Frequently, this starting point involves wrapping a Web Service interface around an existing application. Java applications and Microsoft .NET applications are

² <http://www.devx.com/ibm/Article/26685>

particularly amenable to this wrapping because both platforms have built-in support for Web Services. In this regard, it should be noted that Web Services are an excellent mechanism for creating interfaces to applications residing on back-end servers and mainframes.

2. SOA Integration of Web Services.

This stage involves the integration of Web Services within a single department. It is at this stage that the benefits of open standards that provide a common interface between applications and among systems become evident—especially because once a department moves to SOA, adding new services to the department's function becomes a simple process.

3. SOA Integration within the Enterprise.

In this stage, departments use Web Services to communicate between themselves and, at times, with various suppliers. The loan-broker scenario described earlier would represent this stage. It now becomes clearer how SOA enables a dynamic enterprise that can reconfigure business processes quickly.

4. On-demand Computing through SOA:

Business processes within the enterprise and in B2B contexts run on Web Services. Changing the computing infrastructure to match changes in business processes and to external events (such as glitches in the supply chain, for example) becomes a much more straightforward operation. The IT organization is agile and can provide not only configurability but scalability due to the modular design of its infrastructure.

It is appropriate to identify the stage of your organization with respect to SOA, as the recommendations will vary depending on the developmental stage.

Organizational Web Services projects have been in stage one for several years now. Enablement projects use familiar tools such as Microsoft Visual Studio or Eclipse, and shared code repositories such as CVS for source code control. A number of Integrated Development Environment (IDE) plug-ins exist for taking pre-existing applications and creating a Web Services interfaces to that application. This is referred to as *Web Services Enablement*. This stage of development can be done one application at a time, and on an ad hoc basis.

Organizations are increasingly reaching stage two, where they have dozens or even hundreds of Web Services in deployment. In some organizations, stages two and three happen very close together or even simultaneously. In some cases, stage three includes connections with external suppliers in a business-to-business (B2B) paradigm. In this stage a new set of SOA management tools are used to handle metadata requirements and move to an architectural level of abstraction.

When Should I Use Intentional SOA?

Intentional SOA³ is a design methodology for linking SOA implementation strategy to business value

As such, these ideas can and should be deployed at all stages of Web Services development. However, as an implementation methodology it is of greatest benefit between the first and second stages as enumerated above. Once a significant number of Web Services projects have been enabled and brought online, it creates an opportunity to plan for the larger scale architecture and how it creates business value points that are enumerated below:

What are the Business Benefits of SOA?

What are businesses hoping to gain by adopting SOA? Ron Schmelzer of the leading SOA analyst firm ZapThink touts the four main benefits of SOA as reducing integration expense, increasing asset reuse, increasing business agility, and reduction of business risk.⁴ These are goals frequently held in common among companies that adopt SOA. These benefits are discussed in more detail below.

1. Reduce Integration Expense

By integrating at a higher level of abstraction, integration is much faster and cheaper. In addition, by standardizing the way applications talk to one another, you can greatly reduce the cost of integration. Instead of building hundreds of point-to-point custom integrations which can be expensive and hard to maintain, SOA provides a standards-based open communication system for inter-application integration. Boston-based Forrester Research analysts Ken Vollmer and Mike Gilpin report that, according to early case studies, using SOA can reduce integration, project development, and maintenance

³The term “Intentional” in this context is intended to convey purposefulness or goal-orientation. It is not related in any way to the concept of Intentional Software or the company <http://www.intentionalsoftware.com> both created by Charles Simonyi

⁴ The ROI of SOA,” [Ronald Schmelzer](#), Document ID: ZAPFLASH-20050127, ZapThink, January 2005. <http://www.zapthink.com/report.html?id=ZAPFLASH-20050127>

costs by 30% or more⁵. Gartner estimates an implementation cost decline to roughly double the software cost.

2. Increase Asset Reuse

By cataloging and identifying applications that pre-exist as well as ones being built, SOA can reduce the duplication of efforts and eliminate redundant services. By decommissioning outdated services, significant savings in maintenance and support can be realized; by increasing visibility across services being built, redundant efforts can be coordinated and solutions can be deployed more efficiently.

3. Increase Business Agility

Once a number of services are available in an organization, applications can be developed rapidly out of pre-existing building blocks. This is referred to by Gartner as *Composite Application Development*. Rapidly composing applications from existing services speeds up the time to market and reduces the cost and complexity of application development. This also reduces the dependency on overloaded IT resources as applications can be composed by business units and less technical staff members.

4. Reduce Business Risk

Runtime policy management can enable distributed and heterogeneous applications to run under a single set of rules. Runtime policies enforce business requirements, implementation architecture requirements, regulatory compliance, alignment between business and IT groups, service creation, adoption policies, and metrics. The flexibility of Internet-style distributed development with the power of mainframe-style central policy management is an idea balance of business agility and business control.

What are the Technology Standards Used in SOA?

SOA is a standards-based architecture. Without standards, SOA builders would be unable to reap many of the benefits of the architecture. SOA is often implemented using a set of standards which will be described below. A key recommendation for Intentional SOA is to determine the definitive set of standards with which to build your SOA.

SOA and Web Services

SOA, while technically not dependent on Web Services, greatly benefits from the availability of these standards. With Web Services, SOA implementations gain

⁵ http://www.technologydecisions.com/issue/april05_feature_simple.asp

interoperability between heterogeneous systems such as .NET, J2EE and even mainframe-based Web Services. In addition, Web Services enable powerful intermediation features. Because XML (SOAP) messages are machine readable, intermediaries can transform, reroute, distribute, allocate, secure and provide a host of other features based on information in the message as the messages travel through brokers on their way from service providers to consumers. This level of message transparency can also be important for compliance and policy management applications. Because of this interoperability, transparency and intermediation, Intentional SOA best practices recommends Web Services standards as fundamental building blocks of SOA.

UDDI, WSDL and SOAP

The three most significant standards for Web Service applications are UDDI, WSDL, and SOAP (the acronyms will be explained in a moment). These three components are the substantive portions of the WS-I Basic Profile. UDDI helps applications *locate and discover* service interfaces, WSDL *defines* the structure of service interfaces, and SOAP enables structured *communication* between components of a Web Service.

WS-I Basic Profile

The WS-I, or Web Services Interoperability organization is a membership organization chartered to promote Web Services interoperability across platforms, operating systems and programming languages. It is not a standards body, but it incorporates standards from the IETF, W3C, OASIS and many other well-known standards bodies.

The WS-I Basic Profile defines an interoperability mechanism for the core standards used in Web Services implementations. As such, it represents an agreed-upon set of building blocks for deploying Web Services, as well as mechanisms for ensuring that multiple vendor implementations of those standards can successfully interoperate. The WS-I Basic Profile includes SOAP, WSDL and UDDI, as well as some lower level standards and protocols.

There are some small differences between Basic Profile version 1.0 and version 1.1 including the use of MTOM instead of SOAP with Attachments (SwA). These will be address in greater detail later in this white paper.

UDDI (Universal Description, Discovery and Integration)

The first thing any consumer of Web Services needs to do is to find Web Services to consume. Much like your local telephone directory provides a way to look up phone numbers, UDDI provides a standard for discovering available Web Services. UDDI is a

standard supported by the OASIS organization and is considered a standard part of the Web Services architecture.

UDDI is a “meta service” (a service from which you obtain services) for locating Web Services by enabling robust queries against rich metadata. It was initially proposed as a public location for service providers to register services and thus to enable universal description and discovery of services. The lookup paradigm for UDDI is typically a design-time lookup service, and is usually used by application developers or integration developers to search for and find the interfaces they need.

What is the structure of discovery metadata in UDDI? From the UDDI v3 Specification from OASIS⁶, a UDDI information model is composed of instances of the following entity types:

- **businessEntity**: Describes a business or other organization that typically provides Web Services.
- **businessService**: Describes a collection of related Web Services offered by an organization described by a businessEntity.
- **bindingTemplate**: Describes the technical information necessary to use a particular Web Service.
- **tModel**: Describes a "technical model" representing a reusable concept, such as a Web service type, a protocol used by Web Services, or a category system.
- **publisherAssertion**: Describes, in the view of one businessEntity, the relationship that the businessEntity has with another businessEntity.
- **subscription**: Describes a standing request to keep track of changes to the entities described by the subscription.

The metadata description capability of tModels enables enhanced discovery of Web Services. UDDI also provides support for taxonomic discovery (navigating a tree of services). UDDI describes an interoperability standard for a complete registry for Web Services, which can catalog all of the Web Services available in a community either public or private.

⁶ <http://uddi.org/pubs/uddi-v3.0.2-20041019.htm>

The Web Services Registry

Anne Thomas Manes of the Burton Group has helped to define the role of the Web Services Registry in the diagram below⁷ The Web Services Registry interoperability standard is UDDI. As is evident in the diagram, the registry is at the core of many SOA functions beyond design time discovery of Web Service interfaces. Web Services Registries can also support applications such as service lifecycle management and runtime contract enforcement, and can play a crucial role in corporate compliance, IT governance, and a wide variety of applications.

Web Services Registries help developers locate assets, choose among them, and understand the various tradeoffs involved in their consumption. A Web Services Registry creates a more coherent view of a composite application programming interface which spans the entire company.

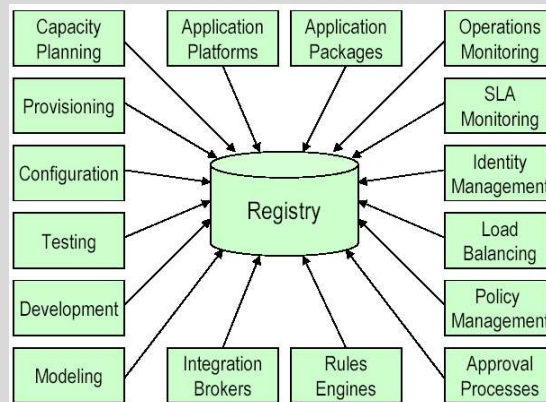


Figure 1. Role of Registry in Enterprise Applications

As can be seen from the diagram, the registry is a central part of many service oriented enterprise applications.

WSDL (Web Service Definition Language)

Central to Web Services is the interface definition. WSDL is the interface definition, and one of the fundamental tenets of SOA is the separation of implementation from interface. Although WSDL stands for “Web Services Description Language,” it has bindings to a diverse number of different underlying service implementation languages and formats including non-Web Services implementations.

⁷ “The Role of Web Services Registries in Service Oriented Architectures,” Anne Thomas Manes, Burton Group, November 2, 2004.

Once a consumer of a Web Service discovers an appropriate service, how do they use this service? The consumer needs an *interface* to the service. An interface provides a mechanism through which a consumer can communicate with the service. WSDL is a description language for describing interfaces to Web Services. When a consumer looks up a service in a UDDI registry, they obtain a WSDL interface to that service.

A WSDL file specifies the location of the service and the operations (or methods) the service exposes. It provides a simple way for service providers to describe the basic format of requests to their systems regardless of the underlying protocol (such as SOAP or XML) or encoding (such as MIME). WSDL is a key part of the effort of the UDDI initiative to provide directories and descriptions of such on-line services for electronic business.

SOAP (SOAP is just SOAP)

SOAP used to be an acronym for “Simple Object Access Protocol”. In version 1.2 of the protocol specification, SOAP is no longer an acronym for anything. Critics suggested that SOAP was neither simple, nor had anything to do with accessing objects. Thus SOAP became “just SOAP”.

Once a service consumer has obtained a WSDL interface to the service, the consumer can initiate a conversation with the service. Typically in Web Services the conversations that pass through a WSDL interface are XML messages. The most common form of Web Services messages used in SOA are SOAP message.

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. SOAP lacks a number of features which are addressed by other Web Services standards. A key example is that SOAP lacks security unless it is enhanced with WS-Security which is a SOAP extension that addresses security issues such as message integrity (guarantee that a message is not tampered with), message confidentiality (guarantee that the content of a message is kept secret), and sender authentication (identify the message sender).

Enterprise Service Bus vs Web Services Fabric

The Wikipedia defines an Enterprise Service Bus⁸ (ESB) as

“an emerging standard for integrating enterprise applications in an implementation-independent fashion, at a coarse-grained service level (leveraging the principles of service-oriented architecture) via an event-driven and XML-based messaging engine (the bus).”

It enumerates the key features of an ESB, including:

- implementation-independent
- XML-based
- utilizes web services
- wrapping of legacy applications and adaptation into the bus
- support for asynchronous processing
- intelligent, content-based routing
- data format conversion

A prime function of the ESB is reliable and scalable asynchronous message delivery and transformation in an SOA.

Phil Wainewright in his popular blog “Loosely Coupled” suggest that ESB will cease to exist⁹, because of the emergence of Web Services standards which make proprietary transport obsolete. An opposing view is provided by David Chappell of Sonic Systems, who suggests that ESB does not rely entirely on proprietary message delivery, in his piece “ESB Myth Busters: 10 Enterprise Service Bus Myths Debunked¹⁰.”

A more moderate view is espoused in Network Computing¹¹, which suggests that an ESB is a generic name for a diversity of Web Services runtime environments which will all coexist with the SOA registry for design time and discovery of services. In this view, ESBs are needed for asynchronous message-based SOA design patterns but are only a part of a larger SOA infrastructure.

From the Intentional SOA perspective, ESB answers questions about transport and transformation of messages, but does not directly address issues such as the web services registry.

⁸http://en.wikipedia.org/wiki/Enterprise_service_bus

⁹ <http://www.looselycoupled.com/blog/lc00aa00073.html>

¹⁰ <http://webservices.sys-con.com/read/48035.htm?CFID=32876&CFTOKEN=F24520B0-E4A4-79B1-02D1100F4ED4033C>

¹¹ <http://www.networkmagazine.com/shared/article/showArticle.jhtml?articleId=23902472&pgno=2>

Extending SOAP with WS-* (Web Services Standards)

Web Services have a number of standards, collectively referred to as WS-* (pronounced *WS Splat*). These standards are primarily developed at the Internet Engineering Task Force (IETF), the World Wide Web Consortium (W3C) and The Organization for the Advancement of Structured Information Standards (OASIS). The following list is derived from several lists including a list maintained by CDBI¹² the BEA list¹³ and the Microsoft¹⁴ list. This list is not intended to be comprehensive, however, in cases where several standards exist (such as WS-Reliability and WS-ReliableMessage) both standards are listed here. More detailed descriptions of these standards are available on the Internet.

Messaging Specifications

SOAP

ASAP (Asynchronous Web Services)

WS-Addressing (superseded by MTOM)

MTOM (Attachments)

WS-Enumeration (specifying lists)

WS-Eventing

WS-Transfer

SOAP-over-UDP

WS-Choreography

Security Specifications

WS-Security : SOAP Message Security

WS-Security : UsernameToken Profile

WS-Security : X.509 Certificate Token Profile

WS-Security : Kerberos Binding

WS-SecureConversation

WS-SecurityPolicy

WS-Encryption

WS-Signature

WS-Trust

WS-Federation

SAML (Identity and Authentication)

Reliable Messaging Specifications

WS-Reliability

WS-ReliableMessaging

WS-Acknowledgment

Transaction Specifications

¹² <http://roadmap.cbdiforum.com/reports/protocols/summary.php>

¹³ <http://dev2dev.bea.com/webservices/standards.csp>

¹⁴ <http://msdn.microsoft.com/webservices/understanding/specs/default.aspx>

WS-Coordination
WS-AtomicTransaction
WS-BusinessActivity
WS-TransactionManagement

Interfaces

WSDL
WS-Remote Portlet

Discovery

UDDI
ebXML Registry
WS-Discovery
WSIL (WS Inspection Language)
WS-Notification

Metadata Specifications

WS-Policy
WS-PolicyAssertions
WS-PolicyAttachment
WS-MetadataExchange
WS-MessageData

XML Specifications

XML
Namespaces in XML
XML Information Set

Management Specifications

WS-Management
WS-Management Catalog
WS-Manegability
WSDM (Web Services Distributed Management)
WS-Provisioning

Business Process Specifications

BPEL4WS

Specification Profiles

Devices Profile
WS-I Basic Profile

This extensive list is presented here for your reference. The approach to selecting standards used by Intentional SOA will be discussed in the next section.

Representational State Transfer (REST)

REST is not a standard, but a way to build Web Services applications. Critics of SOAP suggest that SOAP is unusable without significant tool support. The REST concept was invented by Roy Fielding in his doctoral dissertation.¹⁵ While not a Web Services standard, developers are interested in REST. Like SOAP, REST provides a mechanism for communicating between service providers and consumers. All REST developers need is a browser and a Web programming environment—in some cases a scripting language such as Perl, PHP or Python. REST is a lightweight way of providing and consuming Web Services. The requester in the REST paradigm sends a URL in which all of the requesting parameters are encoded. The response is an XML document which corresponds to this request. This asymmetry initially appears to be client-server and browser-oriented, but there is nothing preventing a server from using URL encoding to request a message from another server.

The REST paradigm is a robust architectural philosophy rather than a specific standard and can be applied partially or completely. For example, traditional REST suggests the use of HTTP GET as transport, when in fact REST paradigm URLs and URIs can be responded to over other transport forms if needed (for example, over guaranteed delivery middleware). REST is much more loosely defined and opportunistic than a SOAP-based approach. It does not rely on heavy tooling, because the calling format is the URL, the debugging tool is the browser, and the parsers are often scripting languages such as Perl or PHP. oriented.

REST, while associated with Web Services is not strongly associated with SOA.

Section Two: Introducing the *Intentional SOA (iSOA)* Methodology

Purpose of Section Two

This section of the white paper introduces Intentional SOA (iSOA), a design methodology for builders of Service Oriented Architectures. This section of the paper is targeted at an intermediate technical level; it is designed to be understood by both business and technical users.

¹⁵ <http://www.ebuilt.com/fielding/pubs/dissertation/top.htm>

Intentional SOA (iSOA)

Intentional SOA is a process for ensuring that SOA projects capture the business value of SOA. As articulated in section one of this white paper, the four principal business values for SOA are the following.

1. Reduce Integration Expense
2. Increase Asset Reuse
3. Increase Business Agility
4. Reduce Business Risk

Each of the four business value objectives is supported by a best practices and patterns-based approach within Intentional SOA. One of the primary axioms held in Intentional SOA is that much of the business value of SOA cannot be achieved without deliberate (intentional) planning, and a coherent architectural view of the software services.

Intentional SOA Readiness Questions

Intentional SOA methodology begins with ten questions that help address organizational readiness for SOA. These questions will assist in the development of a plan for capturing the four business value points of SOA:

Reduce Integration Expense

1. How will you select the standards for use in your enterprise?
2. How will you test to ensure that deployed services are compliant with those standards?

Increasing Asset Reuse

3. How many Web Services do you have in your company?
4. How will users discover available services and ensure reuse?
5. What IT processes will you need for versions, dependencies and change management of services?

Increase Business Agility

6. How do you manage provisioning and service level agreements for internal and external users?
7. How quickly can you integrate and on-ramp new customers to your IT systems?

Reduction of Business Risk

8. What types of users and usage will be permitted and who will approve normal usage?
9. How do we track and enforce compliance with enterprise and regulatory standards?

10. Who will approve the publishing on a service internally? Externally?

Answering these questions provides a starting point for SOA builders to begin to scope out the requirements for appropriate systems for managing and deploying SOA as a true architecture as opposed to a random collection of services.

These questions are exploratory in nature, and may require detailed documentation to address. These questions may produce additional questions and a detailed set of requirements for enterprise-wide SOA deployments. The implications of these questions and a detailed guide to their use will be covered in a future white paper.

Capturing the Four Business Value Points of SOA

In addition to the ten Intentional SOA readiness questions, Intentional SOA has a series of recommendations and considerations for capturing each of the four business value points of SOA. The following four sections will address each of these value points in turn.

1. Reducing Integration Expense

The Intentional SOA approach to reducing integration expense is standardization as well as developing integrations on a higher level of abstraction and productivity. By ensuring interoperability and standardization across all deployed services, the cost of custom integrations and complexity of integration can be mitigated.

One caveat to standardization, however, is that the basic standards that support SOA are not sufficient as a platform to support the other three business value objectives of Increased Asset Reuse, Increased Business Agility and Reduced Business Risk. The following section will address ways to comply with standards, but also to know where the gaps are and how to proceed.

Use Ubiquitous Standards Whenever Possible

The elements of the WS-I Basic Profile are essential technology ingredients for building an interoperable Web Services based SOA. Most practitioners agree that SOAP, or at least XML messaging is a very useful if not essential interoperability strategy between heterogeneous components of an SOA. Also generally accepted is the WSDL standard, which can even be used to describe non-Web Service interfaces such as CORBA bindings for WSDL.

Extending beyond standards can be tricky, as it creates the potential for vendor lock-in and unmaintainable proprietary silos. Unfortunately SOAP, WSDL and UDDI by

themselves do not enable organizations to capture the four business value points of SOA. SOAP by itself does not provide for security, provisioning, access control, identity, management, and a variety of other real world concerns. As SOA carries more business value, the core Web Services protocols of SOAP, WSDL and UDDI require cautious extension.

Intentional SOA distinguishes between “Standards” and “Ubiquitous Standards” based on the old adage “the wonderful thing about standards is that there are so many of them.” Due to the large number of standards and candidate standards that start with the letters WS, Intentional SOA suggests a cautious approach when adopting them for Enterprise-wide deployment.

Extending SOAP with Web Services Standards (WS-*)

An increasingly large number of standards are emerging for Web Services protocols. These standards enable interoperability between heterogeneous systems, particularly different middleware systems, but also runtime systems including Application Servers, Web Services brokers, SOAP processing engines, Enterprise Service Bus (ESB) solutions and integration hubs. These standards are collectively referred to as WS-* and vary to the degree of acceptance.

As you saw in the list provided in section one, there are a bewildering array of standards and proposed standards for extending SOAP and Web Services. For organizations seeking to lower the cost of integration, it is important to be selective when adopting a standard. CBDI maintains a roadmap¹⁶ of standards which clarifies the status of the vast number of WS specifications that are under development.

Intentional SOA Recommended Standards

The following list outlines Intentional SOA recommended core extensions. While a number of promising standards are on the horizon, this subset reflects the most mature and capable extensions to the basic profile supported by WS-I.

¹⁶ <http://roadmap.cbdiforum.com/reports/protocols/index.php>

SOAP MTOM (Message Transmission Optimization Mechanism)

MTOM¹⁷ describes a mechanism for optimizing the transmission and/or wire format of a SOAP message by selectively re-encoding portions of the message while still presenting an XML Infoset to the SOAP application. MTOM also describes an Inclusion Mechanism that operates in a binding-independent way, plus a specific binding for HTTP. Loosely Coupled Thinking Website provides more details on MTOM¹⁸. MTOM is very important strategically because it enables composition of attachments with WS-Security, which SOAP with Attachments does not.

WS-Remote Portlets

WSRP provides a common mechanism for portals to interchange presentation information. defines how to plug remote Web Services into the pages of online portals and other user-facing applications. This allows portal or application owners to easily embed a Web Service from a third party into a section of a portal page (a 'portlet'). The portlet then displays interactive content and services that are dynamically updated from the provider's own servers. Formerly known as Web Services for Remote Portals, WSRP is closely allied with WSIA (Web Services Interactive Applications). A separate Java standard for portlets is known as JSR 168.

WS-Security

WS-Security supports multiple authentication and authorization needs including (as would be expected) Kerberos authentication, role-based security (security tokens), digital signing and message encryption to verify the message has not been altered. This standard is strongly supported because of its usefulness. In order to achieve the broad spectrum of security needed for most Enterprise-wide deployments, other security standards may be needed. Exercise caution when extending Web Services security beyond WS-I recommendations.

Intentional SOA Candidate Standards

The following list outlines Intentional SOA candidate core extensions. We recommend that organizations pay close attention to these standards and their use in real world case studies. The reasons why these are candidates instead of full recommendations are stated in each description, below.

¹⁷ <http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/>

¹⁸ <http://blogs.msdn.com/jevdemon/archive/2005/05/05/415126.aspx>

WS-BPEL

Formerly known as BPEL4WS, BPEL is the *de facto* standard and state of the art for specifying Web Service-based business process orchestration and workflow. SOA Composite Application Development frequently involves workflow services, which are often implemented as a stateful service with a pre-defined start and end state, and a process definition. This process definition language is Business Process Execution Language, or BPEL. BPEL is widely touted as a business process standard by many vendors, particularly Oracle (via their acquisition of Collaxa)--however the number of customer references and case studies is still low at this point.

WS-Policy

WS-Policy will provide configuration of WS-Security through an XML Configuration file. This standard has some extensibility built in and is envisioned to cover a larger area of policy management in the future. WS-Policy is a very important standard, but it is not yet ubiquitous.

ebXML and ebXML Registry and Repository

ebXML is an internationally supported standard promoted by UN/CEFACT, the United Nations Centre for Trade Facilitation and Electronic Business. As such, adoption among ebusiness providers, government and international suppliers is substantial. The functionality relevant to SOA includes business-to-business(B2B) message format integration as well as a fully auditable SOA repository functionality. ebXML registry and repository are suggested for situations which require integration with ebXML-based trading partners or an auditable repository for SOA messages and artifacts.

Registry vs Repository

A registry does not contain the actual objects, implementation or artifacts associated with an SOA. Registries are expected to contain references or pointers to those items, as well as metadata about those items. On the other hand, a repository actually stores the objects themselves in addition to just references and models.

Intentional SOA Non-Standard Requirements

Most SOAP extensions are in the process of being standardized. The following areas of WSDL and UDDI have not yet been standardized, but are needed by customers seeking to obtain some of the business value points of SOA. These areas are included because solutions are needed, but standards have not yet arrived in these areas. Intentional SOA

recommends that organizations work with vendors with a commitment as well as a roadmap to implement major standards.

Extending WSDL with Contracts

In the case of WSDL, one important part of a service which is not provided by the WSDL is the “contract”. Contracts are widely discussed in SOA design, and can range from low level addressing terms up through security parameters, unique consumer specific functional declarations or Service Level Agreements (SLA).

A contract is an agreement between a service provider and service consumer. A contract extends the interface of a service by adding terms that define the relationship and expectations that go with it. Contract terms can include the specific security protocols and providers, SLA terms such as response time, and even higher level contractual terms such as billing, auditing, metering and legal terms.

Contracts are part of SOA metadata, and should be stored in a deployment descriptor or an SOA Metadata Platform. The ability to flexibly configure new contracts based on arbitrary provider/consumer pairs provides business agility without resorting to cumbersome custom software development. Instead of writing a whole new service for each service consumer, a new contract can be formed declaratively.

Extending UDDI with Identities, Contracts and Other SOA entities.

The ability to discover WSDL interfaces is essential to an SOA. Having a *Web Services Registry* is a key architectural building block of an SOA. However, even with a Web Services Registry in place, it is possible to descend into Web Services chaos. In the business reality of SOA, issues such as Access Control, Service Level Agreements, Billing, Metering, Service Publication Approval, and many other business issues need to be taken care of above and beyond basic WSDL discovery.

UDDI is a provider centric model for storing service metadata. In addition to information about the service and the providers, UDDI can be extended to include information about consumers and contracts.

Intentional SOA Integration Cost Recommendations

Organizations should maintain a list of standards for developing SOA. Departments and subsidiaries should be utilizing the same subset of standards in order to maximize interoperability and reduce the cost of integration.

In addition, organizations need to ensure that published services comply with standards such as the WS-I basic profile. Compliance checking should be an automated part of the IT governance system surrounding the deployment of Web Services.

Intentional SOA suggests that organizations will need to extend SOAP using standard WS-* protocols. In addition, Intentional SOA recognizes that non-standard extensions will be needed for WSDL and UDDI in order to reach the business value points of SOA. Intentional SOA recommends caution when extending standards and suggests getting assurances from vendors about their roadmap and commitment to standards.

2. Increase Asset Reuse

The Intentional SOA approach to asset reuse centers around establishing a system of record for all services.

The first Intentional SOA Readiness Questions relating to Asset Reuse is “How many services do you have at your company?” Surprisingly, many organizations do not know the answer—or worse yet think they have the answer, but are incorrect. In order to establish this answer, a typical organization catalogs all services to identify redundancies and opportunities for reuse. In the early stages of Web Service enablement, there are a small number of services available in the company. Services are discovered by talking with coworkers at the water cooler, or described in a large Microsoft Word document. As the number of services starts to move from a dozen to over a hundred, a more comprehensive approach is needed.

Typically, at the core of a cataloging project is a system of record. Intentional SOA suggests that this system of record should be a Web Services Registry. However, other types of service registration systems may also suffice—the key at this stage is to use the system of record to drive a social process across the organization to register and account for all available services.

By cataloging services in a registry, organizations gain visibility and insight into the large number and diversity of services in their organization. Due to the highly parallel and distributed nature of Web Services deployment, there may be dozens of ERP projects, CICS applications, J2EE applications, .NET applications and other service-enabled projects happening all over the company. In a company of sufficient size, there are

virtually always Web Services available that are not known to all of the participants in an SOA project.

In some cases where development is distributed globally, a registry can serve to clarify aspects of a set of services being built in each location. This is especially vital in the case of “cross-cutting concerns” or infrastructural services needed by all other services in an SOA. In one example, a loan processing application being developed in India had a concisely defined specification and scope, as did a credit application service being developed in New Jersey. However, both of these services needed an authentication service in order to function. Without a registry, both teams were building security services in order to support their applications—whereas proper reuse processes would enable these teams to coordinate their efforts better.

Non-Web Services Interfaces in Your SOA System of Record

Eliminating redundancy and achieving reuse is also a priority for non-Web Services components of SOA. Often, the same organizational processes and IT governance are needed for the publishing and management of non-Web Services components. In these cases the system of record should have the ability to reference if not store non-Web Services components, and to provide a rich metadata attribute system for those services.

Many of these services can be migrated to Web Services using a variety of tools and technologies, including standard J2EE and .NET tools as well as available tools for Web pages, 3270 terminals and other mainframes, CORBA and a host of other solutions. However, in some cases there may be legitimate reasons why these services are not Web Services-enabled, including cost, as well as the possibility that the services are provided by an external partner, or that these components are internal to a higher level service.

Using a Web Services Registry based on UDDI can be a good starting point for such a cataloging process. To be more precise, a large organization, such as a Fortune 500 or government branch, is very likely to have multiple registries. Each registry provides a meaningful collection of services which represent the necessary business processes for that community. Registries are often *federated* meaning that multiple registries are interconnected so as to form a larger registry of registries.

Asset reuse should be a part of IT Governance—a set of processes that ensures that activities are efficiently provisioned and that limited IT resources are applied in the most strategic ways possible.

Intentional SOA Reuse Recommendations

Organizations should identify and catalog services in a system of record, in order to ensure reuse.

A Web Services Registry, or federated set of registries can provide a valuable system of record for this purpose.

Organizations should establish IT Governance processes to ensure reuse.

All IT software deployments should be considered for reuse processes, not just Web Services deployments.

3. Support Business Agility with SOA Metadata

Business agility was defined by a panel at the Infoworld SOA Executive Forum¹⁹ as the ability of an organization to see and rapidly respond to changing conditions. Rapid response can be enabled through configuration rather than customization of software—the speedy changing of metadata attributes on top of services.

A Word About Metadata

Metadata is defined in the Wikipedia²⁰ as “Data about Data” and is characterized as having three types: data warehouse, filesystem and program metadata. While this is far from a complete characterization of metadata, the distinction between executable (program) metadata and other types of metadata (data warehouse and filesystem) is significant. Program metadata influences the execution of a system and is used by the runtime environment to modify machine behavior. Other types of metadata are primarily used for resource description and discovery.

In the context of SOA, both types of metadata are important. One of the requirements that supports loose coupling is the ability of interfaces to be both self-describing and discoverable. Also, the ability to impact the execution of services through contracts or metadata attributes requires the participation of the runtime system.

¹⁹ <http://www.infoworld.com/event/soa/executive.html#t2>

²⁰ http://en.wikipedia.org/wiki/Metadata_%28computing%29

The Ability to See Changes in the Business: Enterprise Systems Management

Businesses have robust systems for monitoring enterprise systems such as network nodes, clustered disk arrays, server farms, databases, transaction processing systems and other resources. These Enterprise Systems Management (ESM) packages come from a variety of vendors.

SOA and Web Services solutions are no exception—a robust solution is needed in order to deliver Web Services information into ESM consoles. Once the information is delivered to ESM, the traditional integrated view, alerts, availability, service level monitoring, provisioning, service management and other functions can become available to your Web Services.

A number of benefits accrue to organizations who monitor applications at the Web Services level. Chief among these is the ability to measure high level business functionality. Instead of reading SQL calls into a database, business users are empowered to ascertain the number of Purchase Orders issued, or the number of new customers added to the system. These types of events fall into the category of Business Activity Monitoring.

The Ability to Rapidly Respond: Processes and Declarative Programming?

The responsiveness of IT architecture is achieved by the enablement of flexible and dynamically configurable software functionality. These capabilities are characterized by the speed which new applications can be developed, and the ease with which such applications can be understood and configured by business users. A business is more agile when it can quickly add new customers, develop differentiated products and services, and develop and refine business processes.

Business Processes Programming

Once a robust series of services are available in the network, a new type of application development becomes possible. Gartner refers to this as “Composite Application Development.” One of the key benefits is the ease and speed at which applications can be developed. The most common type of composite application is a “workflow” application, which is a process oriented application which involves messages that sequentially move from service to service. In addition to moving from one machine to the next, workflows can also include human intervention such as approval processes, reviews, forms processing and other interactions.

Declarative Programming

An underappreciated aspect of SOA is the ability to enable declarative programming. Declarative programming involves the application of metadata, sometimes in the form of “deployment descriptors”. This metadata includes descriptions of the service, but can also influence the runtime attributes of the service. An example would be metadata that describes a contract between service consumers and producers. Declarative programming by manipulating metadata attributes can provide a very fast way for non-technical users to configure runtime instances of services, particularly if a simple configuration user interface is provided. The concept of contracts enables a single service to be deployed to many endpoints all of which have different requirements.

SOA Metadata Enables Declarative Programming

Much like the core of the Windows Operating System is the Windows Registry, an SOA needs a location to store metadata about all of the components in the system. Intentional SOA recommends that an SOA Metadata Platform (based on a Web Services Registry) be used to store metadata such as contracts.

Handling Declarative Metadata in Application Platform Middleware

Platform middleware provides a number of facilities for declarative programming and middleware. In J2EE, the use of JSR175 metadata attributes and deployment descriptors provides a strong mechanism for managing metadata in Java code. There are also robust mechanisms for handling deployment descriptors and property files in .NET. ESBs also have a very similar form of deployment descriptor which enables loose coupling and metadata management on a per-service basis by using XSL stylesheets, XPath expression, scripts, and parameters in an ESB specific configuration repository.

Three significant problems arise when users depend solely on application platforms to manage middleware.

Firstly, business agility is impacted, because there is no business user interface for managing this metadata. Typically, deployment descriptors and software embedded metadata are used by IT developers and technical staff. Sending customizations of this type back to IT hampers agility and results in long cycle times.

Secondly this style of metadata does not cut across platforms. Having standard metadata that can be deployed across J2EE applications, .Net applications and even systems like Web Services enabled CICS mainframes require an SOA Metadata system that can handle heterogeneous services.

Lastly, there is no way to uniformly view all of the metadata distributed in large numbers of different platform-specific deployment descriptors scattered throughout your organization. This means that there is no mechanism for viewing and understanding how attributes, contracts, policies and terms are implemented across all of the systems

SOA Metadata Platform Features

Intentional SOA defines an SOA Metadata Platform as a UDDI-compliant Web Services Registry with applications built on top which are designed to support the four business value points of SOA. Features which may help support these points might include:

- An easy-to-use configuration tool for configuring metadata such as contracts
- Role-based security and access control for services
- Standards Compliance Tests such as WS-I Basic Profile test
- The ability to recognize which consumer invoked what service and how often for capacity planning and load balancing.
- A repository for storing assets and artifacts associated with services
- IT reuse and governance capabilities which extend to non-Web Services

- Provider validation, Service Level Agreements, contract negotiation, and trust establishment.

In addition to contracts, an SOA Metadata Platform can drive other runtime issues—provisioning, activation, assurance, and maintenance/retirement.²¹ As stated before, driving runtime often requires support for contract terms within the runtime environment.

Intentional SOA Business Agility Recommendations

Organizations should integrate Web Services brokers and agents into their Enterprise Systems Management consoles for monitoring their SOA and Web Services implementations.

Organizations should store SOA metadata (including contracts) in a way that is:

- Cross Platform
- Business User Configurable
- Provides a single view of metadata across all services

Organizations should establish business agility metrics in order to determine the ROI of their projects. Included in this could be terms such as the number of customers gained through rapid on-ramping or the cost reduction of having business users configuring application systems at runtime.

4. Reduce Business Risk with Runtime Policy Management

Intentional SOA reduces business risk by using SOA for regulatory and corporate policy compliance. Analyst firm Redmonk describes SOA as “Compliance Oriented Architecture²²”

In this white paper, Steven O’Grady outlines a number of core services which can be deployed to support regulatory compliance capabilities, such as Access Control (used for example to protect patient medical records under HIPAA), Analytics (for Basel II metrics), Destruction (for SEC records management compliance), Disposition (DoD 5015.2), Notarization (FDA CFR 11), Policy Engine (Gramm-Leach-Bliley privacy), Version Control (SEC submissions), Workflow (CORI Criminal Offender Record

²¹ “Web Services in the Investment Industry, Mid-2004 Update,” [Mary Knox](#) and [Maria Luisa Kun, Gartner, Inc., ID Number G00123851](#), 4 October 2004.

²² http://www.redmonk.com/public/COA_Final.pdf

Information). This is only a partial list of compliance oriented services outlined in the paper.

Compliance issues are very common in geographically dispersed and multinational organizations and yet, according to a 2003 META Group survey, only 20% of Global 2000 companies have formal IT governance practices.²³ Also, business units may not be fully aware of the issues related to intellectual property, lifecycle management, privacy, and security surrounding Web Services. Ultimately, the CEO, CFO, CIO, CISO/CSO, and/or CPO will be held responsible for any breaches. It is absolutely essential that IT step in and take control of proliferation of Web Services and address governance issues that provide detailed lifecycle management of a Web Service—its creation, usage, QoS/SLA, and end-of-life requirements.

Recent legislation passed in the U. S. (e. g., California SB1386, Gramm-Leach-Bliley Act, Health Insurance Portability and Accountability Act, USA PATRIOT Act, Sarbanes-Oxley Act, and SEC Rule 17a-4) and elsewhere (e. g., Basel II in Europe) require organizations to provide auditable, clear systems controls, and this is affecting how systems architectures are being approached and revised. Architectures are no longer being designed just around function, performance, and scalability. As an example, because of the need for auditability, a Web-based security system must offer the ability to see into its systems to ensure proper authorizations and authentications as well as to determine who is connecting to which systems. This translates to determining which consumer invoked what Web Services and for what reason. Without a registry, this is a daunting task. A Web Services Registry is an essential ingredient in any company's initiative to adopt SOA. While it may be true that a registry is not required to build and deploy Web Services, it is required if one intends to manage the ensuing environment. Registries give companies a central point for viewing available services and a structured process for keeping track of Web Services in use.

Using Web Services to enforce policy across many disparate services requires uniformity of policy enforcement. A presentation by Burton Group discusses the concept of Policy Enforcement Points²⁴ as a significant theme for the role of registries. Ultimately, managing runtime policy depends on the relationship between the system of record and the runtime environment.

²³ "Portal Governance," Line56, February 8, 2005.

²⁴ <http://www.burtongroup.com/events/downloads/ppt/972.ppt>

Intentional SOA Business Risk Recommendations

Organizations should consider SOA for developing a compliance oriented set of software services.

Build a best of breed portfolio of commercial off-the-shelf components to support compliance-oriented applications.

Organizations should seek to use policy management and policy enforcement points as a means of controlling policies uniformly across service platforms.

Intentional SOA Recommended Next Steps

Intentional SOA doesn't stop after reading this white paper. Immediate steps would include documenting your organizations SOA readiness by using the SOA Readiness questions presented here in the white paper.

Other steps would include contacting several vendors to hear their perspectives. Vendors often get a bad name for trying to push their products aggressively. But ultimately, vendors gain valuable knowledge in serving the requirements of their customers. So contact a number of vendors to obtain a variety of opinions about what is needed to achieve SOA business value. Obviously be wary what vendors say. Get the opinion of competing vendors on topics of importance to determine what are commonly accepted facts and what are being contested.

Join a users group, whether online or in person. Go to SOA conferences. There's very little that can beat talking with other end user SOA builders. Download some more white papers (and discuss them in your organization). Follow up on case studies and call the people involved. Actively monitor or join standards bodies and technical committees in order to evaluate the latest emerging standards.

Try to organize your company or group around SOA. SOA requires many changes including job titles, organizational support systems and new business processes. By establishing SOA-focused councils, groups and boards, your organization will be a step ahead in achieving Intentional SOA.

The author of this paper also welcomes feedback, comments and discussion. Feel free to contact him at the email address Miko (at) Infravio (dot) Com.

About the Author

Miko Matsumura is currently the vice president of marketing for Infravio, Inc., a leading provider of SOA Web Services Management products.

Before joining Infravio, Matsumura emerged as an industry thought leader at The Middleware Company, where he was a co-creator responsible for building the partner program for SOA Blueprints, the first complete vendor-neutral specification of an SOA application set, supported by BEA, Borland, HP, Microsoft, Oracle, Sun Microsystems, Veritas and others. As the vice president of product marketing at Systinet Corporation, Matsumura worked with the executive team and offshore development center on product development, product strategy, and outbound marketing, including representing the company at industry events.

Earlier in his career, Matsumura held the position of Chief Java Evangelist for Sun Microsystems, where he was a visible spokesperson for Java technologies and worked closely with Java ISVs and licensees to further the developer community. Before Sun, Matsumura worked at Wired Digital (acquired by Lycos) and the Well online community (acquired by Salon). He has also worked extensively with software start-up companies, including Biztone and Kalepa Networks (acquired by Semio) raising more than 12 million in capital for Java start-ups.

Matsumura is currently a limited partner with Focus Ventures and was an advisor to the Asia Java Fund, as well as start-ups TogetherSoft (acquired by Borland), Dejima (acquired by Sybase) and Kendara (acquired by Excite).

Matsumura holds an MBA from San Francisco State University and a Masters Degree in Neuroscience from Yale University.

Appendix A: A Peek into an SOA Repository

Earlier in the white paper, a distinction was made between a Registry and a Repository. Registries contain references, interfaces and metadata, while repositories contain files, binaries and payloads.

A healthy SOA Repository contains more than just references to WSDL interfaces. What types of entities would be seen in a healthy SOA repository?

WSDL interfaces

The primary purpose of a Web Services Registry is to contain interfaces

Identities

First of all, a healthy SOA Metadata Platform needs to be able to track the identities of users, service providers and consumers. With respect to each there should be identity

information, usage information, payment or billing information and other key metadata to provide services.

Contracts

In addition to the people and the interfaces to services, the registry should contain information about their relationship

Access Rights

The ability to control visibility and execution based on identity is an important role function.

Schemas

XML schemas govern message format and often need to be referred to by development or runtime contexts

Sample Code

The ability to see sample code is a powerful tool for service consumers to understand and be able to exploit services discovered in the registry.

Demo Services

Demo services enable service consumers to play with the services without compromising the integrity of the full service.

Attribute Sets, profiles

A robust metadata system need the ability to flexibly add new metadata attributes and schemas based on application needs.

Dependencies

Tracking dependencies between services enables versions to be smoothly updated and maintains change control and a map for understanding interdependence in service implementation.

Metrics

By bringing usage metrics back into the registry, development can understand better the patterns of usage and provide better services and upgrades.

XSLT

Critical for doing data integration and transformations, XSLT can be stored in the registry to associate it with the appropriate integration points.

Taxonomies

Hierarchical methods of navigating services, taxonomies can be general such as the UDDI standard hierarchies or they can be industry-specific or customer-specific. Each

registry would be expected to contain a number of taxonomies through which services can be discovered

Conformance Documentation

Whether documenting conformance to technical standards or to corporate policies, these documents represent part of the overall SOA design

Auditable Message Logs

In some SOA design patterns a log of transactions and messages are needed. Examples of such messages might include a set of ebXML transactions with business trading partners. These messages and logs can be stored in a repository.

Security Certificates and Encryption Keys

Security artifacts needed for WS-Security, SAML, and other security applications

Appendix B: InfraVio X-Registry Metadata Platform

InfraVio X-Registry™ provides an SOA Metadata Platform as described in this white paper. This functional architecture is provided as an example of a business focused metadata platform including contracts, governance processes and other capabilities.

X-Registry includes metadata entities other than the basic UDDI businessEntity, businessService, bindingTemplate, tModel, publisherAssertion, and subscription entities. These entities are focused on describing the service provider and metadata about the provided service.

X-Registry Metadata Entities

Service Consumer Identities

By modeling these entities, the system enables the deployment of metadata descriptors on a per consumer basis. Without providing a model of consumer identity, the system would have no way to provide user specific context with regard to service delivery

Contracts

One extremely powerful form of service delivery context is the contract. A contract is metadata around a service. Separate contracts can be established in order to customize a service to a consumer or class of consumers.

Processes

Business agility is often achieved through developing business processes that differentiate a company from its competition. The ability to monitor and change business processes based on changes in strategy and implementation tactics is an essential part of reaping the benefit of SOA.

Dependencies

Services have complex interdependencies on other services. By their nature, services are opaque to the consumer and therefore the dependencies are not clear at the consumer side. However, since Web Services enable radically distributed application components built out of other services, a clear map of interdependencies is needed in order to manage service levels, change management and to deal with versioning of services.

Access Rights

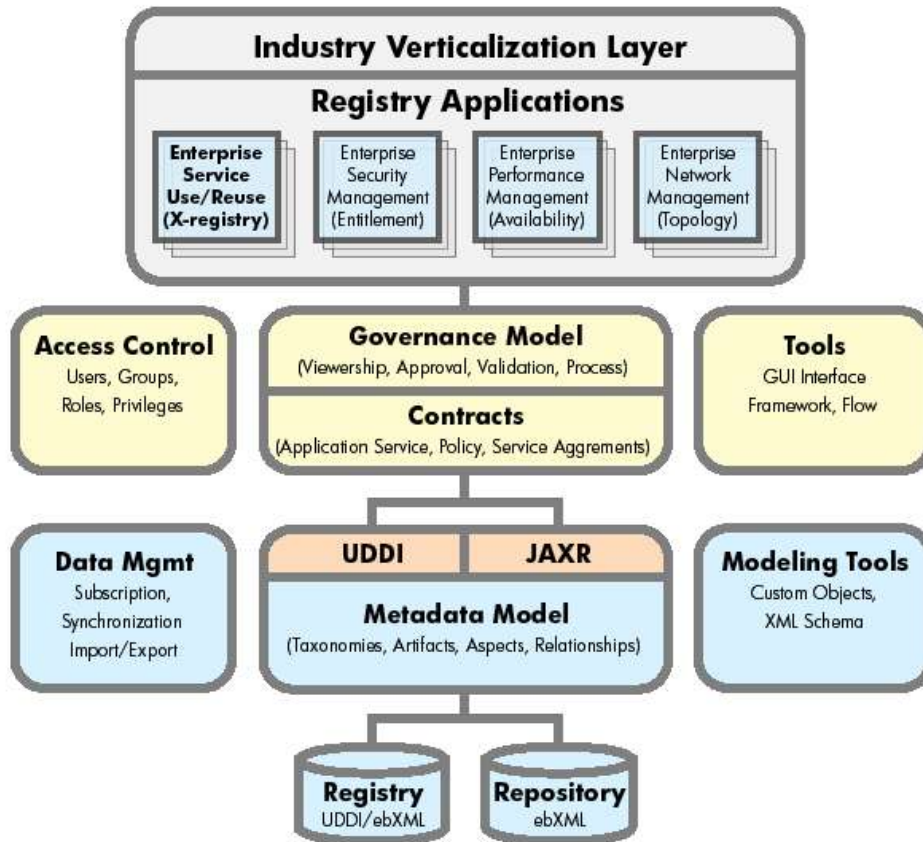
Especially when services are provided to external parties, but even in cases where internal services provide sensitive and regulated private information (for example, patient medical records), tight access controls are needed on service interfaces.

Flexible Metadata Attributes

In addition to fixed schemas and attributes, an SOA Metadata Platform should provide flexible means for adding metadata attributes to all entities in the system.

Organizations must control, manage, and monitor the interactions between Web Services providers and consumers. Web Services Delivery Contracts™ contain configurable terms, defining both the delivery preferences (i.e., security, messaging, and transformation) and the operational requirements (i.e., SLA, alerts, and notifications) between Web service consumers and providers. In addition to the standard UDDI registry interfaces, X-Registry provides support for standard repository functionality which can be accessed via JAXR or ebXML interfaces.

The diagram below provides insight as to the layers InfraVio provides above and beyond the UDDI layer for support of the four business value points of Intentional SOA. Each of the layers is described below:



X-Registry Functional Model

Registry Applications(SOA Management)

X-Registry provides a comprehensive Web-based user interface. From these screens you can manage all aspects of your SOA including policy configuration and governance. This dashboard includes connections to Infracore X-Broker™ runtime management and Infracore X-Console™ Web Services monitoring solutions, also from Infracore.

Governance

X-Registry ensures compliance, conformance, data integrity and quality of your SOA implementation. These can include Viewership, Approvals, Validation and Processes for interacting with the registry including publishing and reusing services.

Contracts (Service Configuration)

Infravio X-Registry contains a powerful service configuration capability called Web Service Delivery Contracts™ (WSDC). WSDC can enable different users of a service to be configured for different levels of access control, logging and monitoring, service-level agreement (SLA), metering, and business contract terms. This functionality is ideal for enabling external partners, as well as for providing different services to different groups of users.

Repository

Beyond merely storing WSDL interfaces and metadata models, X-Registry 4.7 can store assets associated with services, users, publishers and relationships including dependencies, schemas, sample code, documentation, identities, Service Level Agreements, Web Service Delivery Contracts™, security keys and certificates, and a host of other SOA resources.

Verticalization

Infravio X-Registry 4.7 includes an industry verticalization layer which can enable vertical solution packs to be deployed in telecommunication, healthcare, financial services and other industries. These vertical solution packs include industry-focused taxonomies, metadata schemas and governance processes