zapthink research report

THE VALUE PROPOSITION FOR SERVICE-ORIENTED INTEGRATION

ZapThink, LLC • 11 Willow Street, Suite 200 • Waltham, MA 02453 • info@zapthink.com • www.zapthink.com

THE VALUE PROPOSITION FOR SERVICE-ORIENTED INTEGRATION

USING WEB SERVICES AND SERVICE-ORIENTED ARCHITECTURES TO FACILITATE INTERNAL AND EXTERNAL INTEGRATION

March 2003

Analyst: Ronald Schmelzer

Abstract

Connecting systems both within the enterprise and among suppliers, partners, and customers is of critical importance to today's enterprise. However, integration remains complex, expensive, and risky. The increasing movement toward data virtualization, B2B systems, and legacy reuse is driving a need to integrate dozens, if not hundreds of systems in a single environment. The end result is a tangled web of point-to-point integrations that becomes increasingly brittle over time. The costs for both maintaining and changing systems can become exorbitant, since changing requirements necessitates manual recoding of applications. Enterprises are thus faced with an integration problem that grows at a rapidly increasing rate.

Standards-based *Service-Oriented Architectures* (SOAs) built on Web Services interfaces enable a new approach to integration. Known as *Service-Oriented Integration* (SOI), this integration approach leverages open standards, loose coupling, and the dynamic description and discovery capabilities of Web Services to reduce the complexity, cost, and risk of integration. While SOAs have certainly been around for a while, and certainly users have been experimenting with point-to-point implementations of Web Services, it is the combination of Web Services and SOAs that is especially potent.

Targeted at companies who must contend with an overwhelming number of integration projects, this paper introduces the concept of SOI, how it improves upon previous integration methods, and identifies key elements that are required for an SOI solution. Furthermore, this paper explores the ROI and value metrics users can realize from SOI approaches, and when SOI approaches are applicable and useful.

Table of Contents

I. The Integration Challenge

Some of the most important assets for any organization are the information they create and store, and the processes by which they use and assimilate that information. However, in order for companies to realize the value of information stored in various systems and processes, they must integrate and connect the disparate silos of information in the enterprise. Integration is not a simple issue of merely plugging two systems or organizations into each other. The vision of "plug and play" application and system integration is a pipe dream that may come true some time in the future, but today's enterprises face the more immediate challenge of connecting relevant systems in a manner that is flexible, cost effective, manageable, and secure.

Over the years, companies have pursued a wide variety of integration approaches aimed at solving the underlying requirement for connecting disparate systems in the enterprise. Why are the problems with integration still troubling companies, even though these solutions have been around for a generation or more? On one level, the cause is the lack of standard ways of programming different systems to communicate. For any two different systems, the traditional approach to integration is to write programming code for each system that teaches it how to talk to the other system. Such an approach is expensive and time consuming, and doesn't scale well or respond to change in a flexible way. This approach to integration is also *tightly coupled*, which means that one programming team must control the integration code on both systems to get them to communicate with each other. In addition, such integration is point-to-point, which means that the complexity of maintining the connections and managing the change of distributed systems dramatically increases as the number of connected systems goes up.

The result of all these attempts to solve the problems of integration is a mix-and-match set of approaches and technologies that are ill-suited to fundamental requirement of seamlessly connecting disparate systems in the enterprise. Clearly, what is needed is not just a new set of integration technology approaches, but also a fundamental change in the way companies architect systems and processes to handle integration requirements.

Drivers for Integration

Why is integration important to the enterprise? The following major reasons are some of the more tactical motives for integrating disparate systems:

- *Connect organizations* – Integration allows business units and third parties to interact as a connected entity, facilitating functions vital to the flow of commerce. Furthermore, companies seek integration as a means of "extending the enterprise" to include new users outside the corporate firewall as participants in corporate IT systems.
- *Get a better understanding of customers and business operations* — Integration allows a company to serve its customers and stakeholders better by gaining better access to corporate information.
- *Increase utilization of existing systems and reduce complexity* – Integration allows enterprises increase utilization of applications that are already deployed in the enterprise, reduce IT infrastructure complexity, simplify management, and reduce the cost of change.
- *Allow systems to evolve* – Provide flexibility in the IT environment by allowing legacy systems to expand to new areas of functionality or provide a way to migrate functionality between systems without wreaking havoc on the rest of the corporate computing ecosystem.
- *Value-add existing applications* – Integration allows companies to make better use of existing systems by continuing to find ways to leverage these systems for new applications.

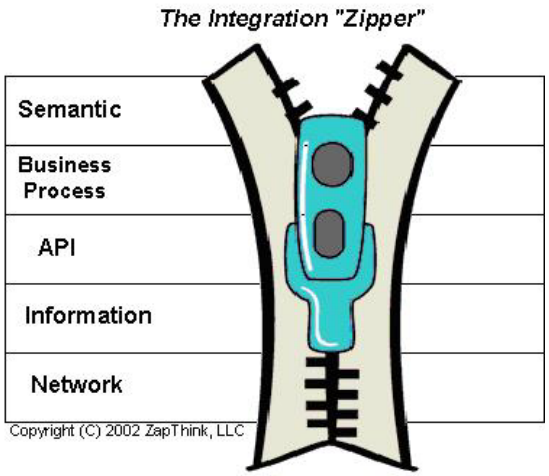
However, a more strategic driver for organizations is the need to achieve greater *business agility.* Business agility is more than simply being able to respond quickly to change; it also means the ability to leverage change for competitive advantage. For businesses to be agile, their technology must be agile as well. Yet, many integration approaches fail to meet this basic business goal.

The Integration Zipper

Part of the reason why companies fail to solve their integration challenges is that integration problems never go away. Rather, as standardized, "commodity" technologies increasingly solve the problems for lower-level integration issues, the higher-level integration issues remain. These issues relate less and less to specific implementation decisions; instead, they are increasingly business and concept-oriented. Instead of having to worry about the connectors, adapters, and interfaces between systems, businesses need to worry about how to map the way that they represent information itself. Even if the connection between businesses and systems were seamless, companies must still resolve the issue of the meaning of the information passing between the organizations.

As a result, integration is a concept that embodies not only a range of different objectives, but also spans a number of different logical levels. Rather than viewing these levels as a stack, it makes more sense to think of integration problems as a "zipper," where lower level integration problems must be solved before higher-level issues can be successfully addressed. Figure 1 illustrates this concept below:

Figure 1: The Integration "Zipper"



Enterprises have successfully addressed the challenges of integrating at the network and information layers using technologies such as TCP/IP, HTML, and XML at the bottom of the zipper. However, enterprises are currently facing the challenge of integrating disparate application-level interfaces as they move up the zipper. Rather than seeking simply to connect to machines on the network or access underlying data sources, companies are looking to access the business logic of mission-critical systems such as CRM, ERP, legacy or other internal or external systems.

Instead of simply connecting different interfaces together in a hodge-podge manner, we are in effect creating *composite applications* that create a higher-level application composed of different, discrete systems that all together meet the requirements for a particular business need. For most enterprises, the majority of logic required to run their business already exists somewhere in the application portfolio. However, companies increasingly need to realign these discrete functions to support new and emerging business needs and processes that require the combination of existing functionality and new capabilities in different ways. The desire to create this new class of composite application and integrate at successively higher levels of the integration "zipper" remains as the primary integration challenge facing enterprises today. As companies successfully solve the lower, technology-focused integration issues, they soon face the reality having to solve successively more complicated, but business-oriented layers of integration.

The Challenge of Integrating Legacy Systems

Complicating these integration challenges is the desire to make use of systems that have long operated successfully in the enterprise computing infrastructure – so called "legacy" systems. Much of the core functionality needed for enterprise IT initiatives is contained on legacy systems, but that functionality was built to serve a specific departmental purpose. Consequently, the required data and logic remain trapped in host silos that aren't flexible enough to meet the needs of emerging sets of users or processes. The challenge for IT is to transform the host functionality so it can be combined with new logic and used in Web applications, packaged applications, or portals.

One of the primary reasons why legacy systems aren't flexible enough to meet business demands is that presentation logic, business logic, and data are frequently intermingled. It is often very difficult to extract the required functionality of the system in a way that is separate from the presentation and data layers. In addition, legacy system flexibility is encumbered by a lack of APIs that provide the appropriate level of integration required by developers. So, while legacy systems are an absolute requirement for system functionality, working with them in the context of a heterogeneous environment is frequently very complex.

In addition, companies must grapple with the current economic forces that encourage them to approach their IT investments with a *thrift* mentality. Thrift, however, means more than simple cost savings. True thriftiness means making do with what you have – squeezing value out of every asset. No longer are "rip and replace" technology strategies viable in which companies simply throw out their old implementations and replace them with newer solutions that may not fully solve the issues that led to the replacement of the old system in the first place. Companies must consider their existing technology investments to be systems that will continuously be leveraged until they can be easily replaced. Therefore, there is increasing need to wrap legacy systems in new interfaces that allow them to be used in ways that the developer of that system didn't envision.

II. Challenges with Existing Integration Solutions

Traditional integration approaches have aimed at solving three different intersystem communication challenges:

- *Enterprise Application Integration (EAI)* – Solutions applied to solving the problem of integrating disparate systems, applications, and data sources within the corporate enterprise.
- *Business-to-Business Integration (B2Bi)* – Integration solutions focused on enabling secure, robust, electronic communications among businesses and their information systems, across the corporate firewall.
- *Data Integration (DI)* – Integration solutions focused on connecting data storage and information representation systems for the purpose of facilitating data exchange.

Using these approaches, companies have often implemented "point-to-point" integration approaches where systems that need to communicate are connected directly to each other. In this scheme, the number of integration or interconnection pathways that must be established grows geometrically (or n-squared) with the number of systems to be integrated. Unfortunately, this integration problem is compounded by the fact that most companies perform integration in an *ad hoc* manner, narrowly focusing on short-term integration needs rather than overall solution effectiveness.

As a result of the complexity and chaos of *ad hoc*, point-to-point integrations, efforts have been made to standardize and productize various integration solutions. These EAI, B2Bi, or data integration solutions have the following primary benefits over ad-hoc integration:

- *Reduce the cost of adding new systems* – Integration solutions reduce the cost of adding new systems to a corporate ecosystem, since they can be added to the existing integration framework.

- *Better performance* – Integration solutions offer better throughput, scalability, and reliability than ad-hoc integration approaches.
- *Maintain legacy systems* – Integration solutions allow legacy systems to remain operational in the enterprise contributing to new applications for new users.
- *Reduce time to market* – Integration solutions allow businesses to shorten the time it takes to introduce new products or services.

Data Integration Approaches

A mechanism for accessing system data is to directly access the underlying databases and file structures that store the application information. This form of data access incorporates three major concepts: *extract, transform and load* (ETL). The ETL process extracts data from a system on a scheduled basis by copying batch feeds of data from one data source, transforming the data, and then loading the transformed data onto a separate system. However, ETL processes by definition are point-to-point, tightly coupled, and asynchronous since the data load process can occur at any time – most likely in batch processes when the data are already stale. In addition, when integrating with data through ETL means, users frequently bypass the business logic inherent in the system, making it difficult to get timely data that is relevant and related to the business need.

EAI and B2B Integration Middleware

In the past, integration among systems has been accomplished by implementing a proprietary middleware tier such as those provided by traditional Enterprise Application Integration (EAI) or Business-to-Business Integration (B2Bi) solutions. These solutions are built primarily on proprietary or system-specific messaging platforms that aim to provide a complete, end-to-end platform for integrating and communicating with various business components. The typical method for accessing these systems is through a wide assortment of pre-built adapters that provide bi-directional connectivity to many types of applications and data sources, such as enterprise software applications, databases, file systems, directories, as well as mainframe and other legacy applications. In simple terms, the way these integration solutions work is by extracting or inserting data from these various adapter-enabled systems, transforming the data and converting their representation or schema to a different format, and then shipping the data to their destination.

One of the traditional EAI architectures most commonly in use is the "hub-and-spoke" topology in which a single integration server functions as a central point ("hub") that handles information exchange and transformation for many different applications and data stores ("spokes"). Implementation of the hub-and-spoke architecture is not particularly efficient or scalable, and is often the root cause for many failed EAI solutions. Hub-and-spoke solutions are resource-constrained, since all processing occurs on a single integration server. At some point, the amount of traffic and number of connections will saturate the available resources of the integration server, degrading system performance. The integration middleware hub is also a single point of failure, which limits the architecture's robustness.

While increasing the number of hubs or distributing hubs in a federated environment can improve the scalability and robustness of the hub-and-spoke

model, the bus topology provides a better model for EAI. In the bus topology, integration brokers sit on each of the data and application sources and share their information on a common bus. These brokers transform application data to a standard message format that is published directly to the bus. This activity generates an event that message subscribers can retrieve and transform into their own native data format. For this reason, the bus architecture is also known as a "publish/subscribe" architecture.

However, these integration approaches are fundamentally brittle. Even though EAI solutions provide a brokered integration environment, the end points are fundamentally hardwired and difficult to change. Changing and maintaining integration endpoints is typically an expensive and complicated endeavor, since the centralization of the integration approach requires each endpoint to be individually configured in the system. In addition, brokered environment typically are difficult to extend outside the walls of the enterprise, limiting such approaches to integration scenarios that are centrally controlled. EAI and B2Bi approaches, while meeting the needs of enterprises for decades, are expensive, complicated, and cumbersome ways of integrating an ever increasing set of applications, systems, and businesses.

Requirements for an Optimal Integration Solution

Many of the current implementations of EAI solutions have been targeted at specific, custom problems and do not create flexible processes that can reused in a variety of different, higher level business applications. So, while EAI systems *may* be enable reusable processes, their inflexibility and expense prevents them from being used that way in practice. Also, there has been little incentive for EAI solutions vendors to make their systems more efficient. Feature and function enhancements such as more adapters and process improvements are more important to the product's success than improving the efficiency of pushing information through the application or connecting to large numbers of systems.

Rather than thinking about how to get information into or out of different systems, we can think about merely how to expose a system in a Service-oriented manner to whatever system cares to access it. In this way, we release ourselves from thinking of information in a point-to-point fashion and instead think of information as freely available on a network of Services. In this way, we can reuse existing functionality in different ways – for building new, composite applications, or extending existing functionality. Thus, the requirements for a new class of integration solution include:

- *Loosely-coupled* – Consumers don't need to have knowledge beforehand about a given piece of system functionality, other than where to find it. Application functionality and the programs that invoke them can be changed independently of each other, instead of requiring a redesign of the involved components.
- *Coarse-grained* – Rather than interacting with a large set of detailed, fine-grained APIs, users can interact with systems through *coarse-grained,* business-level interfaces that roll up the functions of many different API calls into a small number of business-oriented messages. Although low-level, fine-grained, services already exist in the enterprise, they exist separately from each other in discrete silos, and usually are not participatory in a service-oriented architecture. Often, the first step is simply to create fine-grained, low-level services and then bring them together as higher-level Services to produce a discrete set of capabilities that didn't exist before.

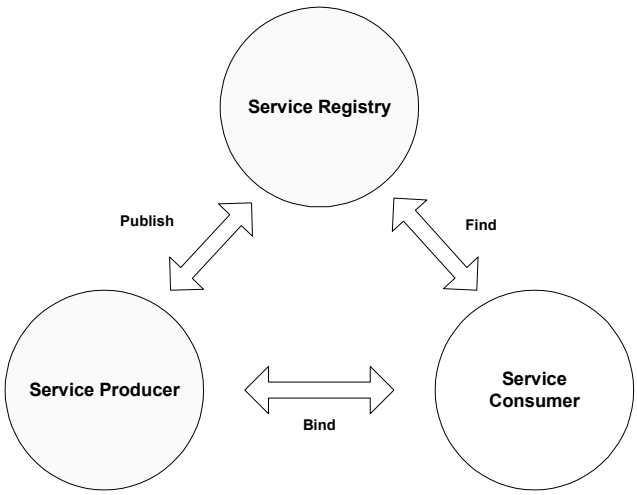
- *Standards-based* – Instead of utilizing proprietary or closed APIs that require users to learn the intricacies of a particular vendor's platform, integration tools that successfully address the challenges of business agility will be standards-based, lowering the cost of integration and allowing the widest possible range of developers.
- *Process-driven* – Instead of dealing with concrete requirements from business, users must be able to respond to changing requirements. The entire architecture—from the hardware on up—must reflect the business agility requirement, because any bottleneck in an integration implementation can substantially reduce the flexibility of the entire IT environment, and hence the business as well. Products and methodologies that allow line-of-business users to create and manage processes facilitate business agility.

III. Service-Oriented Integration

The challenges of integration combined with the opportunities posed by XML, Service-Oriented Architectures, and the Internet have combined to provide a new class of solution for integration: *Service-Oriented Integration (SOI)*. However, in order to understand SOI, we must first understand what a Service-Oriented Architecture is all about.

Service-oriented architectures are an approach to designing distributed computing infrastructures that considers software resources as services available on a network. Producers of these services must be able to publish information about them in a service registry or repository, where service consumers can then look up the services they need and retrieve the information about those services they need to bind to them. This "publish-find-bind" triangle forms the core of a Service-oriented architecture.

Figure 2: The Service-Oriented Architecture Triangle



Just as Web Services are an evolution of traditional distributed computing techniques, the practice of Service-oriented architecture is an evolution of the practice of enterprise architecture, which is an aggregation of all the individual IT systems within an organization. The potential rewards for enterprises that understand this evolution and make the move to such architectures are enormous. Finally, distributed computing technology promises to be flexible and

nimble enough to respond to business needs and provide the business agility that companies have craved for so long, but which has always been out of reach.

The vision of using Web Services-based Service-Oriented Architectures for integration is known simply as Service-Oriented Integration (SOI). Rather than explicitly declaring how systems will interact through low-level protocols and object-oriented architectures, as was the case with previous EAI and B2Bi efforts, SOI provides an abstracted interface that systems can interact with. Systems merely need to expose their capabilities as Services, and other systems that choose to interact with them can simply discover those Services and bind to them either at runtime or design-time. Rather than planning in advance how a specific application will tie into other applications, developers should plan, develop, and deploy their applications in a service-oriented manner. The point of integration is to allow arbitrary applications, systems, and data stores to communicate without concern as to the other system's requirements, and SOI fulfills these requirements. Specifically, companies should repeatedly identify the toughest integration problem in the enterprise that is suitable for a Service-oriented integration solution, and solve that problem in a Service-oriented way.

The Economics of Integration

Since Web Services, and in particular Service-Oriented Integration (SOI), improves the general approach towards integration, we can expect some significant improvements on ROI over what was possible with traditional approaches to EAI and B2Bi. The improvements to be seen include dramatic reductions in TCO as well as improvements on the tangible and intangible aspects of ROI as follows:

- *Reduction of Cost* – SOI approaches are easier to integrate, utilize lower-cost and more widely available tools, and require less time to create. This reduction plays into the hands of IT managers who have a big investment in existing "legacy" infrastructure and tools that cannot be simply "ripped and replaced."
- *Improvement in Efficiency* – SOI promises a high degree of reusability, faster time-to-market, and ability to integrate with third-party systems and trusted business parties. An increasing set of developer tools offer sophisticated functions to turn existing interfaces into Web Services, and thus participants in an SOI implementation.
- *Streamline Business Operations* – SOI solutions enable a common architecture and approach to be pervasive in an enterprise containing heterogeneous legacy systems. Generation facilities streamline the creation of Web Services interfaces, such as the generation of WSDL from existing service implementations or from models.
- *Faster time to market* – Web Services enable companies to become more agile, offering services in increments when they become available, rather than waiting for companies to change or complete whole systems.

Applicability of SOI Approaches

As discussed earlier, companies are really looking to integrate at various levels in the enterprise. However, not all of these different integration needs are best solved by a Service-Oriented Integration approach. Despite the increasing desire to move up the integration zipper, there are still a vast number of systems that need to be connected at the data tier. This need to integrate data is really about *data consistency*. For those applications that must have a version of the data

stored locally with the system, data integration techniques are the most optimal. As a result, SOI approaches may not be appropriate for those simple integration requirements.

Likewise, service-orientation by itself does little to integrate systems at the semantic or business process levels. While certainly the SOA architectural approach greatly facilitates the exchange of vital business process and semantic information, other, additional integration technologies are needed. Specifically, to create a business process composed of multiple orchestrated Services, users would need a separate tool for visual process definition and execution. Likewise, trying to integrate different representations of information at the semantic level requires a tool that is capable of understanding the specific context of a requested piece of information and making guesses as to how to translate it to into the required format.

So, the "sweet spot" for Service-Oriented Integration is squarely focused on the problem of integrating multiple systems at the business logic and programmatic interface level into a cohesive enterprise-wide solution geared at facilitating system change. This notion of creating *composite applications* through integration approaches is where companies can realize the most significant return on investment on SOI implementations today.

IV. Implementing Service-Oriented Integration Solutions

There are a number of major differences between tackling integration from a Service-oriented point of view and the traditional approach to EAI taken by existing vendors. Many of these differences represent the architectural change required by companies to implement integration solutions that both meet the requirements for composite application integration as well as SOAs. In particular, companies should create SOI solutions that:

- Reduce the cost and complexity of managing IT infrastructures by planning for ongoing change.
- Provide a consistent platform for both internal as well as external integration requirements.
- Move away from closed, brittle integration technologies and solutions.
- Enable application and data reuse.
- Provide fine-grained access to data, functionality, and logic as well as the ability to create coarse-grained access to composite application logic.

However, in order for integration to move from its perception as a consistent IT bottleneck to a motivator of business agility, it must evolve from an expensive, complex, custom solution to relatively inexpensive, pre-packaged products. While a number of traditional EAI and B2Bi vendors are evolving to support SOI, the requirements for SOI support is far from trivial. In particular, these vendors must meet the following requirements to have a viable SOI solution:

- *Apply SOA techniques to integration* – Instead of tightly-coupling systems through interfaces, hard-coded adapters and point-to-point integration approaches that "cast business processes in concrete," SOI solutions must approach integration through applying the SOA fundamentals of loose coupling and coarse granularity. In addition,

these systems must support the dynamic nature of Web Services by providing capabilities for test and configuration of SOI solutions.

- *Support for core Web Services standards* – Integration solutions must fully and aggressively support the Web Services standards, including the ability to generate and exchange SOAP-based messages, processing of WSDL documents, and easy connectivity to private and/or public UDDI registries or repositories.
- *Provision of Web Services-enabled adapters or connectors to data sources* – Rather than providing COM or EJB-only adapters to various data and application sources, integration solutions must provide full Web Services-enabled mechanisms for accessing and managing end data sources. Integration solutions must either provide new Web Services interfaces to systems that don't already have them, or provide support for integration with existing Web Services interfaces.
- *Support for Web Services security* – Since security is highly variable between Web Services implementations, it is the role of the integration middleware to normalize the security infrastructure. Therefore, integration solutions must support the exchange of secured SOAP messages over a range of protocols and standards. Integration solutions should also provide hooks to security safeguard systems such as policy management and authentication for the access and usage of Web Services.

While Web Services offer a standardized way to implement Service-Oriented Architectures, it is not absolutely imperative that Web Services be used in exclusion of other, more prevalent technologies that could participate equally well in an SOA. An SOA mandates an architectural approach that serves to make functionality loosely coupled, asynchronous, and coarse-grained. As such, it is quite possible to include COM, CORBA, EJB, and a wide variety of application and data sources into the mix. An SOI solution that aims to serve a wide customer base will equally be able to handle the requirements of standardized, Web Services-based SOAs as well as a wide range of interfaces that are widely available in the enterprise.

Most likely, enterprises with significant investments in existing EAI and B2Bi solutions will not be very eager to rip them out and replace them with new, unproven technologies. Therefore, it makes sense for these enterprises to look at incrementally adopting SOI techniques where their value can be most critically realized. In this manner, SOI adoption does not require wholesale replacement, but rather, encourages an organic, "creeping" approach to integration that serves to address the most pressing problems first and then work its way throughout the remainder of the organization.

Guidelines for Implementing SOI Solutions

Companies that seek to implement SOI solutions should keep the following guidelines in mind when they build integration solutions leveraging SOAs:

- Think beyond a single project integration need, while implementing integration projects in an incremental fashion. Plan for project-to-project reuse of Services and the incremental benefits of that reuse.
- SOI solutions shouldn't require the installation and configuration of a lot of infrastructure before your first integration project. Approach

integration projects in an incremental way – rather than a big rip and replace project.

- Leverage your existing skill sets and technologies.
- Think about the need for a registry or repository and how that helps in achieving the goals of SOI for dynamic management of Services for integration and reuse.
- Don't corner yourself into one particular implementation approach for SOI. Use a wide array of interfaces to access services. For example, use .NET or J2EE-based application server-based components, or message-oriented, document-style approaches, whatever flexibility is required of the integration task.
- Consider your legacy systems to be assets. Think about how to wrap those assets in services so that they can continuously be reused and leveraged. Replace only when significant ROI can be realized – and do so at a Service level.
- SOI is not suitable for all scenarios, so users should identify when this approach is appropriate. For data integration tasks, find the most appropriate pathway for integrating data, and then, if possible, expose that pathway as a service. Composite applications are at the sweet spot for SOI approaches, so think about how to build coarse-grained business applications from a constantly changing set of composite applications.

V. WRQ's Verastream: Meeting the Requirements for Service-Oriented Integration

Fortunately, there are a number of products on the market that can help meet the needs and demands of companies looking to solve their Service-Oriented Integration challenges. One of these solutions is WRQ's VeraStream product.

WRQ Verastream provides a secure, Service-oriented approach to integration that lets users integrate with and repurpose legacy functionality without disturbing valuable code and associated business processes. Verastream provides an advanced abstraction process designed to speed up host integration. It takes place in multiple layers: abstraction of communication details of the legacy application, abstraction of operational details of the legacy application via encapsulation of its functionality into Services, and abstraction of the programming details required to combine low-level tasks from legacy applications into high-level business functions, via Service composition. These layers facilitate the realization of the benefits of Service-Oriented Integration.

On top of their infrastructure for achieving these layers of abstraction, Verastream provides support for Java and .NET for implementation of services and legacy system wrappers. In order to access a wide variety of legacy, database, and enterprise application systems, Verastream provides adapters that automatically establish and maintain connections with the source application, as well as provide the technology abstraction layer, which makes the communication details of underlying systems appear the same regardless of the actual system being accessed – and there is no hard-coding of adapters. Using Verastream, enterprise developers can put multiple interfaces on the same Service without having to recreate the Service. In addition, the product provides a Web-application generator to automatically produce new Web-based applications based on those Services.

Verastream further facilitates these layers of abstraction by providing a Service encapsulation process that hides all the nuances of how the source application performs tasks, and isolates the required functionality from the rest of the source application so the target application is not constrained by the user interface or the flow of the source application. In addition, Verastream provides a point-and-click composition tool that combines low-level Services from one or many applications into higher-level Services that can provide more complete functionality. Finally, to ensure that developers can easily organize and manage Services, Verastream provides a Service repository. This directory catalogs Services in a variety of ways and enables searches to find the appropriate Service for a given project.

Underlying all of these services is Verastream's runtime server that provides a robust and scalable environment for deploying abstracted Services. Verastream runtime servers provide load balancing, failover support, scalability, and security. The product also includes a management console that provides a centralized interface to manage runtime servers, and lets administrators remotely configure, deploy, and monitor services.

VI. Conclusions

Integration has been a headache and challenge for most enterprises for decades. Traditional EAI and B2Bi solutions have offered some relief, but at great expense, complexity, and rigidity. Web Services offer a better path for integration through Service-Oriented Integration (SOI) techniques that provide a standards-based, loosely coupled, fine-grained approach to connecting systems.

The real costs in building and integrating such SOAs is in the system re-architecting. In an SOI solution, it is not sufficient to simply wrap an existing API with a Web Services interface. It is too simplistic to create a one-to-one mapping between system APIs and Web Services interfaces. Rather, businesses must spend time analyzing their business processes and creating business Services at varying levels of granularity, perhaps even requiring the orchestration and choreography of multiple layers of Web Services to accomplish a single task. This support for multiple levels of granularity enables the SOA to support frequent changes in the underlying systems, as well as changes to business processes and underlying business assumptions, without the need to make interface changes that break the loose coupling of the Services. The real win with SOI, therefore, is in not only in reducing the cost of integration, but also in facilitating *business agility*.

WRQ Profile

Profile: WRQ	(March 2003)
Date Founded: 1981 Funding: Privately-held CEO: Doug Walker Employees: 425 Address: 1500 Dexter Avenue North Seattle, WA 98109 USA URL: www.wrq.com Main Phone: +1.800.872.2829	

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

ZapThink is an IT market intelligence firm that provides trusted advice and critical insight into XML, Web Services, and Service Orientation. We provide our target audience of IT vendors, service providers and end-users a clear roadmap for standards-based, loosely coupled distributed computing – a vision of IT meeting the needs of the agile business.

ZapThink's role is to help companies understand these IT products and services in the context of SOAs and the vision of Service Orientation. ZapThink provides market intelligence to IT vendors who offer XML and Web Services-based products to help them understand their competitive landscape and how to communicate their value proposition to their customers within the context of Service Orientation, and lay out their product roadmaps for the coming wave of Service Orientation. ZapThink also provides implementation intelligence to IT users who are seeking guidance and clarity into how to assemble the available products and services into a coherent roadmap to Service Orientation. Finally, ZapThink provides demand intelligence to IT vendors and service providers who must understand the needs of IT users as they follow the roadmap to Service Orientation.

ZapThink's senior analysts are widely regarded as the "go to analysts" for XML, Web Services, and SOAs by vendors, end-users, and the press. They are in great demand as speakers, and have presented at conferences and industry events around the world. They are among the most quoted industry analysts in the IT industry.

ZapThink was founded in October 2000 and is headquartered in Waltham, Massachusetts. Its customers include Global 1000 firms, public sector organizations around the world, and many emerging businesses. ZapThink Analysts have years of experience in IT as well as research and analysis. Its analysts have previously been with such firms as IDC and ChannelWave, and have sat on the working group committees for standards bodies such as RosettaNet, UDDI, CPExchange, ebXML, EIDX, and CompTIA.

Call, email, or visit the ZapThink Web site to learn more about how ZapThink can help you to better understand how XML and Web Services impact your business or organization.

ZAPTHINK CONTACT:
ZapThink, LLC
11 Willow Street, Suite 200
Waltham, MA 02453
Phone: +1 (781) 207 0203
Fax: +1 (786) 524 3186
info@zapthink.com

ZapThink, LLC • 11 Willow Street, Suite 200 • Waltham, MA 02453 • info@zapthink.com • www.zapthink.com