Doc. ID: ZTR-WS103-1
Released: June 10, 2002

zapthink
foundation report

SERVICE-ORIENTED INTEGRATION

USING WEB SERVICES AND XML TO INTEGRATE SYSTEMS

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

SERVICE-ORIENTED INTEGRATION (SOI):

USING WEB SERVICES AND XML TO INTEGRATE SYSTEMS

June 2002

Analyst: Ronald Schmelzer

Abstract

Connecting systems both within the enterprise and with suppliers, partners, and customers is of critical importance to today's enterprise. However, integration remains complex, expensive, and risky. While Web Services won't be the magic bullet that immediately solves these problems, they enable a new approach to integration. Service-Oriented Integration (SOI) leverages open standards, loose coupling, and dynamic description and discovery capabilities of Web Services to reduce the complexity, cost, and risk of integration. This report identifies the key aspects of SOI, solutions for implementing SOI, ROI metrics, and critical challenges.

Key Points:

✦ Market Overview

- Service-Oriented Integration (SOI) simplifies system integration by providing a single, simple architectural framework based on Web Services in which to build, deploy, and manage application functionality.

✦ Facts & Figures

- The SOI market is expected to grow from $435 Million in 2001 to about $6.2 billion in 2006.
- The top three EAI vendors have over 43% of the overall EAI market. With the entrance of Microsoft and other vendors in 2002, this landscape is expected to change.

✦ Analysis

- Web Services are not themselves an integration technology, but a distributed computing technology that lends itself well to being used for integration.
- In a Web Services context, there really is no fundamental difference among EAI, B2Bi, and Data Integration.
- SOI solutions allow users to get a greater level of interaction and granularity with components deep within the application.

✦ Future Trends

- EAI and B2Bi vendors will have to evolve to an SOI model in order to remain competitive.
- SOI faces major challenges: immature specifications, insufficient reliability, security, and transaction control.

✦ Decision Points

- The potential ROI realized by adopting Service-Oriented Architectures (SOA) far outweighs the slight benefits an organization gets from using Web Services as simply a "better API" for accessing application functionality.
- Integrating systems between two businesses is not only a technological problem; it requires pre-existing business relationships between the companies, workflow, and other human involvement.

Table of Contents

I. Report Scope

Enterprises face the challenge of connecting arbitrary systems in a manner that is cost effective, manageable, efficient, and secure.

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 be appropriate for some time in the distant future, but right now, enterprises face the more immediate challenge of connecting arbitrary systems in a manner that is cost effective, manageable, efficient, and secure. Systems developers and administrators should be able to integrate any system of any type with any other arbitrary system as driven by short-term business requirements, rather than surmount the near-impossible challenge of managing disparate systems in the face of constantly changing business needs.

This problem represents today's challenge with business process, application, and system integration that emergent Web Services and XML technologies may solve. Instead of implementing discrete, distinct technologies for Enterprise Application Integration (EAI), Business-to-Business Integration (B2Bi), and Data Integration (DI), Web Services and the Service-Oriented Architecture (SOA) represent an approach for integrating systems using an abstracted methodology called Service-Oriented Integration (SOI).

This report addresses the specifics of how SOI technologies and approaches solve lingering integration issues, and illustrate the potential return-on-investment (ROI) that can be realized by implementing SOI approaches. In particular, this report covers:

- The key challenges facing enterprise and organizational integration
- The basic elements of Service-Oriented Integration
- How SOI approaches will supplant traditional EAI, B2Bi, and data integration solutions
- Drivers and motivators for SOI adoption
- Analysis of key barriers to SOI adoption
- Convergence and trends for vendors offering SOI solutions
- Market segmentation and sizing
- ROI analysis for implementation of various SOI solutions
- Key vendors and technologies offering SOI solutions

It is our hope that by reading this report, you will gain a fundamental understanding of how Service-Oriented Integration (SOI) can be successfully applied to solving the key integration problems faced by industries of all types and sizes.

II. Integration: The Challenges to be Solved

2.1 The N-Squared Integration Challenge

Integration is a critical challenge at the heart of most enterprises' key business systems, including Customer Relationship Management (CRM), Supply Chain Management (SCM), e-Business and e-Commerce systems, and enterprise portals. Companies experience integration problems once they company installs their second enterprise system. The need for these two systems to communicate forms the basis of the integration "problem."

Why is integration important to the enterprise? For the following major reasons:

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

- *Connect organizations* – Integration allows business units and third parties to interact as a connected entity, facilitating functions vital to the flow of commerce.
- *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.
- *Lower the cost of ownership* – Integration allows enterprises to lower their cost of ownership of applications by reducing system complexity, simplifying management, and reducing the cost of change.
- *Allow systems to evolve* – Integration allows for legacy systems to be replaced or retired 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.

The simplest integration methodology is "point-to-point" 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. The end result is a tangled web of point-to-point integrations that neither meets business requirements nor performs adequately.

Most current integration solutions encompass a total of less than six systems.

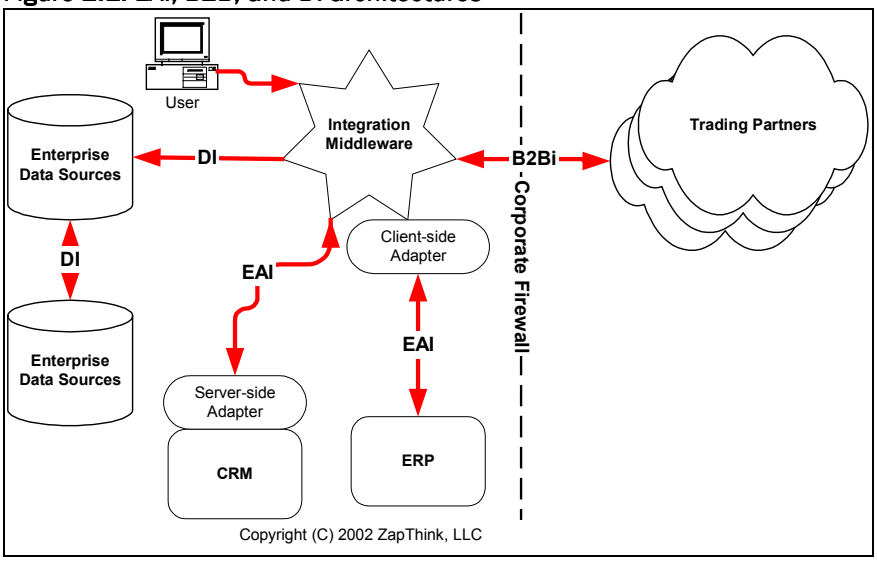
However, the increasing movement towards B2B systems and Service-Oriented Architectures (SOA) is driving a need to integrate with dozens, if not hundreds of systems in a single integration environment. As the number of connection points increase, so do the complexities and inefficiencies of data transformation, manipulation, and exchange. We are thus faced with an integration problem that grows at a rapidly increasing rate.

2.2 Classes of Integration Problem

Traditional integration approaches have aimed at solving three different types of inter-system communication challenge:

- *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, crossing the firewall to enable cross-enterprise business applications such as supply chain management (SCM), collaborative e-Commerce, and customer relationship management (CRM). The approach taken by these vendors is to electronically enable and automate business processes within the enterprise and across the "extended enterprise," including supply chain partners, distribution channels, customers, and third-party channels of all sorts.
- *Data Integration (DI)* – Integration solutions focused on connecting data storage and information representation systems for the purpose of unifying data sources and providing a "virtual" DBMS environment.

Figure 2.1: EAI, B2B, and DI architectures



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 DI 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 and reliability* – Integration solutions offer better scalability and performance than ad-hoc integration approaches.
- *Maintain legacy systems* – Integration solutions allow legacy systems to remain operational in the enterprise even after they cease being part of the daily business operations.
- *Reduce time to market* – Integration solutions allow businesses to shorten the time it takes to introduce new products or services.

2.3 Traditional Integration Solutions

In the past, integration between 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.

EAI and B2Bi solutions typically have robust transaction control, workflow, and business process management features in order to enable the many inter-dependent and necessary steps to accomplish any specific enterprise task. The workflow control features are leverage message-oriented communication to enable asynchronous system integration and a more scalable architecture. These various messages, in the context of a larger workflow, are built on top of and integrated with messaging middleware systems such as Message-Oriented Middleware (MOM), RPC systems, or distributed TP monitors.

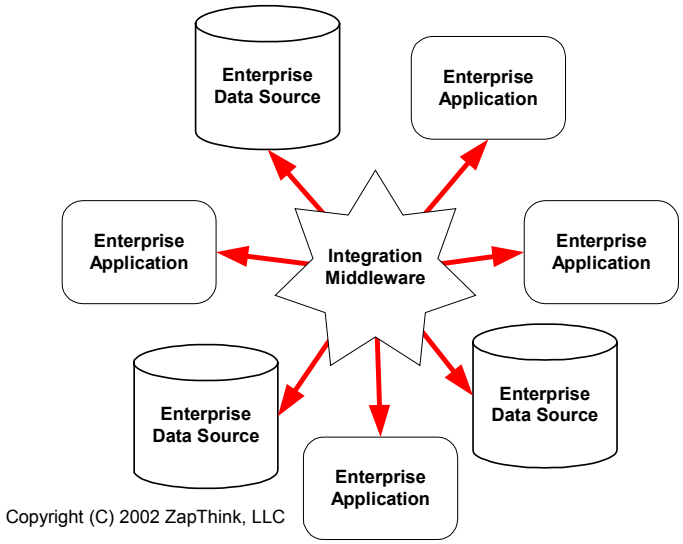
There are three main components to a traditional EAI solution:

- *Message Transport* – Typically implemented as Message-Oriented Middleware (MOM), EAI systems use messages to exchange application logic and data information between systems. Most implementations use asynchronous message queuing technology that guarantees delivery of messages and helps to loosely couple implementation from interface – at least to some degree.
- *Message Transformation and Routing* – EAI solutions function as "bridges," mapping the interfaces of one system to another. Transformation functionality helps to keep implementations separate while allowing them to understand each other. Routing functionality helps these systems locate and communicate with each other.
- *Business Process and Rule Integration* — Allows multi-step and long-lived processes to occur by automating the flow and distribution of messages to appropriate systems and managing process events.

2.3.1 Current Architectures for Integration: Hub and Spoke Topology

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").

Figure 2.2: Hub-and-Spoke Topology

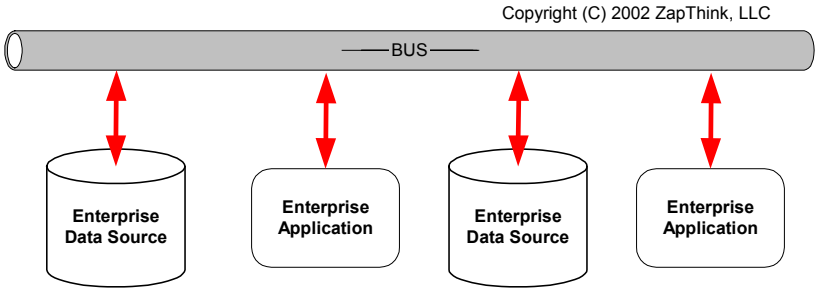


Implementation of the hub-and-spoke architecture, as illustrated above in Figure 2.2, is not particularly efficient or scalable is often the root cause for many failed EAI solutions. Hub-and-spoke solutions are resource-constrained, since all processing occurs on a single integration server. At some point, the amount of traffic and number of connections will saturate the available resources of the integration server, causing the system to perform poorly. The integration middleware hub is also a single point of failure, which limits the architecture's robustness.

2.3.2 Current Architectures for Integration: Bus Topology

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 to be integrated, and share their information on a common "bus".

Figure 2.3: Bus Topology



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

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. Rather than relying on a central hub, the bus architecture distributes message transport, transformation and routing, and business process logic across the application adapters.

2.3.3 Current Architectures for Data Integration: "Screen-Scraping"

At the lowest and most primitive levels, users can access legacy application code through "screen scraping," which has been one of the most popular methods for gaining access to data and application logic located on mainframe and other closed legacy systems. Rather than having to gain access to a programmatic interface, users can simulate user interaction by means of *terminal emulation*, mimicking keystrokes and screen navigation to get to key pieces of information that they can parse directly from screen outputs. Screen scraping is useful when it is difficult or impossible to modify actual application code.

Of course, screen scraping is a very crude way to get at information, but a surprisingly large number of mission-critical applications use screen scraping in the banking, travel reservation, insurance, and other industries. Since users can only retrieve data through screen navigation and keystroke emulation, this method of integration is by necessity synchronous, since the application must be online in order for systems to interact with it. Also, screen scraping is a very "brittle" form of integration since any modification to screen outputs will corrupt the data gathering process. Thus, screen scraping is not only synchronous, but also very tightly coupled.

2.3.4 Current Architectures for Data Integration: Extract-Transform-Load (ETL)

A better 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. 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 is already "stale."

In addition to ETL techniques, data integration can also happen through real-time or near real-time data replication and synchronization techniques that utilize triggers to automatically extract and transform data between sources. Another method creates a "virtual" database from multiple sources, allowing a single database middleware product to span multiple data sources, thereby integrating them.

All the above data transformation techniques work well when a user has access to the data sources and the user is capable of gaining valuable information from just the data sources. However, there is tremendous value in application logic that can make sense of often confusing data structures. Many times, ETL and replication techniques are simply not usable for mission-critical integration needs. Furthermore, data integration techniques can disrupt the operation of applications that require database locking. Finally, these data integration methods can become complicated and impractical in a multi-vendor scenario.

2.4 Why Current EAI and B2Bi Solutions are Not Sufficient

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. The challenges of integration combined with the opportunities posed by XML and the Internet have combined to provide a new class of solution for integration: *Service-Oriented Integration*.

Until very recently, computer functionality resided on individual computers. Developers programmed computer applications as discrete chunks of computer code that ran on these distinct, individual computers. While object-oriented programming methodologies aimed to abstract much of this functionality, the technology itself originated in a single-computer environment, thus being an outgrowth of the isolated computer mindset.

The first popular communications protocols such as UNIX's Network File System (NFS) or Microsoft's Distributed Computing Environment (DCE) protocols focused on the lower layers of this network stack. Once an established set of communications protocols existed, vendors turned their sights on higher level protocols for distributed computing, in particular the Object Remote Procedure Call (ORPC) protocol for Microsoft's DCOM and the OMG's Internet Inter-ORB Protocol (IIOP) that forms the basis for CORBA. These systems depended heavily on object-oriented techniques to accomplish their goals, since functionality for handling communications could be easily and separately encapsulated, and the objects themselves maintain their own state and information identities.

There are many problems with this object-oriented, RPC-style approach to distributed computing. First, the mindset is still focused on building code that exists or "virtually exists" on the local computer. The result is a somewhat complex and inefficient system for packaging (or "marshaling") applications on a system and shipping it to another system. RPC-based mechanisms have typically faced challenges resulting from their requirements for matched system architectures, byte formats, and other strict computing requirements. These challenges spurred development of object request brokers (ORBs) and interoperability formats, which themselves have faced an uphill adoption battle.

In addition, these different integration approaches are typically proprietary to various platforms on which they reside. DCOM is a Microsoft-only architecture, and CORBA's goal of providing cross-platform interoperability is foiled by its complexity and semantic ambiguity. CORBA and DCOM also differ in their formats -- DCOM uses the Network Data Representation (NDR) format for payload parameter values and OBJREFs for endpoint naming while CORBA relies on the different, and thus incompatible, Common Data Representation (CDR) format and Interoperable Object References (IOR) for endpoint naming. Out-of-the-box DCOM and IIOP interoperability is not a practical reality, and is especially severe when attempting to integrate machines across the Internet. Since the launch of CORBA 2.0 in 1995, many gateways were produced to enable IIOP tunneling over HTTP, but the end results have been far from successful.

Many EAI solutions are focused on data level integration, rather than application or service-level integration. As a result, 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, they are rarely 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 improvement are more important to the product's success than are more efficient means for pushing information through the application or connecting to large numbers of systems. Dependence on the traditional hub-and-spoke architecture is partly to blame for these efficiency and scalability issues. Message brokers, which are the most popular form of EAI technology, are based on the requirement that information is processed through a central hub, following the basic hub-and-spoke topology. Message brokers must chop up large chunks of data into multiple messages, and many such large queries can easily overwhelm messaging systems.

Finally, these methods are inappropriate for integrating systems across the firewall, due to their complexity and security concerns. Since their formats use binary wire protocols, IIOP in the case of CORBA and ORPC in the case of DCOM, they don't easily pass through firewalls that require the formats to be easily inspected (and thus text-based). Therefore, enterprises have adopted DCOM and CORBA have most commonly within the corporate walls, rather than integrated with other firms and systems outside the firewall.

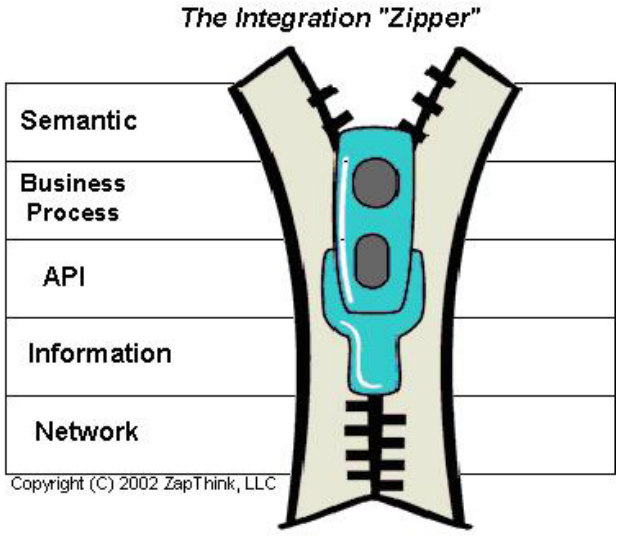
While the Web certainly galvanized a new approach towards application development and delivery, Web-based application delivery is simply too lightweight and insufficient to handle the robust requirements of distributed computing. Rather than being too tightly linked to a development platform, the Web application approach suffers from the opposite problem – the applications are too loosely coupled from systems to provide effective integration. HTTP and HTML are oriented towards stateless, user-oriented interactions, rather than transaction-based, machine-oriented interactions. What Web technology clearly lacked was a melding of the lightweight, platform-agnostic capabilities of the Web, with the richer functionality, application integration, and security capabilities of existing distributed computing technologies.

2.5 The Integration "Zipper"

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 occurring in a zipper-type format, where lower level integration problems must be solved before higher-level issues can be addressed. The layers of the Integration "Zipper" thus are:

- *Network* – integration at the "wire" protocol level
- *Information* – integration at the data tier
- *Application Interface (API)* – Integration at the application interface level
- *Business process* – integration at the application workflow, orchestration, and choreography layers
- *Semantic* – integration at the concept layer

Figure 2.4: The Integration "Zipper"



Integration problems never go away. Rather, as standardization and commoditization of technologies solve the problems on the lower levels of the integration stack, higher-level integration issues remain. These issues connect 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 will 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.

Currently, Web Services and SOI are focused on solving API and Business Process-level integration issues and doesn't deal with semantic or information-meaning level integration.

Currently, Web Services and SOI are focused on solving API and Business Process-level integration issues and do not deal with semantic or information-meaning level integration. ZapThink expects that these issues will be solved in the distant future (2007 or further) if at all.

III. Service-Oriented Integration Approaches

3.1 Using Web Services for Integration: Service-Oriented Integration (SOI)

The vision of using Web Services, or any Service-Oriented Architecture, 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 with which systems can interact. 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.

In a Web Services context, there really is no difference between EAI, B2Bi, and Data Integration.

In a Web Services context, there really is no difference between EAI, B2Bi, and Data Integration. Rather, we can take the same technological approach (namely Web Services) and just apply it in different ways to solve these classes of integration problems. Of course, Web Services aren't the solution to these integration problems; they just provide a technology on which to produce solutions.

Most existing EAI, B2Bi, and Data Integration solutions involve the connecting of different applications and data stores to each other in a point-to-point approach. The Service-oriented vision changes that. 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 bus or web of Services.

Web Services aren't an integration technology at all, but just a distributed computing technology that lends itself well to being used in integration scenarios.

Another way of thinking of the difference between existing EAI applications and SOI is that traditionally EAI has taken the approach of requiring explicit connections to an integration hub or bus in order to achieve a "deeper" integration, while SOI advocates keeping all integrations loosely coupled. As a result, some say that Web Services are just an *ad hoc* way of doing integration, and that this is somehow a negative knock on the technology. From a Services point of view, certainly, the exposure of component functionality as dynamically bindable, discoverable objects can be considered ad-hoc integration.

However, this dynamic binding ability may be a desired goal. Rather than planning in advance how a specific application will tie into another applications, developers should be thinking about how a specific application exposes itself to any application that cares to speak to it. The point of integration is to allow arbitrary applications, systems, and data stores to communicate without concern as to the other system's requirements.

A comparison of the different integration approaches -- ad-hoc integration, Data-level integration, EAI/B2Bi integration, and Service-Oriented Integration – can be seen below.

Table 3.1: Comparison of Integration Approaches

Integration Approach	Advantages	Disadvantages
Ad-hoc integration	➢ Simple to implement ➢ Can use existing tools ➢ Short time-to-market	➢ Doesn't scale ➢ Management nightmare ➢ Difficult to add systems ➢ Very tightly coupled
Data-level integration	➢ Easy to implement ➢ Little change to existing application logic ➢ Use some existing tools	➢ Does not allow addition of new functionality ➢ Synchronous only ➢ No addition of new business logic ➢ Hard to integrate multiple data streams ➢ Tightly coupled
EAI / B2Bi Integration	➢ Asynchronous model ➢ Integrate business logic ➢ Rules-based transformation ➢ Hub-and-spoke or bus model	➢ High TCO ➢ Proprietary ➢ Tightly coupled ➢ Problem in synchronous models ➢ Scalability ➢ Complex to manage

Service-Oriented Integration	➢ Standards-based ➢ Loosely coupled ➢ Asynchronous and synchronous ➢ Easy to add new logic ➢ Lower cost than EAI ➢ Business agility ➢ Multiple models	➢ Standards immature ➢ Missing robust security, reliability, transaction support ➢ Needs new tool sets

Source: ZapThink, LLC

3.2 Methods for Implementing SOI

There are predictably many entrants into this new and emerging space of Service-Oriented Integration. What differentiates the various players is the depth and breadth of their strategy, the current availability of their tools, and their ability to expose legacy and existing system functionality as Web Services. In general, SOI solutions fall into the following categories:

- Integration-enabled Web Services Platforms
- SOI-enabled EAI and B2Bi Solutions
- Emerging SOI Solutions
- Data-focused SOI Solutions

Each of these solutions offer a different approach towards Service-Oriented Integration, and while some are quite early in their product development cycle, we can expect these vendors to become more "complete" in their product definition over the next few years.

The SOI vendors are identified in the table below by solution segment.

Table 3.2: Market Segmentation of Service-Oriented Integration Solutions

Market Segment	Solutions
Integration-enabled Web Services Platforms	BEA IBM WebSphere and Xperanto Microsoft BizTalk
SOI-enabled EAI and B2Bi Solutions	Peregrine SEAGULL SeeBeyond Sybase (Formerly NEON) TIBCO Vitria WebMethods
Emerging SOI Solutions	Actional Attunity Cape Clear Infoteria IONA Software AG EntireX
Data-focused SOI Solutions	Coherity iWay Nimble Technology XAware

Source: ZapThink, LLC

3.2.1 Integration-enabled Web Services Platforms

Web Services can be crafted easily from existing legacy application components, such as existing COM and J2EE applications. The addition of Internet-aware components and the development of open, non-proprietary systems such as XML, Linux, and Java have paved the way to relatively easy adoption of Web services technologies. As a result, integration can be accomplished by simply creating and exposing Web Services from existing objects and interfaces.

Web Services Platforms allow users to create Web Services, either from scratch or by wrapping existing functionality such as EJB or COM components or other legacy functionality. Various additions to existing object platforms, such as the Model-View-Controller (MVC) architecture in J2EE and Microsoft's COM+ platform, have simplified the process of moving to a service-centric computing model.

Vendor Focus

BEA
IBM
Microsoft
Oracle
Sun

The major platform vendors (**Microsoft, IBM, Oracle,** and **Sun**) as well as **BEA** will be the most likely solution vendors to add integration capabilities to their Web Services development and runtime environments in a credible and sustainable manner. Most notably, Microsoft and IBM have made strong entries into this space that will be a challenge for other providers of SOI solutions. Microsoft has built its own integration technology, namely BizTalk, whereas IBM has acquired much of their functionality when they purchased CrossWorlds.

Decision Point

Microsoft and IBM have made strong entries into this space that will be a challenge for other providers of SOI solutions.

While Microsoft is definitely casting most its chips in favor of their new Web Services-oriented strategy, its approach has it traveling in a separate direction than its competitors. While surely .NET services will be interoperable with non-.NET created Web Services, they will only be able to be run on the .NET platform. This more general portability issue may cause concern with many developers who will surely be looking to leverage their skills more widely. More relevant to our discussion here of integration technology, Microsoft's BizTalk product is aimed squarely at enabling cross-platform integration using XML and Web Services technology.

All of the other vendors named (BEA, IBM, Oracle, and Sun) are producing J2EE-based integration solutions, with IBM notably in the lead with its recently re-branded WebSphere environment that consists of an integration server in addition to its popular Application Server. BEA is following suit with a similar integration server

3.3 SOI-enabled EAI and B2Bi Solutions

The emergence of Web Services doesn't represent a threat to the existence of traditional EAI and B2Bi vendors, but rather a new opportunity for them to produce product and services. These vendors have been living the integration challenge for years, if not decades. While it is true that many of the vendors bring old-world, rigid concepts that need to be realigned in the face of Service-Oriented thinking, these vendors have time-tested functionality for creating, testing, deploying, publishing, and managing Web Services within an enterprise as well as between businesses.

The integration middleware vendor plays a valuable role that the Web Services Platforms cannot provide the integration of multiple disparate technologies without prejudice or bias as to the native platform on which those technologies are executed.

However, in order for SOI to become a viable technology solution to most businesses, it needs to evolve from an expensive, complex, custom solution to a pre-packaged, out-of-the-box product. As a result, custom coding, even through the use of Web Services platforms mentioned above, is not a viable solution. The integration middleware vendor plays a valuable role that the Web Services platforms cannot provide: the integration of multiple disparate technologies

without prejudice or bias as to the native platform on which those technologies are executed. Integration middleware has traditionally integrated a wide range of technologies and provided a common framework on which companies can achieve reliable enterprise computing.

A number of the traditional EAI and B2Bi vendors are evolving to support SOI. This support is far from trivial, however, as these vendors must meet the following requirements to have a viable SOI solution:

- *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.
- *Support development of Web Services from at least a "wrapping" perspective* – Integration brokers 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. As a result, these solutions must support simplified Web Services development, deployment, and publishing. In addition, these systems must support the dynamic nature of Web Services by providing capabilities for test and configuration of Web Services solutions.
- *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 brokers must provide full Web Services-enabled mechanisms for accessing and managing end data sources.
- *Management of 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 need to support the exchange of secured SOAP messages over a range of protocols including HTTP, HTTPS, SMTP, and FTP. In addition, communications with UDDI registries should be secure, and the solutions should handle all SOAP communications in a secure manner. Integration solutions should also provide security safeguards such as policy management and authentication for the access and usage of Web Services, as well as "normalization" of identity and policy between different systems with which they integrate. See the upcoming ZapThink Report *XML and Web Services Security* (ZTR-WS104) for more information.
- *Provide Workflow, Transactional Integrity, Auditing, and Monitoring Control to Web Services* – Integration solutions must be able to support robust and reliable Web Services exchanges between systems. As a result, they must support long-lived transactions, an absolute necessity in mission-critical operations. Multi-step processes should be managed through workflow management solutions. The solutions also need effective audit mechanisms through which companies can closely monitor the quality of service, health, access, and usage of Web Services.

Decision Point

Enterprises with significant investments in existing EAI and B2Bi solutions should encourage their solution providers to support SOI requirements more aggressively.

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 encourage the current integration vendors to adopt Web Services within their own product lines. Enterprises with significant investments in existing EAI and B2Bi solutions should encourage their solution providers to support SOI requirements more aggressively.

ZapThink believes that the effect of the combination of old and new integration styles is resulting in a hybrid model for Web Services. Simply using SOAP as a communication protocol does not enable companies to take full advantage of the Service-Oriented Architecture. Current EAI and B2Bi vendors are stuck implementing the "first-generation" of Web Services. In addition, the utilization of Web Services by current integration vendors is not necessarily resulting in a benefit by end users today. In order for SOI solutions offered by current EAI and B2Bi vendors to be useful, they must result in reduced cost of integration as well as complexity.

Current EAI and B2Bi vendors are stuck on implementing the "first-generation" of Web Services.

Right now, the Web Services solutions provided by EAI vendors offer little more than one more integration interface for their platform. Many of these systems use their SOAP interface primarily for synchronous, remote procedure call (RPC) applications. These EAI and B2Bi vendors must move to a more SOI-based approach in order to remain competitive and viable in a Services-oriented world. However, this first generation of Web Services support is extremely important. It introduces the first "crack" in the model of how systems are currently integrated—using proprietary, tightly-coupled, inefficient wire-level protocols and object component technologies.

Some of the moves that current EAI and B2Bi vendors should be making in order to make SOI more than an extension of existing, tired integration technology include:

- *Movement to a distributed (SOA) architecture*. Many existing EAI vendors provide hub-and-spoke or bus architectures, which are not standards-based, loosely coupled architectures that are self-described, published, and dynamically bound. Existing integration approaches need to move to SOA architectures to take full advantage of Web Services for integration.
- *Sophisticated support for asynchronous processes*. Instead of the traditional store and forward approach to asynchronous messaging, SOI-enabled EAI vendors must provide solutions that can enable high-volume messaging infrastructures to process incomplete pieces of data that are received out of sequence.
- *Robust Business Process Management (BPM).* EAI vendors must provide visual BPM tools to create and manage business processes on a high level, optimizing the use of the available underlying systems.
- *Broad transaction management.* SOI-enabled EAI vendors must be able to support long-lived and asynchronous transactions as well as more short-term, synchronous forms of transaction management. The systems should enable simultaneous collaboration and parallel processing services, in addition to sophisticated transaction management capabilities that assure ACID (atomicity, consistency, isolation, and durability) compliance for all transactions, including those involving multiple heterogeneous systems and asynchronous processes.

3.4 Emerging SOI Solutions

In addition to the platform and EAI vendors who are moving their existing legacy and hybrid solutions to the Service-Oriented Integration world, a number of established and start-up vendors are looking to provide a pure XML and Web Services-based approach to integration. These vendors seek to apply their technology across multiple platforms and application servers in order to provide a cross-vendor integration middleware approach that leverages SOI techniques.

Many of these vendors seek to be the Web Services equivalent of the integration broker. As a result, they provide:

- Tools and infrastructure to generate Web service interfaces to enterprise applications and middleware.
- Service registry, security, reliable transport, service configuration, process monitoring, and other enterprise quality of service technologies.
- Tools to develop automated business processes using Web Services, RosettaNet, ebXML, and other XML standards, while also interoperating with binary protocols such as IIOP, RMI, and JMS.
- Tools to manage interactions between Web Services.

One of the difficult problems in the middleware arena is that products in this segment are bifurcated into two different sorts of applications: those that are very good at integration, such as integration brokers, and those that are very good at building business logic systems. However, rarely are there technologies and products that offer both—a user has to choose either an integration or application server. These SOI-based tool vendor hope to change that trend.

ZapThink believes that pure SOI-based solutions will have to provide significant value beyond what Web Services platform and SOI-enabled EAI and B2Bi vendors are offering in order to be competitively viable.

ZapThink believes that these SOI-based solutions will have to provide significant value beyond what Web Services platform and SOI-enabled EAI and B2Bi vendors are offering in order to be competitively viable. In particular, they can be competitive by focusing on the higher layers of the integration stack (business process, workflow, and semantics) rather than on the information or adapter layers.

3.5 Data-focused SOI Solutions

At a lower level in the integration "zipper" exist vendors whose primary goal is not integration at the API, business process, or semantic levels, but rather the integration at the information or data level. This information-level integration is notably much simpler to achieve than more logically and semantically challenged application and business integration technologies. However, the goal is somewhat different. Rather than trying to tie together applications for the purpose of creating an aggregated application to access, data integration vendors seek to tie together disparate data sources for enabling a single "view" across multiple data sources. In effect, the data integration vendors provide a virtual database that can be queried, viewed, and modified.

At this latter layer, XML-based integration has plenty to offer. Namely, XML can be used as an "equalizing" data structure that helps to unify the different data structures and types between systems and provide a cohesive view of the aggregate data. XML provides a simple and robust way to describe complex datasets that can be extended over time.

Efforts to XML-enable the enterprise have been slowed by efforts to extract and manipulate information from multiple, disparate data sources.

Information grows organically, especially as different branches of an organization buy and implement software, but management needs are centralized. This conflict between management needs and operational reality is a problem. Historically, the only way to solve this problem has been through custom coding, data warehousing, or use of central repositories. While there has been an increasing trend in the market towards the use of XML for data storage and exchange, the vast majority of information in an enterprise currently resides in a wide assortment of data formats—most of them not XML or even structured. As a result, efforts to XML-enable the enterprise have been slowed by efforts to extract and manipulate data from these sources. In addition, companies have long sought after the goal of aggregating data from disparate data sources and providing a single interface for querying and retrieval of results.

These two goals have merged in the concept of data integration through the Virtual XML Database – a means by which disparate data sources can have a unified XML-based front end that performs query operations and merges results from a variety of different data sources. Applications that must cut across disparate, heterogeneous data sources can find information independent of their underlying data structure.

Virtual XML Databases don't actually store data, but rather integrate different enterprise data sources with any application that consumes XML. In order to perform this functionality, these systems must handle data aggregation, synchronization, bi-directional transport, data chaining (retrieving data from one system and use that as key to access related data from other system), decomposition of data, and workflow control.

These systems work by connecting to many different data sources, including RDBMS systems, Content Management systems, file systems, and applications of different types, and normalizing or abstracting the data into a single view with XML. Vendors can perform these multi-source "joins" synchronously or asynchronously, allowing one-to-many, many-to-one, and many-to-many views of data. The vendors have service-enabled their sources by creating a WSDL view of the query and registering it in a UDDI registry.

The main challenge with these data integration vendors is that they are only as efficient as their least efficient data source. Some vendors are looking to use native XML data stores as a way of improving data query speed by caching XML data. However, in the end, these systems are only as fast as the slowest system they are accessing.

The data integration approach can be compared to other approaches, including XML-native data stores and Enterprise Application Integration (EAI) solutions. However, EAI usually consists of message-oriented middleware and transactional integration that pieces applications together so that they can talk to each other. While EAI attempts to integrate applications, it doesn't solve the basic problem of providing common data from heterogeneous data sources to an application. Users can get better optimization by using a query language instead of custom integration code. Other competitors also provide different approaches, such as data warehousing, custom code, Extraction, Translation, and Loading (ETL), or even SQL-based integration. For more information on XML-native data stores, please see the ZapThink report *XML Data Storage Technologies and Trends* (ZTR-ST101).

IV. Drivers for SOI Adoption

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 real improvements over how integration was performed prior to SOI solutions. In particular, some of the key benefits of implementing SOI technology for integration include:

- Reducing the cost and complexity of managing IT infrastructures
- Providing a uniform platform for B2B exchanges and eMarketplaces
- Moving away from proprietary technologies and solutions
- Enabling application and data reuse
- Simplifying business modeling
- Providing fine-grained access to data, functionality, and logic

4.1 Reducing the cost and complexity of managing IT infrastructures

Throughout the evolution of computing, layers upon layers of applications, custom code, and middleware were required to achieve the necessary levels of integration, management, and application functionality. Soon, it became more expensive to replace the technology than it was to simply add more code to "patch" any holes or malfunctioning functionality. The end result was millions of lines of spaghetti code, intricately linked, and all absolutely necessary.

Object-oriented programming promised to solve these problems by introducing notions of modularity and reusability that would solve the code management nightmare. Unfortunately, few of the promised gains of OO technology came to pass, mainly due to the complexities inherent in distributed systems.

A number of wire-level networking protocols sought to further simplify the IT computing mess by injecting distributed computing frameworks for providing Remote Procedure Call (RPC) and message-oriented architectures. Microsoft's DCOM and the competitive CORBA technology were the emergent leaders in this trend, but their solutions were still too complex to gain widespread adoption in enterprises looking to integrate heterogeneous systems.

SOI simplifies system integration by providing a single, simple architectural framework in which to build, deploy, and manage application functionality.

SOI simplifies system integration by providing a single, simple architectural framework in which to build, deploy, and manage application functionality. Rather than building increasingly larger sets of application functionality, users can build small, finely grained application code that can be published, dynamically located, and bound to without a significantly complicated OO or messaging solution. With simplified components comes reduced cost. Web Services allow users to build complex systems out of very simple components without sacrificing power. This is the key vision that the Service-Oriented Architecture (SOA) brings us.

The proliferation of tools and the fact that SOI solutions are based on open standards means that SOI solutions can be simpler to design, implement, and maintain. Rather than having to learn the intricacies of different systems, communications technologies, and platforms, users can implement a more widely understood SOI methodology for integration.

As a result of their simplicity and reliance on open standards, SOI solutions can be less expensive to implement than traditional EAI and B2Bi solutions.

As a result of their simplicity and reliance on open standards, SOI solutions can be less expensive to implement than traditional EAI and B2Bi solutions. The ROI section below discusses this in detail, but clearly one of the motivations towards moving to a service-oriented approach towards integration is that its simplicity and use of standards will reduce initial and long-term cost of integration. The number of pre-packaged solutions available to developers of Web Services will no doubt increase dramatically, as will the number of skilled developers. Part of this reduction in cost is that Web Services, upon which SOI is based, can be implemented using relatively simple technologies and languages, including scripting languages.

4.2 Providing a uniform platform for B2B exchanges and eMarketplaces

The primary means for integrating different businesses and organizations throughout the 1980's and 1990's was (and still is) Electronic Data Interchange (EDI), a relatively arcane and aging data format and networking specification for B2B integration. While EDI has gained widespread acceptance, it is a fairly rigid language that accepts little modification to its data format. In addition, despite how rigidly EDI defines documents, there is still a large degree of discretionary

changes and variability that leads to semantic ambiguity when implementing EDI. Users must resolve these ambiguities in a tedious, manual manner, which can be quite cumbersome to the party that must deal with dozens, hundreds, or even thousands of trading partners.

The Internet promised not only to simplify dealing with multiple systems, but also to reduce the cost of connecting to various business endpoints. The primary challenge in working with various business and industry endpoints is that each business and industry has different data requirements, and even worse, defines the same things in different ways. These semantic issues were made significantly worse by the use of different, and sometimes arbitrary, data formats.

XML tackled this "Tower of Babel" problem, since XML's extensible data format provided the basis for building different industry and organization vocabularies. However, XML's greatest strength is also its greatest weakness, because its extensibility leads to the creation of hundreds of different business vocabularies, often with overlapping applicability, creating its own Tower of Babel. Rather than fully solve the semantic problem, XML merely provided a new medium onto which companies could move their semantic ambiguities. What was clearly needed is some agreement among businesses as to how to represent key pieces of information. However, it is clearly wishful thinking that businesses of different types will agree on a single, coherent nomenclature for naming critical data entities.

The Web Services model helps to provide a better answer to this semantic problem by providing a means for Services to describe themselves in a dynamic manner. It no longer becomes necessary for everyone to "talk the same language." Rather, we can use Web Services as a way of "normalizing" the conversation and allowing each party to dynamically discover the capabilities and requirements of the other party. While this approach is still a far cry from the vision of global, seamless, automated e-Business, at least business systems are able to locate and transact with other companies' systems in an automated and electronically enabled fashion .

4.3 Moving away from proprietary technologies and solutions

An interesting trend over the past decade is the movement away from single-vendor, proprietary systems toward "open" systems consisting of technologies that are not owned or managed by any one particular vendor. Key technologies that illustrate this point are Java, XML, and Linux. However, what is the importance of being "open" and how does that help the corporate IT environment?

The definition of "openness" we seek here is not in the definition of the specification, but how companies use it. A truly open technology can be created by Corporation A's tools and processed either by Corporation B, C, or D's tools, by open-source applications and tools, or be created and processed by any combination of different tools and applications by competing tools vendors.

For vendors of software applications who use "open" XML protocols and standards, this flexibility means that users can replace their software with a competitor's.. This ability is primarily an advantage to the consumer who has increased choice in solution provider. But this ability is also an advantage for the software vendor, because they can develop open interfaces that always keep their software applications current and open for modification. In any case, no company can do everything well. Great server vendors may be mediocre players in the middleware, desktop, and editor environments. The adoption of open

standards allow them to "play well" with others in the space and focus their own efforts on their core competencies.

Web Services are clearly the efforts of a combination of vendor-driven initiatives and open standards, which could be the balance that the market needs. To be sure, the Web Services standards themselves are open, but many individual implementations of Web Services are certain to be commercial products, pitched and delivered by both single vendors and the open source community alike.

Service-Oriented Integration depends on the usage of Web Services for integration, and as such, leverages open standards including XML, HTTP, SOAP, WSDL, and other key non-proprietary, non-vendor specific formats. Most EAI solutions, on the other hand, involve some level of lock-in to some vendor-specific software or communications technology. The use of open standards allows organizations to have the widest possible selection of options and alternatives for SOI solutions.

The use of open standards allows organizations to have the widest possible selection of options and alternatives for SOI solutions.

4.4 Enable application and data reuse

Reducing the cost of IT management is one of the primary pressures for most organizations. One of the most common ways to reduce such costs is by enabling the reuse of applications that developers have already created and configured for the enterprise. In the past decade, especially in the past 3-5 years, millions of dollars have been spent by companies on enterprise software applications of all sorts: CRM, ERP, and other operational applications. The next few years will be less about new application development, and more about existing application integration and reuse.

Object component technology has not provided the desired level of reusability since object components' static nature is incompatible with the constantly changing field on which to develop applications. Object components that may have been perfectly acceptable one quarter may be irrelevant the next. This constant flux creates a poor environment for object components.

The Service-oriented approach helps solve the challenge of reuse by imposing a design methodology that promotes the use of self-describing, published, loosely coupled, and dynamically bound components rather than static, tightly-coupled components. Reuse becomes a matter of publishing available Web Services and developing the Services themselves to make sure they are not inadvertently tightly coupled.

The Service-oriented nature of Web Services and SOI allows it to retain a level of flexibility and "future-proofing" that is simply not capable with point-to-point EAI solutions. By allowing components to be self-described and dynamically bound, developers can eliminate the complexity and rigidity of establishing point-to-point integration connections. Also, Web Services offer dynamic interfaces to systems and applications, adding a considerable amount of flexibility beyond traditional, static EAI interfaces.

4.5 Simplifying business modeling

Businesses must model the various constituent parts of the enterprise and how they interact in order to gain a fundamental understanding of them. In many ways, business modeling is taking design methodologies appropriate for object-oriented computing and applying them to various business functions in a recursive manner. Coarse-grained business processes consist of business components that contain the more fine grained software objects.

Business modeling achieves a number of objectives:

- Helping to identify the level of granularity of systems for reuse.
- Providing requirements for business logic components and subsystems, and their interaction with other subsystems and external businesses.
- Identifying areas where components can be combined or separated for greater flexibility and reuse.
- Developing systems than can evolve over time.

However, to meet these goals, businesses must emerge from modeling components using complex, nonstandard interfaces. Modeling in a service-oriented architecture would enable reuse and the other goals mentioned above. Just like Web Services themselves, business models shouldn't make any assumptions of the underlying architecture or framework on which the solutions must be developed. Rather, the enterprise should model the components and their interactions sufficient to enumerate the requirements, but without imposing any restrictions on how those requirements are to be met. This level of abstraction is very much the same as with the SOA: allow services to fulfill needs while abstracting the actual way in which businesses implement them. Without this approach, businesses must build components in a custom fashion, making each component unique and non-reusable.

Decision Point

Business modeling and identifying the granularity of Web Services will become as important as the business logic contained within the Web Services themselves.

ZapThink believes that what will become increasingly important is not the middleware platform itself, but the thought processes that go into to deciding how to create Web Services and SOI solutions. Business modeling and identifying the granularity of Web Services will become as important as the business logic contained within the Web Services themselves, since Web Services components that are too coarse-grained will be just as difficult to reuse as tightly coupled object components. Since it's easier to model processes that are under one's control, the first major Web Services implementations will be internally focused integration efforts.

4.6 Providing fine-grained access to data, functionality, and logic

In a Service-oriented environment, it will make the greatest sense to define a set of fine-grained, loosely coupled Services that have the greatest impact in an enterprise, and can be reused as part of other Web Services.

Rather than treating an application or system as a monolithic block, as do most EAI solutions, SOI solutions allow users to get a greater level of interaction and granularity with components deep within the application.

Rather than treating an application or system as a monolithic block, as do most EAI solutions, SOI solutions allow users to get a greater level of interaction and granularity with components deep within the application. Many enterprise software vendors and developers can thus write Web Services interfaces for individual components within the application, rather than creating a set of complex APIs for a huge system to allow access to the whole application. This granularity also may make SOI more efficient than traditional EAI solutions.

V. ROI for Service-Oriented Integration

Some organizations will benefit more than others from different integration approaches. Companies that only have a single, host-based computer system with few applications and few users running on those applications will find little value in any integration approach—especially with the complexity of traditional EAI and B2Bi solutions. Also, companies that use a single computing environment or platform for their applications and systems will have little trouble integrating their systems without use of EAI or other integration technologies.

However, the above scenarios are not the reality for most enterprises of any size. Even in the case where the IT environment is initially simple, the business environment changes and the single-vendor or single platform solution no longer becomes the IT reality. Therefore, integration solutions are cost-effective for the vast majority of businesses.

The Total Cost of Ownership (TCO) for traditional EAI costs consist of three main parts:

- *Architectural cost* – Including integrated development, execution and operations environments. Architectural costs consist of license costs, new hardware required to develop, run and monitor integrations, and the cost to implement architectural software and hardware.
- *Integration cost* – Integration costs relate to the development of actual system interfaces and producing the collaborations between systems.
- *Operational cost* – The day-to-day costs of managing and operating the integration solution. Operating costs generally are driven by the number of interfaces that need to be maintained and rise as more interfaces are put into production.

For all forms of integration, around 80% of architecture costs are incurred within 6 months of implementation, while ongoing architectural costs are some percentage of the initial costs, often 10-15% spiking to 25% when new licenses must be purchased.

For all forms of integration, around 80% of architecture costs are incurred within 6 months of implementation, while additional expenses for hardware or licenses may be incurred as usage spreads throughout an organization. Ongoing architectural costs are some percentage of the initial costs, often 10-15% spiking to 25% when companies must purchase new licenses for upgrades, add servers for scalability, or other factors.

5.1 TCO and ROI of Traditional EAI and B2Bi Solutions

Most EAI solutions have a fairly high price point, depending on the size of the initial installation. Initial architectural costs often exceed $100,000 and can often be in the millions of dollars for complex installations.

In general, EAI efforts have shown that integration costs are 25-40% lower than custom integration efforts. The main reason for this difference is that EAI provides pre-packaged adapters to connect to most enterprise applications and data sources. In fact, much of the value of the EAI offerings is their pre-packaged adapters and communications capabilities. Prior to the emergence of EAI and B2Bi solutions, the only real approach to tying systems together was to handwrite code for integration. In fact, there's still a considerable amount of programming done today that is this form of custom integration. Many of these efforts could be made dramatically more efficient by the use of EAI adapters and other "standardized" middleware.

Operational cost is one of the areas where EAI solutions can also provide a great deal of benefit beyond what is possible with custom integration. The custom integration approach requires any changes to the integration of systems by spending costly IT resources in re-coding applications—sometimes from scratch! EAI and B2Bi solutions offer a greater value by providing a consistent and efficient architecture for modifying and tuning the integration between systems.

5.2 Improving the ROI Outlook with SOI

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* – Web Service applications 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* – Web Services promise a high degree of reusability, faster time-to-market, and ability to integrate with third-party systems and trusted business parties. Tools offer sophisticated functions to turn existing service implementations into Web Services.
- *Streamline Business Operations* – Web Services 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.
- *Potential for New Revenue Streams* – Use of externally supplied Web Services allow maintenance of existing customers or addition of new revenue streams.

5.2.1 Reduction in Implementation Costs

With SOI, the availability of nodes that are capable of being exposed as Web Services drive the architectural costs, rather than how many adapters or individual points of integration are necessary. An SOI implementation may require a higher initial architectural investment than a custom solution, but is generally much lower than typical EAI solutions.

Users who implement SOI can experience great economies of scale in integration costs as well, achieving up to a 60% savings from custom integration efforts and up to a 40% savings from EAI efforts. The main reason for this savings is that SOI systems can not only leverage existing data adapters provided by EAI or other vendors, but can also access a wider range of data and application functionality through component-level integration. Since all Service-oriented applications communicate using a common methodology, far fewer interfaces must be developed to communicate with those sources that have already been Web Services-enabled.

Experience has shown that while EAI generally provides a 50% to 80% reduction in application maintenance and operations cost by reducing the number of interfaces that must be maintained, an SOI approach can save up to 90% of the application maintenance costs since it provides just a single interface, namely the Web Service.

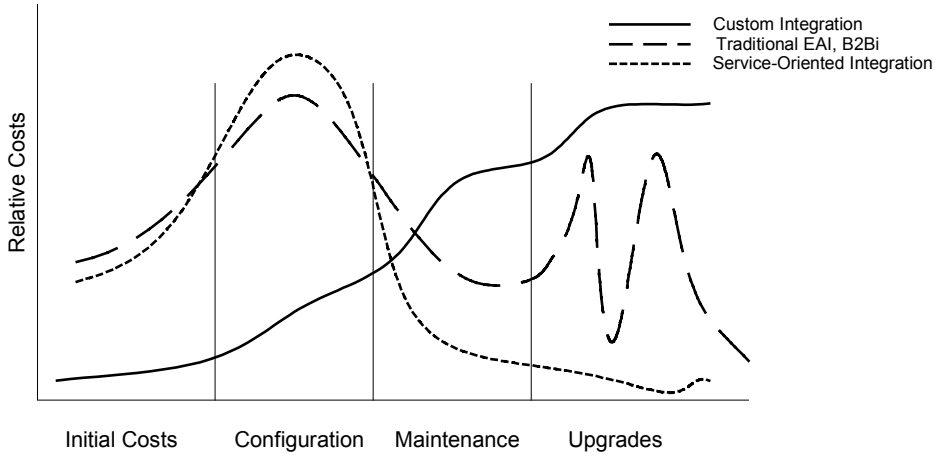
5.2.2 Leverage single SOI investment for multiple integration challenges

Another major point in favor of SOI approaches over EAI and B2Bi solutions is that companies can leverage a single investment in an SOI architecture for multiple types of integration problem—internal integration, external integration, data integration, and combinations of the above. The same cannot also be said for traditional EAI and B2Bi solutions.

5.2.3 Standards-based solutions improve long-term ROI

Basing the organization's IT infrastructure on vendor-neutral standards has a value in itself. Many traditional EAI solutions depend on connecting to data sources through proprietary connection or adapter layers. Each of these layers adds cost, complexity, and inefficiency. Utilizing standards-based technologies such as XML, SOAP, and HTTP simplifies these layers, thus reducing the cost and complexity of implementing solutions. These standards-based solutions may not positively impact efficiency, however.

Figure 5.1: Integration Costs for Traditional and SOI Solutions



5.3 The Movement to the "Agile Enterprise" Offers Greatest ROI

Decision Point

The potential ROI realized by adopting Service-Oriented Architectures (SOA) far outweighs the slight benefits an organization gets from using Web Services as simply a "better API" for accessing application functionality.

Since Web Services are a specific implementation of the Service-Oriented Architecture (SOA) using standard protocols and technologies, it inherits some of the best features of SOA: the ability to dynamically describe, publish, and bind to resources. It is hard to fathom how much impact this will have on businesses. The potential ROI for this usage of SOI far outweighs the slight benefits an organization gets from using Web Services as simply a "better API" for accessing application functionality. Merely substituting tightly-coupled, EAI-style technologies such as CORBA's IIOP with SOAP does not gain much tangible advantage.

In a traditional distributed systems environment, developers must have all the well-defined APIs for all the components they wish to invoke. These APIs must then be established at the time they develop an application. However, upgrades to the various system components will often change the API, requiring users to take care to support a whole range of API versions in order to avoid total system failure. Often, the entire distributed system must be upgraded and fully tested. These upgrades are often expensive and risky, and therefore infrequent. However, Web Services are self-defining and support dynamic discovery, potentially solving the problem of "all-or-nothing" upgrades or system changes. Instead of having an API set at design time, each Web Service points to the interface specified in its WSDL file. Consumers of that Web Service know where to find the WSDL file at design time or at runtime. The Web Service consumer then determines the current interface, and therefore can create SOAP messages the Web Service can understand. If the Web Service changes due to an upgrade

or some other change, then Web Service consumers can find out in a dynamic and adjustable manner.

Decision Point

"Agile Enterprises" consist not of IT fiefdoms that must be forced to reconcile with each other, but modular, self-contained units that can freely interoperate with each other on an as-needed (or even ad-hoc) basis.

Taking advantage of the dynamic capabilities of Web Services allows companies to subscribe to the notion of the "Agile Enterprise" in which companies consist not of IT fiefdoms that must be forced to reconcile with each other, but modular, self-contained units that can freely interoperate with each other on an as-needed (or even ad-hoc) basis. This concept may change the fundamental nature of businesses themselves.

Companies that become Agile Enterprises can realize ROI by:

- *Obtaining functionality as needed from third-parties:* Enterprises can obtain functionality in the form of Web Services from third-party software vendors and pay for it as part of a license fee, on a subscription basis, or by transaction, eliminating the need for purchasing or developing business services in-house.
- *Greater application flexibility:* Services can evolve independently without affecting their ability to interoperate, thus significantly reducing software maintenance time and cost.
- *Dynamic Discovery and Binding to Business Parties:* In the future, when businesses expose their critical business processes in the form of Web Services, it will be possible to discover and bind to these services on-the-fly, radically simplifying the process of connecting enterprises.
- *Web Services Enable New Business Models:* Companies that provide or consume Web Services can take advantage of new business models including selling Web Services, hosting Web Services, providing intermediary services, value-adding Web Services, and other models that may not even be conceived of at the moment.

ZapThink believes that many of the aspects of dynamically discovering and binding to businesses are not going to be a reality for some time—if ever. It would be unrealistic to assume that enterprises will choose to transact with any arbitrary business in an automated fashion without a pre-existing business relationship. There simply are too many trust issues involved in that scenario.

In addition, the service-outsourcing model raises the potential for friction between enterprise IT departments and external Web Service providers. However, there is a need for the dynamic establishment of all the computing and systems aspects of businesses once parties establish a business relationship. So, we can expect to see an increased amount of benefit from these dynamic business models as time progresses.

5.4 Realize Integration ROI Internally First, Externally with Trusted Parties Second

Companies can realize the most immediate value proposition for Web Services by solely implementing Web Services internally, behind the firewall. Why? For a simple reason: companies can maintain control of both ends of the integration in an internally focused integration environment. With fewer variables to be concerned about, enterprises can achieve the greatest ROI from an internal SOI implementation.

Integrating systems between two businesses is not only a technological problem; it requires pre-existing business relationships between the companies.

Once internal integration issues are under control, businesses can solve externally facing integration issues. However, integrating systems between two businesses is not only a technological problem; it requires pre-existing business relationships between the companies. Only when business relationships and contracts have been negotiated can two companies get together and decide how

they will communicate and handle business transactions. As a result, businesses don't have as much control of both ends of the relationship, necessarily slowing the time-to-return of ROI for SOI implementations.

VI. Barriers and Challenges to SOI Adoption

As with all new, emerging technologies, there are substantial risks and pitfalls to implementing SOI in the short term. The problems that will inevitably surface when enterprises implement Web Services will frustrate enterprises looking for an "easy" solution to integration, and they will have to keep their eyes wide open to the potential issues involved in using Web Services for integration.

Current Web Services specifications in widespread use are insufficient to meet many enterprise-class system integration requirements.

The current incarnation of Web Services is limited in many ways and allows companies only to achieve a basic functional level of integration between applications. Web Services are limited by their inability to handle long-lived transactions and basic request/response functionality. Other items that vendors must resolve include issues of interoperability of different types, and challenges to the robustness, reliability, and security of Web Services.

6.1 Interoperability of SOI implementations

Interoperability defines the ability for heterogeneous implementations to be able to communicate with each other as successfully as they would be able to communicate in a homogeneous environment. It is in this area that Web Service implementations currently have some challenges that will soon, hopefully, be surmounted.

In particular, there has been much criticism that Microsoft .NET SOAP implementations are not 100% interoperable with Apache and other major SOAP implementations. However, one should not lose much sleep worrying over this problem, as many organizations have created a number of major interoperability working groups, task forces, benchmarks, and conformance suites to help move the industry towards global Web Service interoperability. One of these key conformance suites is the SOAPBuilders Interoperability test suite that provides a collection of simple SOAP RPC invocations, in which a client sends a parameter of a certain type (integer, string, etc.) and the server simply returns a parameter of the same type and value.

ZapThink expects that vendors will resolve low-level interoperability issues by the end of 2002.

However, ZapThink expects that vendors will resolve these low-level interoperability issues by the end of 2002. What will remain are higher-level interoperability issues surrounding the ability for different Web Services implementations to maintain the same level of reliability, security, transaction control, business process automation, and semantic consistency. In these areas, differing implementations will cause significant interoperability challenges.

6.2 SOI specifications are far from complete

Web Services are a new and immature technology to base SOI solutions on. While simplistic Web Services are possible using the current set of specifications: SOAP, WSDL, and UDDI, these specifications are insufficient to power enterprise-class system integration needs. In particular, SOI solutions require significant additions to the current specification stack to enable:

- Distributed transaction control
- Reliability and security
- Scalability and performance

6.2.1 Distributed Transaction Control

One of the primary challenges with Web Services is the fact that it is very difficult to provide and maintain distributed transaction control within an enterprise, let alone among disparate businesses and applications. Long-lived transactions across enterprises in particular will be very difficult to maintain and control, posing challenges to the ability to develop and deploy enterprise-class Web Services implementations.

6.2.2 Reliability and Security

Vendors must develop a number of standards, vendor offerings, and server-side tools in order enable companies to create robust and secure Web Services implementations. There is no doubt that vendors will improve this situation in the months and years ahead, but until then, Web Services will be a "Wild West" frontier land fraught with challenge and peril.

In general, in order to make true transparent integration happen, we not only need standards for security, but we also need a means to "normalize" security between different implementations. What vendors and standards bodies have not fully specified is how different security implementations could be integrated together.

6.2.3 Scalability and Performance

While Web Services leverage existing technology, their performance in mission-critical and high-performance systems has yet to be fully proven. Many Web Service platform vendors are indeed attempting to make sure that their end products are scalable, and there is no doubt that we will see high performance Web Services implementations. However, the current number of case studies for these high-performance implementations is limited.

It is also a difficult proposition to test Web Services across multiple execution environments and SOAP implementations. Therefore, a number of challenges exist not only in scaling the Web Services themselves, but in testing their interaction with other services.

6.3 SOI can require the re-architecting of systems

As implied above, the application of Web Services is not simply the use of SOAP over HTTP as the communications method. Rather, the Service-oriented approach begs us to re-architect our systems to make them Service-aware. This need to re-architect or redesign systems is potentially a Herculean effort. ZapThink does not believe that Siebel and SAP will take an axe to their systems and chop up their monolithic applications into easily digestible Web Services blocks. Instead, ZapThink expects slow and grudging progress to be made by these vendors in the use of Web Services.

ZapThink expects slow and grudging progress to be made by enterprise application vendors in the use of Web Services.

Vendors will also have to re-architect traditional EAI and B2Bi solutions in order to truly take advantage of Web Services. The requirement will be that EAI solutions will have to provide support for Service integration rather than simply application integration. They will need to be able to treat systems and data sources as Services to be combined with other Services, rather than simply mapping and transforming data and application logic. As the Web Services stack develops, EAI solutions will also need to be able to take care of discovery,

binding, and invocation of Web Services at runtime, not just design-time, what ZapThink calls "just-in-time integration."

6.4 Lack of Semantic Integration

One of the most important facts to understand about SOI is that while it provides a methodology for connecting systems together, it doesn't specify what the systems are actually talking about. Basically, the semantics of the business conversation are left as an exercise to the end user. Web Services, and by extension SOI, does not provide the meaning of the words or grammar of any particular application. How will multiple participants in a value chain agree to the meaning of each of the quantities in their common business processes? In traditional B2Bi, participants must expend significant resources resolving these semantic issues before they can send the first message. Loose coupling helps to solve this problem by abstracting the actual implementation of the semantic interfaces. When B2Bi is conducted in a loosely coupled manner, then it is feasible for an intermediary to provide the semantic translations necessary to resolve the difficult issues of meaning. However, this problem is far from simple to solve—if it can be solved at all.

Rather than worrying about the lower levels of communications and system interoperability, we now have to worry about the higher levels of integration challenge—business process, business semantics, security, and workflow.

As a result, we're in the position of just shifting around integration issues in what might be called a shell game. Rather than worrying about the lower levels of communications and system interoperability, we now have to worry about the higher levels of integration challenge—business process, business semantics, security, and workflow. These are the challenges that come about when trying to use Web Services for "enterprise-class" integration. While there is much work being done to address the issues around business process, workflow, and security, what remains mostly unresolved are the issues of semantic ambiguity.

Some of the ways that organizations are looking to resolve semantic ambiguity include the development of some sort of unified business language such as those promoted by ebXML and RosettaNet. These vocabularies specify how specific documents such as purchase orders and invoices appear in an XML format. However, it is in practice very difficult to actually implement a vocabulary that every business can adhere to. Typically, these vocabularies represent the bare minimum that can be agreed to by all the participating parties ("Lowest Common Denominator") or is the superset of all the requirements of all participating parties ("Greatest Common Denominator"). Either way, businesses never end up using the whole specification; at most they implement just 10% of it. The resultant goal of achieving a universal business library is thus never met.

6.5 Web Services Introduce their own Level of Complexity and Inefficiency

Web Services technology introduces an additional software layer and thus additional complexity, which must be offset against the benefits. This software layer handles the abstraction in a way that that is additive to the underlying object technologies and protocols. For example, Web Services don't negate the need for EJB and COM, they just add an extra interface on which to communicate. As such, they are less efficient than other distributed computing technologies.

Web Services are less efficient than other distributed computing technologies.

In addition, SOI poses some interesting challenges in testing the aggregate application. Whereas with traditional EAI and B2Bi solutions, the user has some level of control over both ends of the communication, Web Services producers and consumers have an arms-length relationship, at best. So how do developers and other participants in the integration process test their long-lived, asynchronous transactions across a distributed Web Services environment?

VII. Market Size and Future Trends

7.1 The Convergence of EAI, B2Bi, and Data Integration Markets

From a Web Services perspective, internal and external integration at the application and data layers are equivalent. Interfaces of any type and at any location can be abstracted as a Web Service, and as such accessed by SOI solutions. The result is that there will be little need in the future for artificial divisions between EAI, B2Bi, and data integration solution classes. From a market perspective, this means that the currently separated integration solution markets will converge on a single market for Service-Oriented Integration (SOI).

ZapThink believes that by 2006, all integration products will be Web Services-enabled, thereby falling into the SOI solution category. The current EAI, B2Bi, and data integration vendors, along with emergent SOI entries and Web Services platform players will converge on a single set of features, albeit approaching the problem from different perspectives. This means that current EAI and B2Bi vendors will have to evolve to an SOI model in order to remain competitive in the future.

Decision Point

The market for Service-Oriented Integration (SOI) solutions will subsume the current markets for EAI, B2Bi, and data integration solutions by the year 2006.

7.2 Market Opportunity and Sizing

The market for Service-Oriented Integration (SOI) solutions will subsume the current markets for EAI, B2Bi, and data integration solutions by the year 2006 as detailed above. The market for SOI solutions was $435 Million (US) in 2001, with growth in 2002 is expected to be relatively flat, (only a 2% increase) due to global economic weakness for these solutions. However, growth in 2003 through 2006 is expected to recover quite strongly due to superior solutions for integration as provided by SOI.

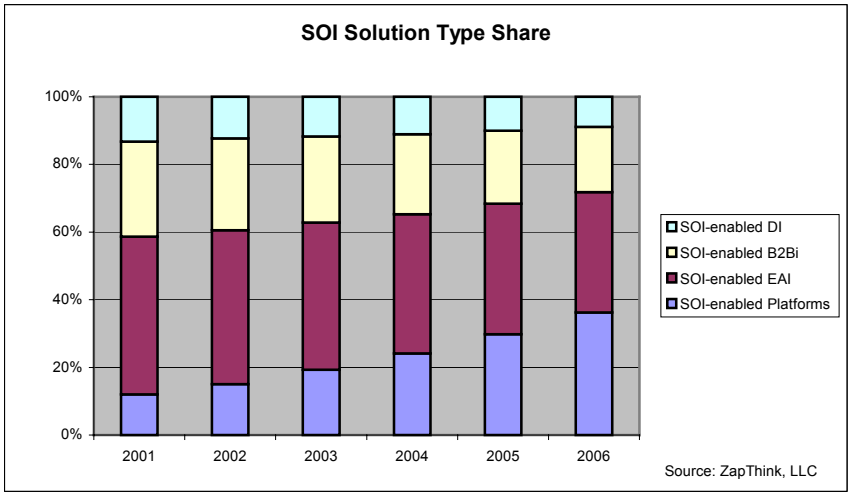
The market for SOI will follow the general trends for the EAI and related markets. Overall, the revenue of the SOI market is expected to grow to slightly over $6.2 billion in 2006.

Table 7.1: Growth of Aggregate SOI Market 2001-2006 ($MM)

	2001	2002	2003	2004	2005	2006
SOI-enabled Platforms	52	99	207	456	959	2013
SOI-enabled EAI	192	211	327	606	1114	2006
SOI-enabled B2Bi	104	114	206	379	701	1262
SOI-enabled DI	87	96	172	310	567	948
Total Market	435	520	913	1751	3341	6228
Total AGR	N/A	20%	76%	92%	91%	86%

Source: ZapThink, LLC

Figure 7.1: SOI Solution Type Share of Total Market



7.3 Current Vendor Positioning and Market Share

The top three EAI vendors have over 43% of the overall EAI market. With the entrance of Microsoft and other vendors in 2002, this landscape is expected to change with vendors that are unable to compete against the major platform vendors expected to consolidate or re-position, while IBM, Microsoft, and key EAI vendors are expected to continue to grow.

Table 7.2: Ranking of SOI Vendors

Vendor	2002 Market Share (%)
IBM	18.1
TIBCO	13.1
webMethods	12.5
SeeBeyond	9.0
Vitria	6.1
Peregrine	6.0
IONA	5.3
Sybase	5.0
Software AG	4.0
Others	16.2

Market shares are based on software license revenues for all integration, adapter, workflow, transformation, and connector technologies that would be subsumed by SOI technologies, and does not include message transport technologies.

Decision Point

ZapThink sees that the SOI-enabled Web Services Platform vendors currently have the competitive edge, while Emerging SOI solutions have the technical edge.

ZapThink sees that the SOI-enabled Web Services Platform vendors currently have the competitive edge, while Emerging SOI solutions have the technical edge. SOI-enabled EAI, B2Bi, and Data Integration solutions will face increased competitive and technical pressure from these vendors as the market matures.

Figure 7.2: Competitive Positioning for SOI Solution Types

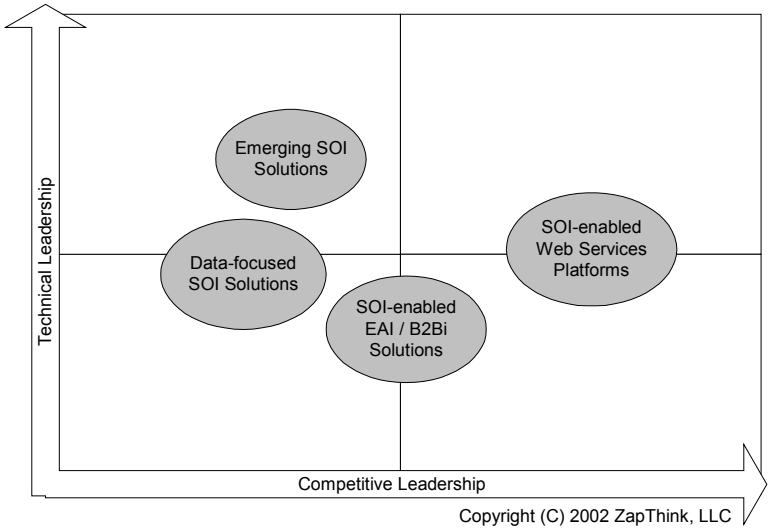


Table 7.3: Vendor Comparison Matrix

Vendor	Key Differentiators	Strengths	Weaknesses
Integration-enabled Web Services Platforms			
BEA WebLogic Integration	➢ J2EE CA based ➢ Combined app server, B2B, and BPM.	➢ Leadership in the App Server Market ➢ Experienced in integration	➢ Lacks development platform ➢ Competing with "bigger" players
IBM WebSphere	➢ Cross-platform ➢ Leverages MQ-Series	➢ Very "complete" product offering ➢ Strong solutions and hardware	➢ Plays better in "large" enterprises ➢ High TCO compared to BEA, Microsoft
Microsoft BizTalk	➢ Leverages .NET ➢ Robust document transformation	➢ Strong software platform ➢ Robust development environment ➢ Low TCO	➢ Doesn't play as well in "large" enterprises ➢ Perceived as closed platform
SOI-enabled EAI and B2Bi Solutions			
Peregrine	➢ B2B integration focused ➢ Document-exchange focus	➢ B2B solutions courtesy Extricity ➢ Strong EDI legacy	➢ Company weakness ➢ Uncertain future ➢ EDI legacy
SEAGULL	➢ Mainly host-based and mainframe offering	➢ History of strong mainframe offerings ➢ Added Hostbridge solution set	➢ Not much EAI functionality beyond mainframes ➢ Not very "deep" integration with App Servers

SeeBeyond	➢ Complex processes that span many applications ➢ Can use other messaging platforms	➢ Distributed technology ➢ Breath of solution ➢ Strong in health care, retail, and financial services,	➢ Limited adapter library ➢ Few partnerships or OEMs ➢ Sees Web Services as "better adapter" ➢ High TCO
Sybase	➢ Focus on enterprise app integration	➢ Strong ebXML solution ➢ Vertical industry solutions	➢ Needs to expand B2B support beyond ebXML
TIBCO	➢ Messaging-focused	➢ Strong EAI player ➢ Fast and scalable messaging technology	➢ Complexity of implementation architecture ➢ Still working on common tooling across the product line ➢ High TCO
Vitria	➢ Visionary leadership in process management	➢ Strong process management ➢ Strong EAI player ➢ Moving "up" the integration stack	➢ High TCO
webMethods	➢ B2B and EAI adapter focus	➢ EAI industry leader ➢ B2B standards support ➢ Embedded in many enterprise apps	➢ Weak process management ➢ High TCO
Emerging SOI Solutions			
Actional	➢ Focused on Web Services management	➢ Microsoft and Java-based platforms	➢ Focused more on management than on integration
Attunity	➢ Resold as Oracle Open System Gateway ➢ Virtual DBMS approach	➢ Focusing on OEM, embedding in apps	➢ Small list of adapter support
Cape Clear	➢ Combined IDE and integration environment	➢ Early market entrant ➢ Lower TCO	➢ Focused on certain integration classes ➢ Tied to development environment
Infoteria	➢ RosettaNet B2B focus	➢ Strong RosettaNet-based B2B offering	➢ Focused narrowly on RosettaNet-based integration
IONA	➢ Combines CORBA, J2EE, Web Services	➢ Early leaders in CORBA ➢ Strong interoperability	➢ CORBA legacy

Software AG EntireX	➢ XML storage and processing model	➢ Strong user base	➢ While an old company, acting as new startup in EAI
Data-Oriented SOI Solutions			
Coherity	➢ Using Native XML Data Store as data "aggregator"	➢ High-performance XML data store ➢ Handles data "diversity"	➢ New to data integration market ➢ Focused on XML-based data
iWay	➢ Hundreds of adapters	➢ Leverages Information Builders ➢ Strong transformation technology	➢ Adapters will be commoditized
Nimble Technology	➢ XML-based data "intermediation" ➢ Supports XML Query standards	➢ Advanced data integration technology ➢ Can link data semantically	➢ New startup ➢ Architecture unproven
XAware	➢ XML-based data aggregation ➢ Hundreds of XML-based adapters	➢ Partners with major data providers	➢ New startup ➢ Risk of commoditization

Source: ZapThink, LLC

ZapThink believes that the emerging SOI solutions vendors and Web Services platforms vendors will apply increasing pressure on the SOI-enabled EAI and B2Bi solutions. Combined with the effects of EAI, B2Bi, and Data integration convergence, ZapThink believes that there will be significant consolidation in the industry by 2004.

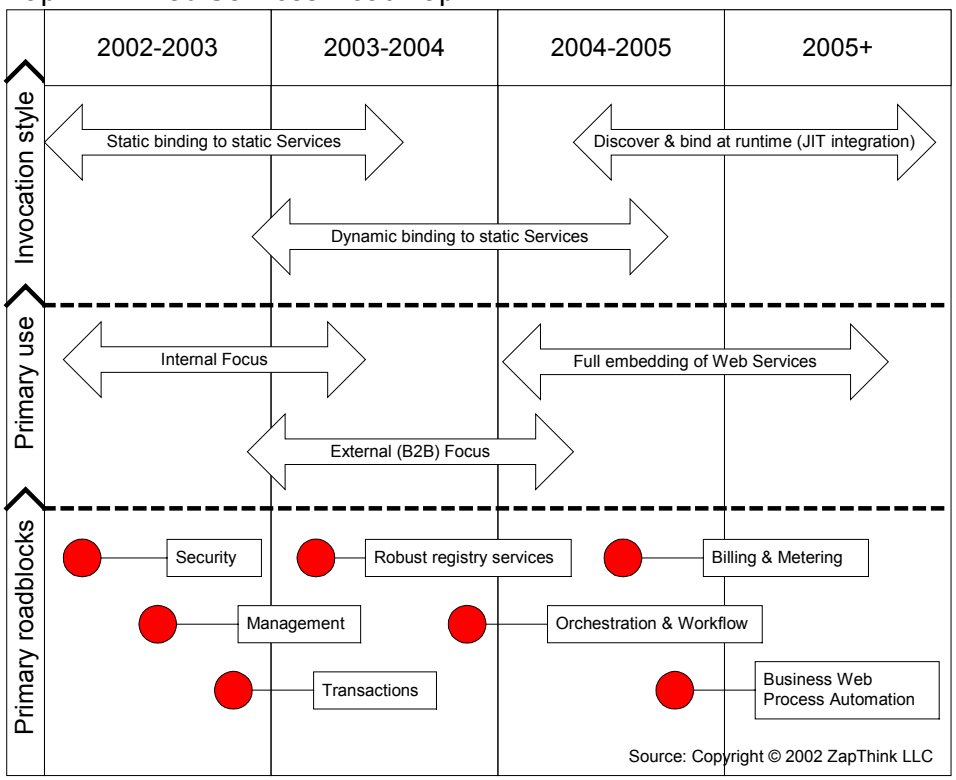
7.4 Future Directions for Service-Oriented Integration

Currently, most Web Service applications are extensions of existing functionality and logic, but the long-term strategic value is in natively Web Service-conversant applications that can provide a rich user experience while simultaneously providing the architectural value of Web Services.

ZapThink's roadmap for the future of Web Services is illustrated below.

Figure 7.3:

ZapThink Web Services Roadmap



SOI solutions simply push integration problems to a higher level in the integration "stack."

7.4.1 Semantic Integration

SOI solutions don't solve all the integration challenges in an enterprise. They simply push the problems to a higher level in the integration "stack." As the Integration Zipper implies, as lower levels of integration fall into place, companies must then face new challenges, for example, the challenge of integrating systems at the semantic layer.

Semantic integration is a classically difficult, if not practically impossible problem to solve. However, there are a number of efforts being made to solve industry-specific semantic representation (ebXML, RosettaNet) as well as cross-industry platforms for semantic integration (the Semantic Web, Topic Maps). However, these efforts have met with significant challenges and have not met with much industry support.

7.4.2 Web Services Management

As Web Services proliferate in the corporate IT environment, so too will tools and solutions for managing those Web Services. A natural extension of that market will be for management of integration infrastructures, in particular, SOI solutions. As such, ZapThink expects that current SOI solutions will have to plan for the capability of integrating with third-party Web Services management solutions in the next 12-18 months.

Read ZapThink's upcoming *Web Services Management Technologies and Trends* Report for more insight on the interaction between SOI and Management solutions.

7.4.3 Grid Computing

Grid Computing represents the ultimate incarnation of the Service-Oriented Architecture (SOA). In a grid environment, computing and data resources are made available in an aggregated, abstracted manner to any client that chooses to tap into these resources. Users will not need to know (or maybe ever know) what specific computing resources they are accessing, and as such the grid makes integration a side-effect rather than a forethought. In order for computing to evolve to a grid-based architecture, it is first incumbent that enterprises integrate using first-generation SOI approaches..

VIII. 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. SOI solutions can be applied equally to solve EAI, B2Bi, and Data Integration problems.

Decision Point

ZapThink expects that SOI solutions will become the predominant way that integration will be handled in the future.

However, SOI solutions introduce challenges of their own, including their relative immaturity, incomplete standards stack, and questions about their security, reliability, and ability to handle distributed transactions. ZapThink expects that many of these problems will be solved by end of 2003, and SOI solutions will become the predominant way that integration will be handled in the future.

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

8.1 Key Notes

- *Enterprises must deal with the challenge of connecting arbitrary systems in a manner that is low cost, manageable, efficient, and secure.*
- *Web Services aren't an integration technology at all, but just a distributed computing technology that lends itself well to being used in integration scenarios.*
- *Currently, Web Services and SOI is focused on solving API and Business Process-level integration issues and doesn't deal with semantic or information-meaning level integration.*
- *In a Web Services context, there really is no difference among EAI, B2Bi, and Data Integration.*
- *The integration middleware vendor plays a valuable role that the Web Services Platforms cannot provide: the integration of multiple different, disparate technologies without prejudice or bias as to the native platform on which those technologies are executed.*
- *ZapThink believes that pure SOI-based solutions will have to provide significant value beyond what Web Services Platform and SOI-enabled EAI and B2Bi vendors are offering in order to be competitively viable.*
- *Efforts to extract and manipulate information from multiple, disparate data sources have slowed efforts to XML-enable the enterprise.*
- *SOI simplifies system integration by providing a single, simple architectural framework in which to build, deploy, and manage application functionality.*
- *As a result of their simplicity and reliance on open standards, SOI solutions can be less expensive to implement than traditional EAI and B2Bi solutions.*
- *The use of open standards allows organizations to have the widest possible selection of options and alternatives for SOI solutions.*
- *For all forms of integration, around 80% of architecture costs are incurred within 6 months of implementation while ongoing architectural costs are some percentage of the initial costs, often 10-15% spiking to 25% when new licenses need to be purchased.*
- *Rather than treating an application or system as a monolithic block, as do most EAI solutions, SOI solutions allow users to get a greater level of interaction and granularity with components deep within the application.*
- *ZapThink expects low-level interoperability issues to be resolved by the end of 2002.*
- *Integrating systems between two businesses is not only a technological problem; it requires pre-existing business relationships between the companies.*
- *Current Web Services specifications in widespread use are insufficient to meet enterprise-class system integration requirements.*
- *ZapThink expects slow and grudging progress to be made by enterprise application vendors in the use of Web Services.*
- *Rather than worrying about the lower levels of communications and system interoperability, we now have to worry about the higher levels of integration challenge: business process, business semantics, security, and workflow.*
- *Web Services are less efficient than other distributed computing technologies.*
- *SOI solutions simply push integration problems to a higher level in the integration "stack."*

8.2 Decision Points

- *Microsoft and IBM have made strong entries into this space that will be a challenge for other providers of SOI solutions.*
- *Enterprises with significant investments in existing EAI and B2Bi solutions should encourage their solution providers to support SOI requirements more aggressively.*
- *Business modeling and identifying the granularity of Web Services will become as important as the business logic contained within the Web Services themselves.*
- *The potential ROI realized by adopting Service-Oriented Architectures (SOA) far outweighs the slight benefits an organization gets from using Web Services as simply a "better API" for accessing application functionality.*
- *"Agile Enterprises" consist not of IT fiefdoms that must be forced to reconcile with each other, but modular, self-contained units that can freely interoperate with each other on an as-needed (or even ad-hoc) basis.*
- *The market for Service-Oriented Integration (SOI) solutions will subsume the current markets for EAI, B2Bi, and data integration solutions by the year 2006.*
- *ZapThink sees that the SOI-enabled Web Services Platform vendors currently have the competitive edge, while Emerging SOI solutions have the technical edge.*
- *ZapThink expects that SOI solutions will become the predominant way that integration will be handled in the future.*

8.3 Figures

8.4 Tables

IX. Profiled Vendors

9.1 SOI-enabled Web Services Platforms

IBM

Please see ZapNote ZTZN-1050

Microsoft

Please see ZapNote ZTZN-1066

Sun ONE

Please see ZapNote ZTZN-1093

9.2 SOI-enabled EAI and B2Bi Solutions

Peregrine (formerly Extricity)
Please see ZapNote ZTZN-1078

SeeBeyond
Please see ZapNote ZTZN-0279

SEAGULL
Please see ZapNote ZTZN-0160

Sybase (formerly NEON)
Please see ZapNote ZTZN-1095

TIBCO
Please see ZapNote ZTZN-1100

Vitria
Please see ZapNote ZTZN-0263

webMethods
Please see ZapNote ZTZN-1107

9.3 Emerging SOI Solutions

Actional
Please see ZapNote ZTZN-0280

Attunity
Please see ZapNote ZTZN-0138

Cape Clear
Please see ZapNote ZTZN-0120

Infoteria
Please see ZapNote ZTZN-0107

IONA
Please see ZapNote ZTZN-0140

Software AG EntireX
Please see ZapNote ZTZN-0116

9.4 Data-Focused SOI Solutions

Coherity
Coherity is no longer in business

iWay (Information Builders)
Please see ZapNote ZTZN-1055

Nimble Technology
Please see ZapNote ZTZN-0114

XAware

Please see ZapNote ZTZN-0154

Related Research

Reports

- *Web Services Technologies and Trends* Report (ZT-WEBSRV)
- *The Pros and Cons of Web Services* Report (ZTR-WS102)
- *XML Data Store Technologies and Trends* Report (ZTR-ST100)
- *XML and Web Services Security* Report (ZTR-WS103)

ZapNotes

- *Attunity* ZapNote (ZTZN-0138)
- *CapeClear* ZapNote (ZTZN-0120)
- *Coherity* ZapNote (ZTZN-0144)
- *Data Junction* ZapNote (ZTZN-0254)
- *DataMirror* ZapNote (ZTZN-0185)
- *IBM* ZapNote (ZTZN-1050)
- *Infoteria* ZapNote (ZTZN-0107)
- *IONA* ZapNote (ZTZN-0140)
- *iWay* ZapNote (ZTZN-0250)
- *Microsoft* ZapNote (ZTZN-1066)
- *Nimble Technology* ZapNote (ZTZN-0114)
- *Peregrine* ZapNote (ZTZN-0186)
- *SEAGULL* ZapNote (ZTZN-0160)
- *SeeBeyond* ZapNote (ZTZN-0279)
- *Software AG* ZapNote (ZTZN-0116)
- *TIBCO* ZapNote (ZTZN-0216)
- *Vitria* ZapNote (ZTZN-0263)
- *webMethods* ZapNote (ZTZN-1107)
- *XAware* ZapNote (ZTZN-0154)
- *XML Global* ZapNote (ZTZN-0134)

Supporting Resources

ebXML web site: www.ebxml.org

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