

zapthink white paper

INFORMATION AS A SERVICE: SERVICE-ORIENTED INFORMATION INTEGRATION



INFORMATION AS A SERVICE: SERVICE-ORIENTED INFORMATION INTEGRATION

February 2004

Analyst: Ronald Schmelzer

Abstract

As organizations grow organically, they often implement multiple systems that contain information that is redundant, conflicting, or distributed across their organization. As such, the seemingly simple task of trying to gain a single view of the information contained in the enterprise is a significant challenge. The requirement for single view, aggregated views, or shared information cuts cross the value chain from sales leads, orders, to products, inventory – including e-government initiatives in public safety, health and defense. While many application integration approaches attempt to solve this disparate information challenge by removing the barriers to accessing information, the challenge still remains of how to gain intelligence from the disparate data in the enterprise.

Previous approaches to information integration have fallen short. Tightly coupled data or application integration approaches that mandate point-to-point connections between systems are too brittle to handle continuously changing business requirements. Message buses and business process management approaches offer some relief for integration challenges, but only solve parts of the overall information integration challenge and leave the task of aggregating disparate views of data to the user. Finally, model-driven approaches to information integration work well when a company has visibility into their data formats and schema, but present challenges in an organization with limited information visibility.

To address these challenges, this paper introduces the notion of *Service-Oriented Information Integration (SOII)*, which approaches information integration in a loosely-coupled, coarse-grained, asynchronous manner that seeks to avoid the requirements of comprehensive data modeling prior to integrating with them. This paper also presents XAware's compelling SOII solution for meeting evolving information integration challenges. Their approach provides enterprises with an XML based abstraction layer that provides aggregated, virtual access to all of a company's existing information assets.

All Contents Copyright © 2004 ZapThink, LLC. All rights reserved. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.

I. Understanding the Information Integration Challenge

Enterprises of all sizes, types, and ages have a wide range of systems and sources of information they depend on for their daily operation. These systems encapsulate a wide variety of information ranging from customer data to operational information and data that power a company's value chain. Much of this information is redundant, disparate, and scattered throughout the enterprise. These systems span many different operating system, application, and exchange architectures and as a result companies and government agencies must cobble them together to produce aggregated results that they can use to drive decision making, empower day-to-day operations, comply with new regulations, and enable connectivity with customers and partners.

Not only are companies dependent on information for their day-to-day operations, but also the information itself is highly valuable. Network or system outages, inefficient processes, or human-intensive operations can dramatically and critically injure any enterprise. On the flip side, any improvements in data storage, exchange, representation, and manipulation can greatly add to a company's bottom-line.

Yet, extracting valuable information from closed systems and aggregating it together with other information continues to be a significant and costly challenge for most companies. The seemingly simple task of trying to gain a single view of the information contained in the enterprise is a significant challenge. Integration is not only a critical challenge for most firms, but is also necessary for addressing critical operations for most firms. Integration allows business units and third parties to interact as a connected entity, facilitating functions vital to the flow of commerce. Integration allows a company to serve its customers and stakeholders better by improving access to corporate information.

Successful integration also allows enterprises to lower their cost of ownership of the applications they have invested in by reducing system complexity, simplifying management, and reducing the cost of change. In addition, integration allows for legacy systems to be replaced or retired without wreaking havoc on the rest of the corporate computing ecosystem. Finally, successfully addressing integration challenges allow companies to make better use of existing systems by continuing to find ways to leverage these systems for new applications.

Information Integration: The Semantic Challenge

There are fundamentally two sides to the integration challenge: dealing with disparate information itself and dealing with disparate applications and interfaces to that data. However, these two concepts—data and applications—are inextricably linked. Data by themselves are often inaccessible and unintelligible without the applications that process them, and applications serve no usable purpose without data. Thus, any complete integration solution must handle both the application interfaces as well as the data that flow over those interfaces.

Companies must integrate data and information from a wide variety of data sources. These data sources might be structured, as in the case of databases and enterprise applications, or might be semi- or unstructured such as web pages, PDF documents, Office files, email, media content, or a wide variety of data feeds and formats. The need to access information of so many disparate types from so many disparate sources and locations forms the *semantic* integration challenge that most companies must deal with today. Simply put,

Any improvements in data storage, exchange, representation, and manipulation can greatly add to a company's bottom-line.

A complete integration solution must handle both the application interfaces as well as the data that flow over those interfaces.

In order to provide the promise of seamless data integration, we must transcend simply loosely coupling the application interface and in addition provide loose coupling at the semantic level.

semantic integration means not just integrating two points so that they can talk to each other, but rather so that they can *understand* each other.

What makes semantic integration a challenge is two-fold: first, the representation of information and the information itself are often bound tightly together; and second, the information frequently lacks context. Developers often think not of the data themselves but rather the structure of the data: schemas, data types, relational database constructs, file formats, and so forth—structures that don't pertain directly to the information at hand, but rather our assumption of what the data should look like. In tightly-coupled architectures, data structures are absolutely necessary, since they provide systems a way of coping with the information they are being fed.

However, when the barriers to application integration are removed, instead of being helpful constructs, these various data structure representations actually get in the way. How systems store and represent information interferes with the meaning of that information. To be more precise, the meaning of information and the structure of that information aren't one and the same. For example, "August 7, 2003" is a date for sure, but whether or not it is stored as a string, date type, or integer shouldn't matter. Yet, developers often needlessly combine structure and meaning together inextricably. Furthermore, there isn't enough context in the structure to understand if the date is a birth date, a purchase order date, or any other date.

Thus, when one developer's assumption of a particular structure for some datum conflicts with another's representation, there is a data integration problem. In order to provide the promise of seamless data integration, we must transcend simply loosely coupling the application interface and in addition provide loose coupling at the semantic level.

The Need for Right-Time Integration in a Heterogeneous Environment

Companies also need access to information in a timely manner. Enterprises have frequently dealt with integration challenges by cobbling together systems that eventually provide some of the information required to empower decision-making. However, many times this information arrives too late to actually enable the business to make those decisions in an agile fashion. In addition, certain business operations, such as fulfilling customer purchases, real-time operations such as stock trading and online banking, and billing operations require instantaneous access to disparate information. In order to meet these needs, companies have resorted to making redundant copies of information just so that it is available to those critical systems when people need them. The end result is a tangled web of redundant information, dispersed throughout the organization and continuously out of synch.

Furthermore, companies and government institutions frequently exchange information with external third parties, such as suppliers, partners, businesses,

TAKE CREDIT FOR READING ZAPTHINK RESEARCH!

Thank you for reading ZapThink research! 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.

Earn rewards for reading ZapThink research! Visit www.zapthink.com/credit and enter the code **XAWSOII**. We'll reward you with ZapCredits that you can use to obtain free research, ZapGear, and more! For more information about ZapThink products and services, please call us at +1-781-207-0203, or drop us an email at info@zapthink.com.



Companies continuously find themselves with the challenge of meeting their data and application integration requirements in a heterogeneous environment of continuous change.

Screen scraping is useful when it is difficult or impossible to modify actual application logic.

citizens, non-governmental organizations, and others. These interactions are made more complicated by the fact that it is difficult to predict the kind and nature of the information these companies share. While supply chain partners trade important data relating to parts, customers, logistics, and purchases, the format of this information varies on a partner-by-partner basis, as do the means for exchanging this information. As a result, in most cases, many-to-many, business-to-business integration is a continuously moving target.

Not only is there heterogeneity in business-to-business relationships, but there also exists significant complexity in internal systems. Companies frequently make independent decisions on technologies for different aspects of their business, and concern themselves with tying together these disparate systems at a later date. In addition, many industries are prone to acquisition and merger activity, leaving their companies a complex checkerboard of differing technology implementations. As a result, there is rarely central decision making on technology adoption, and thus heterogeneity always rules the business that adapts to meet its changing market. Thus, companies continuously find themselves with the challenge of meeting their data and application integration requirements in a heterogeneous environment of continuous change. This is precisely the information integration challenge facing enterprises today.

II. The Problem with Traditional Approaches to Integration

Originally, companies sought to solve their integration challenges by simply connecting their disparate systems individually to each other. In this integration approach, known as “point-to-point” integration, the number of integration or interconnection pathways that must be established grows exponentially (or n -squared) with the number of systems to be integrated. Unfortunately, this integration approach becomes significantly more complicated 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.

Screen Scraping Approaches to Data Integration

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 in that any change to the visual interface requires a change to the integration logic – and data is only available if the original system is available.

ETL processes by definition are point-to-point, tightly coupled, and asynchronous.

Extract-Transform-Load Approaches to Data Integration

Another simple 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, many times, the data sources are unavailable to the user, and the data itself requires interpretation by business logic that is unavailable. Furthermore, ETL 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.

Enterprise Application Integration and B2B Integration Middleware Approaches

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. One of the solutions implemented by many companies is the application of 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 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.

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. Many of the current

While EAI systems may be enable reusable processes, they are rarely used that way in practice.

implementations of EAI solutions have been targeted at specific, custom problems and do not create flexible processes that can be 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.

Business Process Management Approaches to Integration

Business processes, which can be defined as the set of activities and interconnection between those activities that meet business requirements, have been automated and managed using a wide range of proprietary technologies that involved a high degree of customization. These solutions approached business process from a variety of viewpoints: business process as an extension of integration middleware, business process in the context of message-oriented middleware, business process as a separate layer built on top of heterogeneous systems, and business process capabilities built on top of application development platforms. Some of these solutions focused on human-oriented workflow, while others on machine-to-machine interaction, and yet others on both capabilities. Since most business processes connect multiple systems in the enterprise, it is no surprise that companies sought to use these business process tools as a way to solve their integration problems.

One of the traditional ways to solve business process execution is through the use of reliable message queues. Message Queuing (MQ) is a widely used technology that enables applications running possibly at different times to communicate across heterogeneous networks and systems. MQ solutions provide access to system functionality and data that may be temporarily unavailable, and as a result MQ is considered to be a very reliable and scalable way to solve certain integration problems. Companies further extended the MQ model by adding concepts such as the *Publish/Subscribe (Pub/Sub)* model as a way to implement complex business process scenarios.

However, while the MQ and Pub/Sub approaches offered better technical means by which to connect activities and tasks in a process, they did little to actually help businesses design processes from a business level that can be executed by disparate human and machine-based activities in the enterprise. These traditional approaches to business process are less than optimal, in part, because many of these approaches to business process management are complex, proprietary, and expensive. In particular, many of these approaches have suffered the following challenges:

- *Centralized solutions are hard to implement across corporate boundaries* – While many MQ, BPM, workflow, and Pub/Sub implementations have been extended beyond the corporate firewall, their centralized, proprietary interfaces make such implementations often complex, costly, and hard to manage.

MQ and Pub/Sub approaches did little to actually help businesses design processes from a business level that can be executed by disparate human and machine-based activities in the enterprise.

- *Proprietary process modeling and execution flows make interoperability difficult* – Many prior approaches to business process management utilized proprietary process design, representation, and execution architectures making interoperability with other solutions difficult.
- *Lack of standard interfaces to system resources made integration complex and expensive* – Not a fault with prior approaches, but rather with the state of the market at the time these business process management solutions were offered, the lack of standard interfaces to system functionality made the value proposition of a separate business process layer hard to sell.

Model-Driven Enterprise Information Integration Approaches

An emerging way of handling data and application integration needs is the use of modeling-centric approaches to the challenge of accessing disparate information in the enterprise. As opposed to specifically defining connections between systems or data sources, model-driven integration focuses on abstracting information and data sources into a set of models that describe an enterprise's information resources. This model captures both the actual data sources themselves, or the *physical model* of the data, as well as the more abstract way the enterprise uses the information – or the *virtual model*. Once an enterprise captures its information resources in a model, it can easily integrate this information using a middleware server.

Model-driven integration, while still new to most enterprises, offers a few advantages over traditional modes of integration. First, the approach enables platform independence as the models can represent one or more different types of actual physical systems. In addition, model-driven integration enables real-time access, because the model-driven solution does not manipulate copies of the data. Furthermore, model-driven approaches provide a certain amount of extensibility that enables the enterprise to model, integrate, and access legacy, existing, and even future data storage technologies. Finally, model-driven integration enables bi-directional data interaction between systems as enterprise users can model both data aggregation and insertion.

Even model-driven approaches, however, are not optimal in all information integration scenarios. The primary challenge is that enterprises need to set up their physical and virtual data models in advance. In order to achieve any benefit to this approach, enterprises must capture the essence of the information within its systems, including technical aspects of the data (which describe the structure of the data), and business aspects of the data (which describe the way the enterprise uses the data). You have to model all instances of data interaction, regardless of whether or not that model is used, potentially creating maintenance burden. Some organizations simply lack the ability, resources, or discipline needed to make modeling that is based on a centralized approach to information architecture a reality.

In addition, model-driven approaches are problematic in scenarios when multiple organizations must share models. Model development is often a challenge in a single enterprise environment, and trying to extend this across organizations can be especially challenging. Some implementations introduce processing bottlenecks by resolving queries from a dedicated server that houses modeling metadata. As a result, these dedicated servers are potential single points of failure and present reliability and availability issues. Finally, the models themselves are living entities. Companies must manage them when models and information representations change, making model-driven integration difficult in instances of frequent model change.

Some organizations simply lack the ability, resources, or discipline needed to make modeling that is based on a centralized approach to information architecture a reality.

Given the range of solutions for information integration in today's enterprise, what hope can companies find for an optimal approach to the agile, loosely-coupled integration that companies desire that don't require any significant upfront investment in model development or other set up of data sources and endpoints? This is the promise that *Service-Oriented Information Integration* (SOII) solutions offer to enterprises.

III. Introducing Service-Oriented Information Integration

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 perennial problem. Historically, the only way to solve this problem has been through custom coding, data warehousing, or the use of central repositories. Yet, there is another approach that seeks to apply standards-based, loosely-coupled approaches to integration.

Rather than trying to tie together applications for the purpose of creating an aggregated application to access, users seek to expose data sources as a set of Services that data consuming applications and organizations can combine in any arbitrary fashion. To enable this vision, there are two required technology components: standard ways of representing application interfaces and the data flowing over those interfaces, and architectural approaches to guarantee the loose coupling of both the interfaces and data.

XML and Web Services offer compelling solutions for the first part of this equation. XML can be 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. Web Services, then, represent a standard way of describing and interacting with application interfaces and data sources over the corporate network and between organizations.

XML and Web Services provide a better answer to semantic integration challenges than the integration methods described above 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.

Service-Oriented Information Integration

Rather than thinking about how to get information into or out of different systems, we can think about how to expose system functionality to whatever system cares to access it. In this way, we release ourselves from thinking of information in a point-to-point fashion, and instead think of information as freely available on a network of Services. As a result, we can reuse existing functionality in different ways: for building new, composite applications, or for extending existing functionality. In order for this new class of integration solution to escape the problems of previous attempts, the solution must contain the following characteristics:

- *Loosely coupled* – In a loosely coupled system, application consumers don't need to have knowledge beforehand about a given piece of system functionality, other than where to find it. Application functionality and the programs that invoke them can be changed independently of each other, instead of requiring a redesign of the involved components.

The conflict between management needs and operational reality is a perennial problem.

Rather than thinking about how to get information into or out of different systems, we can think about how to expose system functionality to whatever system cares to access it.

SOA is an approach to designing distributed computing infrastructures that considers software resources as services available on a network.

- *Coarse-grained* – Rather than interacting with a large set of detailed, fine-grained APIs, users can interact with systems through coarse-grained, business-level interfaces that roll up the functions of many different API calls into a small number of business-oriented messages. Often, the first step is simply to create fine-grained, low-level Services and then bring them together as higher-level Services to produce a set of coarse-grained, business-level capabilities that didn't exist before.
- *Standards-based* – Instead of utilizing proprietary or closed APIs that require users to learn the intricacies of a particular vendor's platform, integration tools that successfully address the challenges of business agility will be standards-based, lowering the cost of integration and allowing the widest possible range of developers.

In order to meet these requirements, companies are beginning to implement an architectural approach known as *Service-Oriented Architecture* (SOA). SOA is an approach to designing distributed computing infrastructures that considers software resources as services available on a network. Producers of these services must be able to publish information about them in a service registry or repository, where service consumers can then look up the services they need and retrieve the information about those services they need to bind to them. This "publish-find-bind" triangle forms the core of an SOA.

The vision of using Web Services-based SOAs for solving information integration challenges is known simply as *Service-Oriented Information Integration* (SOII). 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, SOII provides an abstracted interface to information that systems can interact with. Systems merely need to expose their capabilities as Services, and other systems that choose to interact with them can simply discover those Services and bind to them either at runtime or design-time. Rather than planning in advance how a specific application will tie into other applications, developers should plan, develop, and deploy their applications in a service-oriented manner. The point of integration is to allow arbitrary applications, systems, and data stores to communicate without concern as to the other system's requirements, and SOII fulfills these requirements.

The Business Value of SOII

SOAs, however, offer more than technical advantages over other approaches to distributed computing. Fundamentally, SOAs offer a different perspective on the way that an organization accesses its IT capabilities. Companies must understand this new perspective in order to get the full advantage of their Web services implementations, and build IT environments that are flexible and responsive. In particular, SOAs offer the following core principles:

- *The business drives the services, and the services drive the technology* – In essence, services act as a layer of abstraction between the business and the technology that hides the implementation details from the business user, and with them, much of the complexity. Both IT and line-of-business personnel must understand the dynamic relationships between the needs of the business and the available services on the one hand, as well as the technical underpinnings that offer the layer of abstraction required by the services on the other.
- *Flexibility and responsiveness are the fundamental business requirements* – Instead of dealing with rigidly-defined, concrete requirements from business, loosely coupled services in an SOA provide

the ability for applications to respond to changing requirements, because of the layer of abstraction the SOAs provide between the services and the underlying technology.

- *An IT environment organized as an SOA is always amenable to change* – As a growing entity, SOAs provide business value incrementally as they are phased in. Businesses participate in an “ecosystem” containing suppliers, partners, and customers. In fact, a company need not transition all of its application functionality to a SOA to get the benefits of such an architecture. Building an SOA is a progressive approach that can mix service-oriented and non-service-oriented elements.

So, it's clear that SOAs, when applied to the integration challenge, can solve a number of difficult integration issues without cornering the organization into a solution that is too rigid to meet continuously changing business needs. However, how does semantic, or information, integration fit into this mix?

SOAs Raise the Importance of Semantic Integration

Applying SOA to solve integration of application functionality doesn't make all the problems of integration go away, it simply brings the issue of semantic integration to the surface. Companies should be thinking of semantic integration in the context of SOA: using the techniques of encapsulation and loose coupling to make data flow seamlessly.

Simply put, the fundamental assumption of SOI is that standards such as XML and Web Services lower the barriers to interoperability by smoothing the differences among different data representations. Furthermore, loose coupling is a discipline that requires application providers to isolate their specific implementations from the consumers of the information their applications provide. Embedded in the concepts of standards-based, loosely coupled computing is the idea that users won't be required—or even able—to know if the data they are consuming originated in a database, an enterprise application, a file system, another company, or anywhere else for that matter. In fact, in a Web Services-based SOA, the data users consume are entirely decoupled from the source of the data. As SOI removes the barriers to data and application integration we are left with one significant challenge: semantic integration.

The practice of SOA includes different ways of looking at the problems that architecture solves. From information perspective, an information architect focuses on the meaning of the information that moves through the company, who is responsible for it, and what people do with it. The work in this perspective includes identifying how information is created, transported, secured, stored, and destroyed. The data architect takes the data perspective, in which he or she focuses on the taxonomies that the company will use. The bulk of the work in this perspective consists of normalizing the various vocabularies across the enterprise, and understanding and delineating just what data the company wants to use. The end products of the activities in this perspective are the schemas and namespaces that the business processes and the business services they contain will reference.

In order to follow the "Just-in-time" integration style required of today's dynamic integration, the data must be decoupled from any specific technical assumption (such as a specific data schema or format) so that they can be accessed via discoverable, loosely coupled, dynamically bindable Services. Now this requirement doesn't mean that the data shouldn't have any structure at all, it just means that the Service interface hides the details of that structure from the user, and the Service interface itself is dynamically created based on the context of the Service requester. Fortunately, we have a bit of a head start: XML provides

Companies should be thinking of semantic integration in the context of SOA: using the techniques of encapsulation and loose coupling to make data flow seamlessly.

the technical means to isolate the specifics of a data format from the consumer of the data. Web Services in turn provide the means to discover and understand how to consume the data.

How the Operational SOII Approach Differs from Other Approaches

SOII approaches differ considerably from other integration approaches. As detailed above, SOII approaches mandate dynamically bindable, discoverable, standards-based interfaces that transcend the tightly-coupled, proprietary, and brittle adapter-based, ETL, EAI, and MOM solutions of yesteryear. Furthermore, SOII approaches mandate the use of SOA as an architectural guiding principle. Rather than imposing additional development burden on the user, as did BPM and model-driven approaches to integration, SOII enables standards-based, distributed, loosely coupled, and coarse-grained interactions through basic architectural practices.

This operational SOII approach to integration, in which integration is performed on an as-needed, *ad hoc* basis without requiring previous setup by developers, has a number of advantages over previous approaches as detailed in the below table:

Table 3.1: Comparison of Integration Approaches

Integration Approach	Advantages	Disadvantages
Custom Integration (custom code)	<ul style="list-style-type: none"> ➤ Simple to implement ➤ No need for additional server, resource investment 	<ul style="list-style-type: none"> ➤ Complexity increases dramatically as scope of integration increases ➤ High cost of change ➤ Any changes would have to be recoded, so very tightly coupled
Adapter-based Integration	<ul style="list-style-type: none"> ➤ Low cost ➤ Little change to existing application logic 	<ul style="list-style-type: none"> ➤ Synchronous only ➤ Hard to integrate multiple data sources ➤ Tightly coupled
Extract-Transform-Load (ETL)	<ul style="list-style-type: none"> ➤ Good for point-to-point movement of data to warehouses, and other batch-mode movement ➤ Time-tested, proven for large amounts of data movement ➤ Both synchronous and asynchronous integration 	<ul style="list-style-type: none"> ➤ Fundamentally point-to-point, tightly-coupled ➤ Not appropriate for B2B or cross-organizational integration ➤ Difficult to maintain in environments of business change
Traditional EAI / B2Bi Integration	<ul style="list-style-type: none"> ➤ Well-established ROI for controlled integration scenarios ➤ Integrates well with business logic ➤ Robust, rules-based transformations ➤ Hub-and-spoke or bus models for scalability 	<ul style="list-style-type: none"> ➤ Proprietary and expensive ➤ Complex to manage ➤ Not as appropriate for data centric activities ➤ Difficult to implement in B2B scenarios ➤ Does not justify investment for small number of data sources, end points, or applications. ➤ Not suited to incremental implementation.

Message-Oriented Middleware (MOM)	<ul style="list-style-type: none"> ➤ Handles asynchronous and synchronous needs well. ➤ Loosely coupled ➤ Provides guaranteed, end-to-end message delivery. 	<ul style="list-style-type: none"> ➤ By itself does not provide necessary solutions for information integration ➤ Integration is tightly coupled at the message level. ➤ Doesn't handle data aggregation from multiple sources ➤ Difficult to implement in B2B scenarios
Business Process Modeling / Business Process Integration	<ul style="list-style-type: none"> ➤ Process loosely coupled from business logic and data ➤ Asynchronous and synchronous modes ➤ Lower cost than EAI 	<ul style="list-style-type: none"> ➤ Standards immature ➤ Missing robust security, reliability, transaction support ➤ Process platforms not focused on the aggregation of information between systems, rather than a process flow. ➤ Difficult to implement where there is no central process control or definition.
Model-driven Integration	<ul style="list-style-type: none"> ➤ Loosely coupled, coarse-grained integration ➤ Can provide bi-directional integration with multiple data sources ➤ Doesn't require changes to business logic 	<ul style="list-style-type: none"> ➤ Challenging to implement in changing environments of limited data source knowledge. ➤ Requires upfront work to model the data sources. ➤ Not necessarily standards-based.
Service-Oriented Information Integration	<ul style="list-style-type: none"> ➤ Standards-based ➤ Loosely coupled ➤ Asynchronous and synchronous ➤ Allows ad-hoc integration and unbounded queries ➤ Best use for SOII is in the one-to-many, bidirectional environment of change. 	<ul style="list-style-type: none"> ➤ SOA infrastructure in most companies still immature ➤ Requires mapping of data sources to XML formats. ➤ Most appropriate for incremental deployment.

Source: ZapThink, LLC

IV. The XAware SOII Solution

Looking to address the needs of Service-Oriented Information Integration (SOII), XAware introduced in 2000 its *XA-Suite*, a powerful, flexible, and easy-to-use platform that leverages XML, Web Services, and Service-Oriented Architecture (SOA) to enable data integration and data migration. Through the use of a unique construct called *XML Views*, XAware enables users to treat many data sources as if they were a single logical source. XML views enable the bi-directional exchange of information with a wide range of data sources, systems, and applications, enabling integration both internally and externally.

The XA-Suite contains components that contribute to implementing the SOII vision detailed above:

- *XA-Designer* is XA-Suite's design environment used to create, test, and deploy XML Views, known in XA-Suite nomenclature as *BizViews*. Available as a Java Swing IDE, the environment provides visual, drag-and-drop data integration definition, wizards and template-based tools to speed the creation of XML Views and their mappings to enterprise data. Mappings created in XA-Designer are reusable, allowing enterprises to map data sources to their XML representation just once for use in any number of BizViews.
- *XA-iServer* runs and executes the BizViews in order to accomplish the required data integration goals. Available as a Java servlet for execution in any J2EE application server or as a standalone server, XA-iServer implements the complex integration operations of XAware's platform. Leveraging application servers, XA-iServer is multi-threaded, scalable, clusterable, and offers high volume transaction processing capabilities.
- *XA-Adaptors* provide a means for XA-iServer to communicate with enterprise data sources and applications via their native interfaces. The XA-Suite can communicate with over 250 data source types including RDBMS, Mainframes, ERP, CRM, XML databases, and other data sources.
- *XA-Application Interfaces* enable applications to communicate with XA-iServer via various protocols. Application Interfaces enable deployed BizViews to be accessed on demand via protocols such as Web Services and other access mechanisms.

XAware's XML Views treat many data sources as a single logical source and enable the bi-directional exchange of information across disparate applications, supply chains and trading partners, both internally and externally to an organization.

How does XAware's XA-Suite work?

The core of the XA-Suite is the notion of the *XML View*, or *BizView*, which is itself a set of XML metafiles that describes how to accomplish a particular data integration task. XML Views treat many data sources as a single logical source and enable the bi-directional exchange of information across disparate applications, supply chains and trading partners, both internally and externally to an organization.

Users first create BizViews by importing a document schema or creating an "instance" document (a static instance of the schema) through a tool such as XMLSpy. Once the user creates this instance of an XML document, known as the BizDocument, the user adds dynamic elements such as input parameters and other variable elements to help interpret the view and serve up the appropriate data integration activity. The user then inserts these elements into the BizDocument metafile as specialized XML tags that result in dynamic XML documents.

Users then create *BizComponents* that identify the source of information for the BizView document. This data source can be any of 250 data sources that XA-Suite can communicate with, including databases, messaging sources, applications, or any other data source that sends information. In addition to identifying the source of data for a given data integration activity, the BizComponent allows users to specify additional parameters. BizComponents can translate between XML Views and data source-specific integration queries, such as SQL statements. In addition, the BizComponent can transform and combine data integration results as necessary using a wide range of methods, including XSLT and built-in logical operational elements known as "functoids."

The Common Information Model (CIM) is a conceptual information model for describing information and management of that information that is not bound to a particular implementation.

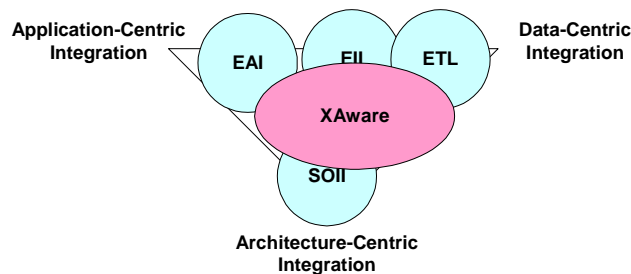
Also, BizComponents enable users to define their bi-directional read and/or write data flow capabilities.

The BizComponent utilizes *BizDrivers* to connect physically to the data source. These BizDrivers also enable design-time users to visualize data within data source to enable drag-and-drop mappings between physical data sources and XML data elements. There is a one-to-one relationship between a BizDriver and a data source. Finally, users create the BizView by aggregating BizDocuments, BizComponents, and BizDrivers together for a particular data integration activity. XA-Suite then exposes the BizViews to consuming client endpoints in a few different ways, including as Web Services, Enterprise Java Beans (EJBs), or through an industry standard XML over HTTP interface.

XML Views are also composable at a variety of levels. First, users can aggregate BizDocuments with other BizDocuments to create higher-level integration activities. Second, users can also aggregate BizComponents to combine integration activities from multiple sources. Users can combine individual data sets retrieved from data sources at the BizDriver level to aggregate information within a particular data source. Finally, the system can aggregate any result with arbitrary Web Services requests, allowing the system to connect to external systems.

The system also allows for "data chaining" in which data integration results from one system can be used in the context of the heterogeneous join, as well as "data shredding" in which a single XML document can be decomposed into multiple non-XML data sources. The system supports transactions at the XML View layer, so if something fails on one system, the user can roll back operations on all systems. Finally, XA-Suite supports rich role-based security as well as selective data encryption, and can integrate with LDAP, single sign-on systems, and other security mechanisms.

Figure 1: XAware's Market Position



Enabling *Ad Hoc* Integration through the Common Information Model

The XA-Suite leverages XML as the means to provide loosely-coupled data integration by providing a metadata-based model of the information to be integrated, known as the *Common Information Model* (CIM). As opposed to the rigorous data and information models required of model-driven integration approaches, the CIM is a conceptual information model for describing information and management of that information that is not bound to a particular implementation. This model allows for the exchange of information between management systems and applications.

CIM-based exchange can be successfully used for exchange networks, migration, and synchronization, and is ideal when the exchange participants have vastly

different architectures. It abstracts the exchange process into two phases: extraction and transformation of source data to the CIM format, and transformation and loading from the CIM format into the target environment. This abstraction into two phases makes the overall process easier to implement, and also facilitates reuse. For example, using CIM as a means to migrate business information from a legacy system into a new application creates both the source to CIM transformation, and the CIM to target transformation. Each of these transformations can be used for other purposes. For example, a later migration from a different set of source systems requires only that the source to CIM transformation be developed. The previous CIM to target transformation and loading can be entirely reused.

XA-Suite leverages CIMs to dramatically reduce the complexity of integration projects. Providing support for Common Models, the XA-Suite supports the implementation of industry standards (CIMs) such as ACORD, RIXML, MDDL, FpML, XBRL, HR-XML, X12, ebXML, MISMO, HL7, and many other current, emerging and custom XML standards.

What makes use of the CIM compelling is that information reuse does not need to be considered ahead of time by the user who wishes to integrate information between systems. Rather, reuse happens on an *ad hoc* basis through the use of metadata models of the information. Users can achieve reuse through domain-specific models (such as vertical vocabularies) as their metadata model.

V. Conclusions

Companies have a wide range of diverse, disparate data sources that they must contend with on a daily basis in order to run their daily operations, make informed decisions, and work with their customers, partners, and suppliers. The challenge companies face is gaining intelligence from their disparate data—and not just a single view of those data, but actionable information from this disparate information. Yet, data integration is still a significant challenge for most firms. Compounding the problem of data integration is the fact that much of the existing technology for accomplishing integration is too brittle, tightly-coupled, and limited in scope to deal with the continuous change that most businesses face.

Standards-based technologies such as XML and Web Services and architectural approaches to integration such as Service-Oriented Architectures represent a new way to solve these seemingly intractable information integration challenges. Rather than attempting to connect systems in a tightly-coupled, point-to-point fashion, Service-Oriented Information Integration (SOII) mandates exposing systems as standards-based Services that can be connected on an as-needed, ad hoc fashion. As a result, companies can implement decentralized data sources that can be connected centrally as needed. In this way, XML provides the glue to normalize information, and SOAs provide the means to connect these systems together.

SOII is not just all promise and hype, however. XAware is unique in providing a pure SOII while allowing prior generations of technologies like CORBA, text structures, mainframe data sources and EJBs to participate in web services based architectures. XAware's solutions offer a unique and compelling solution to firms looking for the advantages of loosely-coupled, standards-based integration without the need for detailed modeling of data sources. In environments of continuous change and dynamic information sharing requirements, SOII, and specifically XAware's XA-Suite provide solid answers to today's information integration challenges.

Copyright, Trademark Notice, and Statement of Opinion

All Contents Copyright © 2004 ZapThink, LLC. All rights reserved. The information contained herein has been obtained from sources believed to be reliable. ZapThink disclaims all warranties as to the accuracy, completeness or adequacy of such information. ZapThink shall have no liability for errors, omissions or inadequacies in the information contained herein or for interpretations thereof. The reader assumes sole responsibility for the selection of these materials to achieve its intended results. The opinions expressed herein are subject to change without notice. All trademarks, service marks, and trade names are trademarked by their respective owners and ZapThink makes no claims to these names.

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

