zapthink

white paper

THE PATH TO SUCCESSFUL WEB SERVICES: NAVIGATING THE PITFALLS

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

THE PATH TO SUCCESSFUL WEB SERVICES: NAVIGATING THE PITFALLS

October 2003

Analyst: Jason Bloomberg

Abstract

Web Services are a reality in today's enterprise, but they're only the first step to building a more agile IT infrastructure. Using Web Services to solve integration problems reduces the cost of integration, to be sure, but the promise of substantial ROI is much greater as companies move toward Service-Oriented Architectures (SOAs). Such architectures can support flexible business processes and offer return on investment in the form of increased business agility.

The best approach to building such an SOA, especially in light of today's tight IT budgets, is to take a phased approach that yields positive ROI at every step. The path from simple applications of Web Services to enterprise SOAs, however, is a difficult one. There are many pitfalls that companies might encounter that left unresolved can dramatically increase the risk and cost of an SOA buildout.

This paper outlines one economically viable path to building an enterprise SOA, and addresses many of the pitfalls that companies are likely to encounter along the way.

Table of Contents

I. Web Services Implementations: the Path to Service-Oriented Architectures 4
SOAs in support of business processes 4
Grass roots/point-to-point Web Services 5
Mission-critical, static Services 6
SOA pilots 7
SOA buildout 7
II. The Pitfalls and How to Avoid Them 8
Lack of visibility into active Web Services 9
New security holes 9
Unexpected demand 10
Operational failures 10
Mismatched versions 11
Lack of business visibility 11
III. Confluent Software: Keeping Enterprises on the Path 12

I. Web Services Implementations: the Path to Service-Oriented Architectures

Web Services are just the first step to building a Service-Oriented Architecture.

Web Services are now a reality. Most enterprises are currently using Web Services in their IT organization, typically to reduce the cost of integration. Many of these companies realize, however, that Web Services are just the first step to building a Service-Oriented Architecture (SOA). Nevertheless, Web Services offer a standards-based approach to integration for companies looking to reduce the cost and complexity of point-to-point integration between systems, independent of any architectural improvements they might make. As a result, many companies have found that Web Services offer an important set of tools and techniques for integrating systems—in particular, heterogeneous systems.

Web Services are typically nothing more than another tool in a developer's integration toolbox.

In many cases, today's integration projects that take advantage of Web Services are not particularly differentiated from those integration projects that use more traditional integration technologies. In other words, Web Services are typically nothing more than another tool in a developer's integration toolbox. A useful tool, to be sure, because many IT shops use open source tools to build Web Services, and the skills needed to use such tools are less sophisticated (and hence, more available and less expensive) than the skills needed to use proprietary integration tools like those that traditional enterprise application integration (EAI) vendors provide.

Web Services offer a standards-based approach to building SOAs that can enable loose coupling between Service producers and consumers and coarse grained, business-oriented Services that offer asynchronous interfaces.

The concept of Service-Oriented Architecture, on the other hand, has been around for many years, and has shown promise as well as limitations. Earlier approaches to SOAs struggled with proprietary technologies, tightly coupled communications, and fine-grained interfaces. Web Services offer a standards-based approach to building SOAs that can enable loose coupling between Service producers and consumers and coarse grained, business-oriented Services that offer asynchronous interfaces. Such SOAs promise increased flexibility and responsiveness to business requirements.

SOAs in support of business processes

To many developers, Web Services are simply another interface to software objects and components. As a result, developers apply their traditional component object design methods, deployment technologies, scalability and reliability approaches, and terminology. The result: point-to-point implementations of Web Services that are every bit as brittle, tightly coupled, synchronous, and fine-grained as their object-oriented predecessors. In essence, these developers are taking a bottom-up view of Web Services where the Service implementation is the center of the world. Instead of thinking of Web Services as a collection of interfaces to software functionality that must somehow be made to connect to other such interfaces, enterprises should approach building Web Services-based SOAs in a fundamentally process-driven architecture that leverages distributed processes as well as distributed Services. Such distributed processes are all about the creation of Service-enabled business processes that in turn depend on other Services or business processes that may be defined anywhere in the organization. Those business processes then at some point depend on fine-grained, atomic Services to fulfill the activities required by their process flow.

Approaching SOAs and Web Services from this "Service-oriented process" perspective simplifies and clarifies many of the troublesome issues relating to distributed computing and Web Services. Integration goes from being a

troublesome chore that must be accomplished by implementing layers of complicated and expensive technologies to a side effect of process execution. In fact, it's difficult to create a Service-oriented process that does not provide the fundamental benefits of application and business integration. The mere act of orchestrating and choreographing a Service-oriented business process achieves most integration goals.

In an SOA, a process is actually a Service.

What is unique about Web Services-based SOAs is that in an SOA, a process is actually a Service. Individual Services can be composed into a process flow that itself can be described as a Web Service. The Services that a process consumes might be Services local to an enterprise or available at a remote location. The ability for processes to consume Services where those processes are also Services provide the key to the agility benefit that SOAs promise, because the Service interface abstracts the service requester from knowing whether a Service is atomic or actually the interface to an orchestrated process. In fact, the primary ROI to be gained from an SOA buildout comes in the form of increased opportunities and reduced costs as a result of Service-oriented processes.

The primary ROI to be gained from an SOA buildout comes in the form of increased opportunities and reduced costs as a result of Service-oriented processes.

Calculating exact figures for the return on investment (ROI) of an SOA buildout can be difficult, therefore, because the success of such an initiative frequently depends upon the increase in business agility such architectures can provide. *Business agility* is the ability of a company to respond efficiently to change, and leverage such change for competitive advantage. Change comes in many forms—changing market conditions, changes in the competitive landscape, a changing regulatory environment, or changes in technology. All such change involves unknown future conditions, and thus business agility implies dealing effectively with the unknown. For this reason, ROI predictions are typically very difficult to calculate before a company begins an initiative. However, the fact that ROI is difficult to quantify doesn't mean that Web Services and SOA initiatives don't offer substantial ROI.

In fact, Web Services offer real ROI today in terms of reduced integration costs, and SOAs offer solid ongoing ROI as the result of increased business agility based on Service-oriented processes. The challenge many enterprises face today is connecting the dots between these two types of initiatives, in light of the constrained IT budgets that face today's enterprises. For some companies, the secret to achieving the long-term business advantages of an SOA is to develop a phased approach that yields a positive ROI at every step. Many companies are showing solid successes taking this approach, as Figure I and the examples below illustrate. Other companies, however, would prefer to move forward with their SOA plans in advance of realizing a positive ROI during the early phases of a gradual rollout. Either way, it's important to understand the pitfalls that an organization is likely to encounter as they roll out Web Services, and how best to resolve those risks.

For some companies, the secret to achieving the long-term business advantages of an SOA is to develop a phased approach that yields a positive ROI at every step.

Grass roots/point-to-point Web Services

Companies often get started with Web Services when small project teams leverage the benefits of Web Services to solve point-to-point integration problems, often under the radar of executive management—what might be called "grass roots" adoption of Web Services. Such grass roots projects typically use Web Services as an inexpensive means for connecting heterogeneous systems. IntegratioIntegration engagements that use Web Services in this manner are typically quite similar to more traditional integration projects that do not use Web Services.

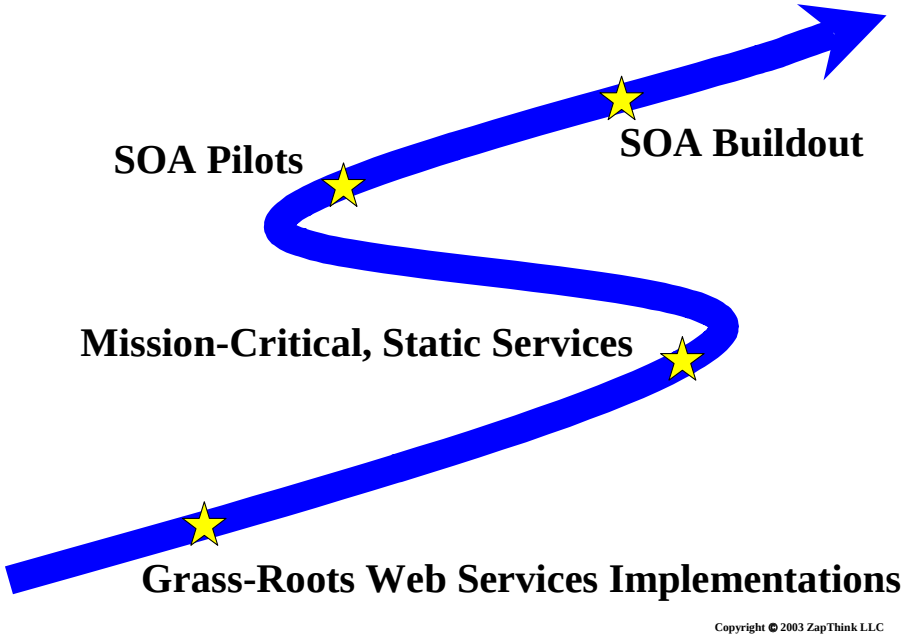

Figure I: Web Services Implementation Path to SOAs

While the number of examples of grass roots Web Services implementations are too numerous to mention, one interesting example is a large media research firm that is using Web Services throughout their product line. Traditionally, this company provided its research data to its customers via either a direct feed from its mainframe systems in batches, or through an awkward client/server application. Customers had to dial into the firm's modem bank to use the client/server app, even though most had direct Internet access. Then they had to download the data in the form of Excel files, which they would have to reformat every time to meet their specific needs.

The internal issues this company faced mainly involved legacy integration, as their core data collection applications resided on mainframes, while their data warehouse was Unix-based. Project teams across the enterprise began to experiment with Web Services as a way of integrating these heterogeneous systems in a more cost-effective and flexible manner. The resulting improvements led to a new customer-facing initiative that involved an n-tier architecture with Web Services links to back-end systems as well as a Web Services front-end that supported a browser interface. The ROI of such an initiative is is measured in terms of increased customer satisfaction and future market growth. This company expects to achieve a substantial return on their Web Services investment.

Mission-critical, static Services

As companies build expertise in Web Services techniques and an understanding of their value, they build individual Services that serve critical roles.

As companies build expertise in Web Services techniques and an understanding of their value, they build individual Services that serve critical roles. This broad application of Web Services to complex integration projects includes n-tier projects like portals and eCommerce engagements, supply chain management projects, and EAI projects that use Web Services extensively throughout the project. Such projects typically expose coarse-grained business Services, but may not offer the location independence and loose coupling of an SOA.

There are several examples of such extensive, coordinated use of Web Services in complex integration projects. An important lesson to be learned from many of these examples is that companies are using Web Services to solve complex integration problems without the need to build a true SOA. One interesting example is a consumer-oriented eCommerce firm who offers a Web Service to their affiliate partners. These affiliates can use this Web Service to build their own browser front-ends to this firm's merchandising and purchase systems. The Web Services involved are quite straightforward—URL-based Services with published, static interfaces. Even though the Web Services are technically simple, this firm has generated substantial revenue through their affiliate channel via their Web Services initiative. This company's ROI on their Web Services initiatives depends on the fact that their investment in Web Services is far exceeded by their affiliate-related revenues, because their 900,000+ affiliates offers them enormous advantages of scale..

SOA pilots

Even pilot projects must show solid ROI as well as build acceptance for SOAs in the organization and expertise among the IT staff.

Once Web Services have achieved a certain level of acceptance in the organization, architecture teams or other key IT personnel should initiate pilot projects leveraging the dynamic discovery, location independence, and loose coupling capabilities of SOAs. Even though these initiatives are pilot projects, they must still show solid ROI as well as build acceptance for SOAs in the organization and expertise among the IT staff. Such pilots are true SOA engagements in that they are projects that include architectural guidance at the enterprise level, providing a framework for building Services within the context of an SOA.

The difference between grass-roots Web Services projects and mission critical, static Web Services initiatives is mostly one of degree, but those two are fundamentally different from SOA pilot projects, because the first two are integration projects, while SOA initiatives are focused on architecture. Therefore, the skills, knowledge, and tools of the respective project teams are quite different.

One notable example of an SOA pilot is an international logistics provider building an SOA on top of their existing EAI messaging infrastructure. Their EAI solution connects multiple international locations and forms a mission-critical backbone for their business, prompting them to build an SOA to provide an abstraction layer on top of the messaging backbone. They crafted a handful of coarse-grained Services with the goal of enabling reuse of these Services across the enterprise. However, in this pilot phase, they weren't ready to provide for the discoverability of these Services, and they also wanted to strictly control the consumers of the Services. So while the Service producers and consumers were not truly loosely coupled, they still achieved substantial cost benefits from building reusable Web Services as part of the SOA abstraction layer of up to 50%.

SOA buildout

As companies complete successful pilot projects, they typically look to leverage their assets and expertise to build out comprehensive, enterprise SOAs that abstract IT functionality both within the enterprise and among business partners, enabling Service-oriented processes.

SOA pilot projects serve several purposes. First and foremost, they offer an approach to yielding a positive ROI in a short timeframe. In addition to this primary advantage, however, SOA pilot projects help to build acceptance across the organization for Service-oriented approaches to IT. Pilot projects also help the IT staff build the necessary skills, and provide a gradual approach to purchasing new software to support a corporate SOA.

As companies complete successful pilot projects, then, they typically look to leverage their assets and expertise to build out comprehensive, *enterprise SOAs*

that abstract IT functionality both within the enterprise and among business partners, enabling Service-oriented processes. Sometimes the transition to an SOA that offers dynamic discovery and integration to internal and external consumers comes about as the result of an increasing number of Services that a growing number of users wish to access. In other cases, companies decide to build out a full-featured SOA soon after completing a small number of pilot projects. Dynamic discovery of Services becomes increasingly important when companies build out their SOA, as the number of available Services increases.

Take for example a prominent investment firm that built an enterprise-class SOA to expose thousands of mainframe transactions. Instead of gradually rolling out Services, they decided to build an SOA that offered Web Services that were discoverable in a UDDI registry. They coded thousands of Services in directly in COBOL to provide interfaces to existing CICS transactions.

This company improved the response times for CICS transactions by 10 to 20 times, by building a sophisticated caching mechanism that brought information from the mainframes into a centralized location. They also increased throughput by a factor of 10 times, and reduced the percentage of cost for infrastructure and integration from 90% to 65%, increasing the percentage devoted to business logic to 35%, by migrating middleware functionality from Intel-based servers to Linux virtual environments on the mainframe. This migration also reduced software licensing costs. Finally, they reduced the time to write programs to get data off of mainframe from 3-4 months to a matter of minutes, by exposing CICS transactions as Web Services that were discoverable via a UDDI registry.

II. The Pitfalls and How to Avoid Them

Pitfalls can occur quite early in a Web Services implementation, even when there are only one or two Web Services in production.

Taking a step-by-step approach to building an SOA can reduce risk and increase the return on investment in new technology, but the fact still remains that there are a number of pitfalls to building an SOA that can lead to increased risk and lowered ROI. In fact, as Figure II below shows, these pitfalls can occur quite early in a Web Services implementation, even when there are only one or two Web Services in production. It's also important to keep in mind that building SOAs isn't easy. There are many aspects of an SOA initiative that can go wrong. Understanding these pitfalls and how to mitigate them is essential to achieving the desired ROI from any SOA initiative.

As Figure II below illustrates, pitfalls can take place at any of the four phases on the path to implementing Web Services and building an SOA.

	Grass Roots Web Services Implementations	Mission-Critical, Static Services	SOA Pilots	SOA Buildout
Lack of Visibility into Active Services	√	√	√	√
New Security Holes	√	√	√	√
Unexpected Demand		√	√	√
Operational Failures		√	√	√
Mismatched Versions			√	√
Lack of Business Visibility				√

Figure II: Web Services Implementation Pitfalls

Fortunately, for each of the six categories of pitfalls facing IT teams building out their Web Services initiatives, there are approaches for addressing those pitfalls and thus lowering the risks inherent in building an SOA. These approaches are discussed below.

Lack of visibility into active Web Services

The first pitfall involves the lack of visibility IT managers can have into what's going on with Web Services in their organizations. IT management must get a handle on the grass roots Web Services efforts in its shop before they get out of hand, consuming resources unexpectedly or leading to needless repetition of work. Without such visibility and the associated oversight, IT organizations face lost opportunities for IT improvement, as well as potential costs and issues when unexpected problems occur, for example, when it takes IT staff a long time to become aware of Service issues.

Traditional network and system management software cannot distinguish Web Services traffic from other forms of network traffic.

The solution to addressing the lack of visibility is to provide that visibility by implementing a Web Services monitoring solution. Because Web Services traffic appears on the network as text messages often going over HTTP, traditional network and system management software cannot distinguish Web Services traffic from other forms of network traffic, including Web site traffic. It is important, therefore, for companies who are just getting started in their Web Services buildout to implement a Web Services monitoring solution that passively manages network traffic at the application layer, where it's possible to distinguish Web Services traffic from other kinds of messages.

Monitoring Web Services remains important regardless of how far along the Web Services implementation path a company is. As companies build SOAs, the Services they must monitor include both the fine-grained Services that act as interfaces to heterogeneous software functionality, as well as the coarse-grained business Services that the line-of-business uses to access the business processes that run on the SOA. They must also monitor Services regardless of the Services' locations in order to have visibility over the complete transaction as it moves from Web Service to Web Service.

New security holes

The second pitfall companies face when rolling out Web Services is the new security holes such Services introduce. Traditional perimeter and application-centric security leave big holes when Web Services provide new interfaces to existing application functionality. After all, if a Web Service runs on port 80 (the Web port), then firewalls typically allow all traffic to cross the port. However,

companies face the compromise of sensitive corporate information if they expose Web Services on the network—externally, to be sure, but also internally.

Even for companies with a few Web Services in production, it is critically important to understand whether the users who are consuming those services are authorized to do so.

The solution to this pitfall is to invest in a Web Services security management solution. Even for companies with a few Web Services in production, it is critically important to understand whether the users who are consuming those services are authorized to do so. Furthermore, as companies build out their SOA initiatives, Web Services security becomes even more important, as the Services abstract underlying functionality that might reside in several applications.

As companies build out their SOAs, the security issues become more complex. Securing open, loosely coupled systems in an SOA requires a much more sophisticated security approach than traditional distributed computing architectures, involving multiple administrators that support distributed users. Different systems now have different policies and possibly different security mechanisms. As a result, administrators within the enterprise must manage security much more actively than was necessary in the traditional application security model.

Unexpected demand

One of the great benefits of implementing Web Services is that many different consumers can reuse those Services. The downside to this reuse is, well, that many different consumers can reuse those Services—even consumers that the team responsible for the Service weren't expecting. Because Web Service producers are loosely coupled from their consumers, a Service can experience an unexpected demand as consumers increase their use of the Service. The obvious risks that companies face as their Services find new users is an increased risk of downtime or lower performance for critical users or Web Service consumers.

Traditional application servers can address some of the risks of downtime that result from unexpected Web Services traffic—but not all of the risks. In the early phases of Web Services rollout, an application server is often sufficient if those Services are being run on the application server. However, as companies move toward SOAs that abstract the underlying functionality, a particular Service may touch multiple applications and platforms, and any single application server may not be sufficient to handle unexpected surges in traffic. The solution to this problem lies in an active Web Services management solution that provides dynamic routing, load balancing, and prioritized messages. Such Web Services management solutions actually enable the SOA by actively realizing the layer of abstraction that SOAs must provide to yield their desired agility benefits.

A particular Service may touch multiple applications and platforms, and any single application server may not be sufficient to handle unexpected surges in traffic.

Operational failures

While layers of abstraction mask the underlying complexity of the technology, providing a simpler, more powerful interface to the business consumers of the Services, such layers of abstraction also have a downside. What if something just stops working? The Services abstraction layer can make solving the problem difficult and time consuming. After all, Web Services are application software running on systems at their core, and if a piece of hardware, or an application, or a network component stop working properly, the Services that depend on it may also fail. Such operational failures can lead to angry customers, increased liability, and diminished revenues.

Web Services are application software running on systems at their core.

The solution to dealing with operational failures is to implement a management system that offers root cause analysis. Root cause analysis is a common feature of traditional system management products, but these products often cannot

apply the root cause analysis to business-oriented Web Services that access coarse-grained functionality. It is important, therefore, for the management tool to have the capability to identify the root cause of problems with the abstracted Services layer. A management solution should also take corrective action immediately in the case of failure in order to maintain desired Service levels and responsiveness.

Mismatched versions

In the case of a complex, enterprisewide SOA, companies can expect Web Services updates to be a routine occurrence.

As the number and complexity of Web Services in the enterprise increases, it becomes increasingly likely that IT personnel will need to update individual Web Services. Therefore, the risk that some Web Service consumer application will break because a Service has unexpectedly changed increases as well. In the case of a complex, enterprisewide SOA, companies can expect Web Services updates to be a routine occurrence. If the IT shop doesn't have an automated approach to dealing with such changes, then any updates to Web Services promise to be time consuming and expensive. Clearly, Web Service consumers must be insulated from such changes for the SOA to be effective.

The solution to the problem of mismatched versions of Web Services is to implement a change management solution. Such a solution should be able to support multiple Web Services versions in production simultaneously, automatically routing requests to the appropriate version.

Lack of business visibility

The first pitfall we considered was a lack of visibility among IT managers into the Web Services in operation. However, as companies orchestrate Web Services into business processes, line-of-business managers also need visibility into those processes, which are themselves Web Services. Without business visibility into the Service-oriented processes enabled by the SOA, line-of-business managers risk poor business decisions that could be avoided with current, detailed information.

The solution to the lack of business visibility is to implement a business activity monitoring solution. Business activity monitoring has had limited success up to this point, because existing IT approaches have only been able to provide the ability to monitor certain aspects of the business. With an enterprise SOA, however, business activity monitoring can provide increased visibility into a wide range of business processes supported by the architecture.

With an enterprise SOA, business activity monitoring can provide increased visibility into a wide range of business processes supported by the architecture.

In fact, the move toward SOAs blur the distinction between integration approaches and business process automation. Whenever multiple applications or organizations behave as if they were part of a cohesive and well-integrated architecture, many dimensions of business process come into play. A goal of many large organizations is to make the various applications, repositories, and even roles or organizations appear to be well aggregated and integrated even though they are discrete entities distributed across many departments and other enterprises. The primary way to make sure that all the various components and resources work together is by coordinating them through formalized business process.

As companies implement SOAs, the enterprise architecture will automatically provide integration capabilities because it will be process driven.

It's quite possible that a few years from now, companies won't be spending so much of their time and money on integration. Instead, as companies implement SOAs, the enterprise architecture will automatically provide integration capabilities because it will be process driven. Enterprises will then be able to focus on more important aspects of their business: modeling their business requirements and processes, and building the IT systems that meet their

business goals. In order for this vision to become a reality, however, companies must avoid the pitfalls of the lack of visibility into active Services, new security holes, unexpected demand, operational failures, mismatched versions, and the lack of business visibility into the Web Services in operation.

III. Confluent Software: Keeping Enterprises on the Path

Confluent Software offers a Web Services management solution that addresses all of the pitfalls discussed in this paper. Confluent enables SOAs by providing a management infrastructure that supports the layer of abstraction that SOAs represent. In effect, SOAs have two levels: the business level, where Service consumers can invoke business-oriented Services in a loosely coupled fashion, and the technology level, where the management infrastructure provides the capabilities necessary to insure that loose coupling.

Confluent tackles these pitfalls discussed above by offering a policy driven management platform for consistently enforcing operational policies over distributed Web services and other software assets. Confluent also has a distributed architecture, working through a set of active intermediaries called either *gateways* or *agents* depending on their configuration.

Gateways are essentially proxy servers that intercept requests, enforce policies, and then forward requests to Services that are registered within Confluent. Web Service consumers must be directed at the gateway instead of the Service for the gateway to act as a proxy for the Service, while agents are policy-enforcement plug-ins that are meant to be deployed inside a SOAP container. As a result, Service consumers are unaware of the message interception that is necessary for Confluent to enforce policies.

Confluent consists of three components: Policy Manager, Monitor, and Analyzer. Together, these components bring consistent policy control and enforcement to distributed Web Services. Confluent is organized around the idea of *policy pipelines*, which are repositories of policies used to control the operational behavior of managed Web Services. These policy pipelines manage several aspects of Web Services, including authentication and authorization, quality of service, load balancing, message transformation, and service-level requirements. IT managers register new Services in Policy Manager, which is also where they create, distribute and manage policy pipelines for those Services. The policy pipelines thus handle the pitfall of unexpected demand.

Confluent Policy Manager also controls several important aspects of Service delivery as well. It supports HTTP as well as messaging-based Service delivery, and it allows the user to configure what happens in the event of Service failure. Another important feature of Confluent Policy Manager is its change management capabilities. As IT managers add Services, they can include a version number with each new Service. As consumers request particular Services, then, Confluent automatically routes requests to the most recent compatible Web Service.

Confluent Monitor provides the operational aspects of the Confluent solution. With Monitor, system administrators can monitor the availability, security, and performance of all the Web Services that Confluent manages, providing the required visibility for IT management as well as the required Web Services security. Confluent can then monitor access-control violations, message traffic, and Service latency. Having such detailed information about the Web Services under management is useful for root cause analysis.

Confluent offers the business visibility companies need to implement agile enterprise SOAs.

The final component of the Confluent solution, Confluent Analyzer, aids in the collection and reporting of business analysis data through the use of rules that reach inside each SOAP message body and apply Boolean expressions to the message parameters. Users can then create reports on these parameters. Therefore, Confluent also offers the business visibility companies need to implement agile enterprise SOAs.

Avoiding the pitfalls with Confluent Software

Confluent Software offers a comprehensive solution that helps companies address the pitfalls they are likely to encounter when building out their Web Services and SOA initiatives.

In summary, Confluent Software offers a comprehensive solution that helps companies address the pitfalls they are likely to encounter when building out their Web Services and SOA initiatives:

- *Lack of visibility into active Services* – Confluent Monitor enables companies to monitor the availability and performance of active Web Services
- *New security holes* – Confluent Monitor also provides the ability to monitor the security of active Services, and Policy Manager enables companies to apply security policies to their active Services.
- *Unexpected demand* – Confluent's policy pipelines enable companies to manage quality of service, load balancing, and service-level requirements, providing the ability to respond quickly to unexpected demand.
- *Operational failures* – Because Confluent provides a broad range of detailed information about the Web Services in production, companies are able to perform root cause analyses of operational failures, reducing downtime to a minimum.
- *Mismatched versions* – The change management capabilities of Confluent Policy Manager enables IT managers to include version numbers with new Services, so that consumers can request the proper version of each Service.
- *Lack of business visibility into the Web Services in operation* – Confluent Analyzer helps business users collect and analyze business data contained in Web Services messages.

Companies need management when they launch their first important Web Service.

As companies move from grass roots Web Services implementations to mission-critical, static Services, and then launch SOA pilots and build out their enterprise SOAs, Confluent Software can provide the essential management capabilities every step of the way. It's important to remember, however, that companies need management when they launch their first important Web Service—a company isn't in control of their information technology if they don't tackle management at the beginning of their Web Services rollout. Confluent Software is able to help companies at all points on the Web Services implementation path, including the early phases.

Take Credit for Reading ZapThink Research

Earn rewards for reading ZapThink research! Visit www.zapthink.com/credit and enter the code PITFALLS. We'll reward you with ZapCredits that you can use to obtain free research, ZapGear, and more!

Copyright, Trademark Notice, and Statement of Opinion

About ZapThink, LLC

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

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

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

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

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

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

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