Doc. ID: ZTF-WS115-1
Released: February 20, 2003

zapthink
focus report

SERVICE-ORIENTED ARCHITECTURE TOOLS

BEYOND POINT-TO-POINT WEB SERVICES

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

SERVICE-ORIENTED ARCHITECTURE TOOLS

BEYOND POINT-TO-POINT WEB SERVICES

February 20, 2003

Analyst: Jason Bloomberg

Abstract

From its inception through 2002, the primary application for Web Services in the enterprise was to simplify point-to-point integration between systems, thereby reducing the cost of integration. This application of Web Services, however, only scratches the surface of the true potential of Web Services -- enabling companies to build agile business processes and IT systems that can respond to change through the use of loosely coupled, standards-based Service-oriented architectures.

The business value of such architectures in terms of the business agility they provide is substantial, but as of early 2003, only a few early adopter enterprises have built such architectures, partly because few tools for building Service-oriented architectures are available on the market, and furthermore, there is little understanding of the best practices companies should follow to build such architectures. This report seeks to clarify the requirements for realizing the value of Web Services by providing a set of emerging best practices as well as an analysis of the tools that are currently available for building Service-oriented architectures.

Key Points:

✦ Market Overview

- Service-oriented architectures built upon open, standards-based Web Services provide a strategic IT direction businesses need to meet their fundamental business goal: agility.

✦ Facts & Figures

- By 2010, ZapThink expects 69% of the total enterprise software market to be Service-oriented.
- The overall market for products and services that support Service orientation will be over $98 billion by 2010.

✦ Analysis

- Reworking existing brittle, high-cost IT infrastructures into flexible, Service-oriented architectures promises substantial long-term cost savings and revenue opportunities through increased business agility.

✦ Future Trends

- Service orientation represents the latest distributed computing approach to affect IT – the fourth major shift since the mid-twentieth century.
- ZapThink predicts that companies will begin to accept Service orientation in 2003, and it will become the dominant distributed computing approach by 2006.

Table of Contents

I. Report Scope

Analyses of emerging markets present several challenges to the researcher: products are immature, customers are rare, and confusion is widespread. Perhaps the greatest challenge is balancing vision with execution: too much vision without practical detail becomes hype, while detail at the expense of a coordinating vision leads people to miss the forest for the trees. Such is the challenge with Web Services. This report seeks to balance long-sighted, strategic vision and short-term, tactical execution in the context of the Web Services marketplace.

Today's decision maker wants tactical solutions to current problems, in particular when there is money to be saved.

Partly in response to the tough economic climate, and partly in reaction to the dot-com boom and bust, today's businesses tend to err on the side of conservative realities rather than unproven promises. Business and IT decision-makers have become jaded regarding discussions of IT that attempt to paint a seemingly unrealistic big picture for the future, especially when those discussions lead to money being spent in the present. Today's decision maker wants tactical solutions to current problems, in particular when there is money to be saved. Ironically, this cynical environment has given birth to Web Services.

Web Services, however, are only the trees; there is still a forest that people are missing—the over-arching vision that gives Web Services strategic business value. That vision is *Service orientation*. Open, standards-based Service-oriented architectures built with Web Services form a matched set of technical principles that can provide companies with the strategic IT direction they need to meet their fundamental business goals. Without such a direction, companies risk losing track of the long-term decisions they must make to remain competitive—the essence of the "forest for the trees" problem.

Decision Point

Open standards-based Service-oriented architectures built with Web Services form a matched set of technical principles that can provide companies with the strategic IT direction they need to meet their fundamental business goals.

Of all the goals that enterprises have today, the one business imperative that Service orientation addresses most directly is the need for *business agility* Business agility is more than simply being able to respond quickly to change; it also means the ability to leverage change for competitive advantage. For businesses to be agile, their technology must be agile as well. Service orientation, therefore, provides the guidelines for building an IT infrastructure that is agile enough to respond to the needs of the agile business – finally putting business in control of IT, rather than vice-versa.

This report, balances the Service orientation vision with real-world execution advice that companies can implement today. It begins with a clear definition of Service-oriented architectures, and then details several best practices enterprises should follow to implement and maintain such architectures. The report will also discuss the tools available on the market for building these architectures, and the vendors who produce those tools.

The report specifically does not cover the following topics, although they may be mentioned in the context of discussing Service-oriented architectures:

- Web Services development tools or platforms. Because most software development tools now support Web Services to some extent, there does not exist a separate emerging market for such tools. Rather, this report identifies the critical tools that enable *Service orientation* beyond simple point-to-point Web Services enablement.
- Service-Oriented Integration, EAI, or B2B Integration solutions (see ZapThink's *Service-Oriented Integration (SOI)* Report [ZTR-WS103]).

- Service-Oriented Process solutions, including choreography, business process automation, and workflow (see ZapThink's upcoming *Service-Oriented Process* Report [ZTR-WS108]).

This report is intended for both the enterprise user as well as the IT vendor. Enterprise users should find the best practices and tools discussion applicable to their Web Services implementation and planning activities. Vendors should gain a clear picture of how enterprise needs will develop as Service orientation becomes prevalent in the enterprise.

II. Context for Service-Oriented Architectures

To place Service-oriented architectures into the proper context, it is important to pull together some important conceptual threads that cross today's IT environment. The relevant threads include concepts as diverse as distributed computing, software development, enterprise architecture, and the movement toward open standards. Each of these threads contributes to the inevitable movement towards service-oriented architectures.

As marketplaces evolve, there comes a time in their evolution where the benefits of cooperating in a standard manner outweigh the advantages of conducting business in a proprietary manner.

The movement toward standards-based computing is the logical starting point, because of the role standards play in the world of business. As marketplaces evolve, there comes a time in their evolution where the benefits of cooperating in a standard manner outweigh the advantages of conducting business in a proprietary manner. While in many situations there are clear economic incentives for vendors to lock in customers with proprietary practices, as a

market evolves, competitors eventually offer sufficient selection and quality to afford end-users choice. At that point, competitors who cooperate on establishing standards can provide themselves with a competitive advantage over those vendors who continue to offer proprietary solutions, since customers no longer have to lock themselves in to a particular vendor. The shift to leveraging standards is typically quite rapid when this tipping point is reached, as market participants who do not participate are quickly left out in the cold.

It is not sufficient, however, for vendors alone to get together to decide upon standards. A standard is just ink on paper or bits in a file until vendors and customers can agree on how to do business in a standard way. The point here is that it is the *business*, not the standard by itself, that is of critical importance: only when buyers and sellers actually use the standards to conduct day-to-day business can they realize the competitive advantages of the standards.

The world of distributed computing is currently undergoing the transition from proprietary approaches to standards-based approaches.

The world of distributed computing is currently undergoing this transition from proprietary approaches to standards-based approaches. XML lies at the heart of this movement toward open standards, because of its ability to flexibly represent information that can be interchanged in a standard manner. Using XML as a foundation, Web Services provides a new approach to distributed computing that help to standardize system interfaces and encourage loosely-coupled communication among systems. Web Services, however, are the trees, but not the forest: by themselves, Web Services are little more than a standards-based way of integrating systems that would otherwise be similarly connected with proprietary interfaces. The real win for companies that leverage Web Services results from when they rethink their approach to distributed computing—taking a new approach known as Service orientation.

Service orientation is an evolutionary approach to distributed computing in which software functionality is made available as business-oriented Services on the network. ZapThink believes that Service orientation is the next logical step in the development of distributed computing technologies, following in the footsteps of previous distributed computing efforts. And while distributed computing is an essential part of IT, inextricably intertwined with how business is conducted today, it is also the backbone of the economy as a whole. Companies aren't interested in investing in proprietary, centralized computing solutions any more. Any company that has systems that talk to other systems over a network has a vested interest in conducting business via standards-based distributed computing—in other words, a vested interest in Service orientation.

Decision Point

Any company that has systems that talk to other systems over a network has a vested interest in conducting business via standards-based distributed computing—in other words, a vested interest in Service orientation.

Just as Web Services are an evolution of traditional distributed computing techniques, the practice of Service-oriented architecture is an evolution of the practice of enterprise architecture, which is an aggregation of all the individual IT systems within an organization, as explained in Section 0. The potential rewards for enterprises that understand this evolution and make the move to such architectures are enormous—finally, distributed computing technology promises to be flexible and nimble enough to respond to business needs and provide the business agility that companies have craved for so long, but which has always been out of reach.

Web Services have moved beyond the hype stage and are now a reality for many enterprises. Throughout the past two years, ZapThink has seen evidence that companies have begun to take tentative steps towards implementing Web Services in ways that provide incremental improvements over existing technologies. Hundreds of companies of different sizes and industries have built Web Services pilot projects, proving that this most recent evolution of distributed computing technology can reduce integration and development costs substantially. Forward-looking enterprises are now looking to take the next step

and figure out how to leverage the power of Web Services strategically across the enterprise.

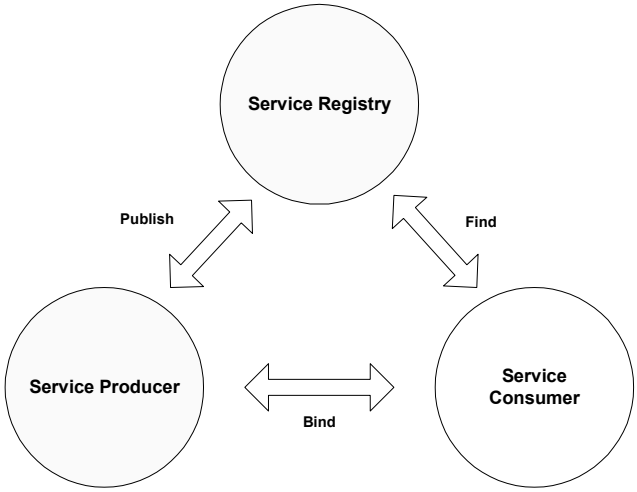
This next step means moving beyond simple point-to-point applications of Web Services to a broad application of Web Service technologies both within the enterprise and increasingly among business partners. This transition requires more than a simple change in programming practices. This broad application of Web Services technologies requires an architectural change—a move to loosely coupled, standards-based Service-oriented architectures. This new architectural approach requires a different perspective on the role of IT in the organization, and a new kind of enterprise architect—the *Service-oriented architect*. The Service-oriented architect must understand the practice of Service-oriented architecture.

2.1. What is a Service-Oriented Architecture?

Service-oriented architectures *are an approach to designing distributed computing infrastructures that thinks of software resources as services available on a network.*

Service-oriented architectures are an approach to designing distributed computing infrastructures that considers software resources as services available on a network. Producers of these services must be able to publish information about them in a service registry, where service consumers can then look up the services they need and retrieve the information about those services they need to bind to them. This "publish-find-bind" triangle forms the core of a Service-oriented architecture, as shown in Figure II.1 below:

Figure II.1: The SOA Triangle



Service-oriented architectures are nothing new; the *Common Object Request Broker Architecture* (CORBA) and the *Distributed Component Object Model* (DCOM) have long provided similar functionality. These existing approaches to service orientation, however, suffered from a few difficult problems. First, they were *tightly coupled*, which meant that both ends of each distributed computing link had to agree on the details of the API. A code change to a COM object, for example, required corresponding changes to the code that accessed that object. Secondly, such Service-oriented architectures were *proprietary*. Microsoft unabashedly controlled DCOM, and while CORBA was ostensibly a standards-based effort, in practice, implementing a CORBA architecture typically necessitated the decision to work with a single vendor's implementation of the

specification, because each vendor's interpretation of the standard varied enough to prevent seamless interoperability.

Finally, CORBA and DCOM are *fine-grained*, remote procedure call (RPC) architectures. RPC architectures approach the problem of distributed computing by simulating a single computer environment—allowing code on the local computer to execute individual calls to the remote computer as though that remote computer was really part of the local one. The platform then masked the low-level communication details that provided the remote access. This view is fundamentally not a view of distributed computing assets, but rather of extending a single system.

Early communication protocols, such as the Network File System for Unix and Microsoft's Distributed Computing Environment, enabled low-level communication between computers at the network layer. These protocols, in turn, led to the development of wire protocols for distributed computing, most notably the *Object Remote Procedure Call* (ORPC) protocol for Microsoft's DCOM and the OMG's *Internet Inter-ORB Protocol* (IIOP) that underlies CORBA.

RPC architectures like DCOM and CORBA enabled programs to be broken up into pieces, with different pieces running on different computers, allowing programmers to continue to work within an isolated computer mindset. Both DCOM and CORBA took basically the same approach: Write your programs so that the remote computer appears to be part of your own computer. If you need code that lives on someone else's computer, you can make a simple request, and the hidden machinery of the distributed platform will marshal the remote code and ship it to you via the preferred wire protocol.

Of course, the process of marshaling executable code and shipping it over the Internet opened up a Pandora's box of security concerns, mostly because this marshaled code would go over a variety of TCP ports that firewalls were supposed to block. In addition, both of these technologies had other serious limitations. DCOM is a Microsoft-only architecture, and CORBA, although intended to provide cross-platform interoperability, is in reality too complex and semantically ambiguous to provide real interoperability without considerable manual integration work. Furthermore, the fine-granularity of the RPC approach to Service orientation added to the problems, because all method calls that attempt to access the remote system must make individual round trips, inefficiently making use of network resources. Such method calls also have serious problems crossing firewalls, typically relegating CORBA and DCOM to only internal enterprise applications.

DCOM is a Microsoft-only architecture, and CORBA, although intended to provide cross-platform interoperability, in reality is too complex and semantically ambiguous to provide real interoperability without considerable manual integration work.

In spite of all of these issues, the concept of Service orientation continued to make sense, provided that the problems of proprietary approaches, tight coupling, and fine granularity could be solved. It is within this architectural context that Web Services were first imagined. Web Services offered the loose coupling, coarse granularity, and open standards support that presented challenges for adopters of CORBA and DCOM, leveraging the original Web Services standards—SOAP for communication, WSDL for self-description, and UDDI for publication and discovery. Web Services, therefore, were originally conceived in the context of Service-oriented architectures.

Despite this early focus, many companies looked to Web Services simply as a way to standardize otherwise proprietary interfaces. While there is clearly a significant value proposition to be had by implementing SOAP as a way to access and integrate with disparate systems, companies that implemented this simplistic point-to-point vision for Web Services soon realized that their cost savings were entirely short-term. That is, in order to realize long-term benefit from Web Services, they would have to not just put Web Services interfaces on

top of proprietary APIs and messaging schemes, but rather look to a new computing architecture as a way of changing the economics of integrating systems. Of course, service-orientation is that key.

Service-oriented architectures implemented with Web Services, however, are a significant improvement upon DCOM and CORBA's weaknesses. What's new about today's Service-oriented architectures built with Web Services are that they are standards-based and can provide loosely-coupled connections between Web Service producers and consumers. The reliance upon universally accepted standards like XML and SOAP provides broad interoperability among different vendors' solutions, and loose coupling separates the participants in distributed computing interactions so that modifying the interface of one participant in the exchange doesn't break the other. The combination of these two core principles means that companies can implement Web Services without having any knowledge of the consumers of those Services, and vice-versa. The Service-oriented architectures discussed in this article are the standards-based, loosely coupled kind, which we'll refer to as SOAs.

> The Service-oriented architectures discussed in this report are specifically those that loosely couple Service providers and consumers, and are based upon open standards—in particular, the standards that underlie Web Services. This report will refer to these architectures as SOAs.

The power and flexibility that SOAs can offer the enterprise are substantial. If an organization abstracts its IT infrastructure so that it presents its functionality in the form of coarse-grained Services that offer clear business value (as opposed to fine-grained Services that simply encapsulate the APIs of the underlying systems), then the consumers of those services (whether they are at the same company or one of that company's business partners) can access those Services independent of the underlying technology that supports them. Furthermore, if Service consumers can dynamically discover and bind to available services while they are active, then the IT infrastructure behind those Services can offer extraordinary flexibility to the businesses that invoke them.

The distinction between the practice of architecture and the architectures that result is subtle, but critical.

Achieving these levels of power and flexibility, however, is a difficult task that requires a new approach to architecture: the *practice* of SOA. This distinction—between the practice of architecture and the architectures that result—is subtle, but critical. This report discusses the practice of SOA, i.e., what are the best practices that Service-oriented architects and their organizations must follow to build SOAs.

2.1.1. Evolution of Distributed Computing

Service orientation is the fourth major distributed computing approach to affect IT since the mid-twentieth century.

Before we can fully understand the business motivations for SOAs, it is important to understand the transition to Service orientation from the macroeconomic, "whole market" perspective. ZapThink believes that Service orientation is the fourth major distributed computing approach to affect IT since the mid-twentieth century, as shown in Table II.1:

Table II.1: Distributed Computing Approaches

Approach	Timeframe	Programming Model	Business Motivations
Mainframe timesharing	1960s –1980s	Procedural (COBOL)	Automated business
Client/server	1980s-1990s	Database (SQL) and fat client (PowerBuilder, Visual Basic)	Computing power on the desktop
n-Tier/Web	1990s-2000s	Object-oriented (Java, COM)	Internet/eBusiness
Service orientation	2000s	Service-oriented (SOAP, WSDL, UDDI)	Business agility

There are several important points to make about the approaches outlined in Table II.1. First, each approach does not replace the one that came before, but merely augments it. We still have mainframes and client/server applications today. Likewise, Service-oriented applications aren't going to make Java obsolete. Instead, as technologies mature, it becomes economically practical to address a new level of business motivation. Client/server approaches only became a reality when businesses could afford PCs. The Internet's rapid rise depended on the service provider infrastructure and the availability of TCP/IP on the desktop. And now, enterprises can have the agility necessary to let business needs drive the technology. In addition, it is important to point out that SOAP, WSDL and UDDI do not constitute a programming model *per se*, but are rather a level of abstraction above the underlying programming models. This abstraction layer is one of the most important characteristics of SOAs. The most important conclusion to be drawn from the table, however, is that the status of Service orientation as a distributed computing approach is every bit as important as the others that came before—but is still an evolutionary step in an ongoing process. And while Table II.1 does not have a fifth row, it goes without saying that computing will continue to evolve beyond Service orientation.

2.2. Business Motivations for SOAs

Business agility *is the ability of a company to respond quickly and efficiently to change, and to leverage change for competitive advantage.*

The difference between the practice of SOA and other approaches to enterprise architecture is in the business agility that SOA offers. *Business agility* is the ability of a company to respond quickly and efficiently to change, and to leverage change for competitive advantage. For the architect, building an architecture that provides business agility means creating an IT infrastructure that meets as-yet unknown business requirements—a situation that throws traditional IT planning and design out the window.

To meet the needs of the agile enterprise, then, the practice of SOA has 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. The Service-oriented architect 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.

- **Business agility is the fundamental business requirement** – Instead of dealing with concrete requirements from business, SOA considers the next level of abstraction: the ability to respond to changing requirements is the new "meta-requirement." The entire architecture—from the hardware on up—must reflect the business agility requirement, because any bottleneck in an SOA implementation can substantially reduce the flexibility of the entire IT environment, and hence the business as well.
- **A successful SOA is always in flux**– To visualize how an SOA is supposed to work, it's better to think of a living organism rather than the traditional "building a house" metaphor that gave software architecture its name. The everyday normal state of affairs is an IT environment that is undergoing constant change, and as a result, the work of the Service-oriented architect is never done. House-building assumes a state of completion and the ability to craft a design that changes invariably over time, which is rarely the case in any business environment.

For the architect used to building houses, tending to a living organism requires a new way of thinking. Fortunately, SOA makes economic sense, and thus there is adequate motivation for architects to learn this new perspective.

2.2.1. The Economics of Business Agility

The current transition to Service orientation is fundamentally different from the last major distributed computing transition: the one from client/server to n-tier architectures in 1996-97. That last build-out heralded the beginning of the dot-com boom, where Internet-related investment coupled with Y2K expenditures created a kind of IT "perfect storm" so dramatic it led to a worldwide economic boom, and subsequent downturn. Today, of course, the economic environment for technology adoption has completely changed, and this return to the "new business normal" is accelerating the move to Service orientation. Rather than promoting massive build-out or extensive rip-and-replace, Service orientation embraces heterogeneity and obtaining greater value from existing legacy technology. Today's distributed computing transition, while every bit as significant as the ones that came before, has an entirely different economic model. Instead of massive IT investment, today's IT executive is concerned with thrift.

The Web Services story to this point in time is all about thrift. By 2002, most companies using Web Services were applying these new technologies to reduce integration costs. Hundreds of enterprises have already learned that taking a Service-oriented integration approach could reduce the cost of an integration project dramatically – with reported cost reductions as high as 80%.

Thrift, however, means more than simple cost savings. True thriftiness means making do with what you have—squeezing value out of every asset. One of the clear benefits of an SOA is that such architectures help companies get more value out of existing resources by wrapping legacy applications in Web Services interfaces and then making those Services available on the network. A second thrift benefit that SOAs provide is that they facilitate heterogeneous IT environments. Instead of "ripping and replacing" existing corporate IT systems by installing new systems and throwing the old ones out, SOAs enable users to build bridges between different systems and applications and leverage existing IT assets.

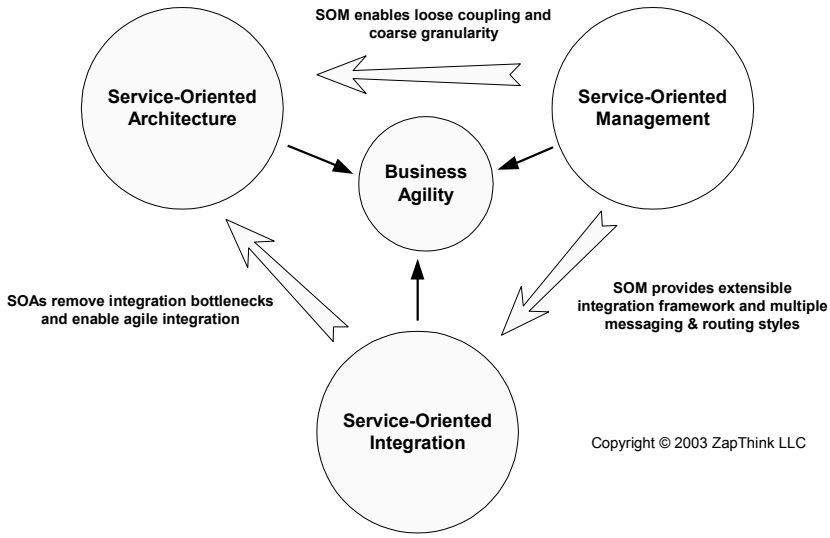
Reworking existing brittle, expensive IT infrastructures into flexible, Service-oriented environments promises substantial cost savings, most dramatically in terms of business agility.

Reworking existing brittle, expensive IT infrastructures into flexible, Service-oriented environments promises substantial cost savings, not just in terms of reduced integration expense and squeezing more value out of existing IT investments, but most dramatically in terms of business agility: the ability to respond quickly and efficiently to changes in the business environment, and to leverage those changes for competitive advantage. Change comes in many forms: changes in the marketplace, in technology, in the world at large. Companies that can make effective use of a changing environment are better able to compete and thrive in any business climate, but especially in tough economic times like those we have since the bursting of the dot-com bubble.

Information technology is often the area most relevant to discussions of business agility, because achieving agility begins with removing the bottlenecks that impede it, and IT has traditionally been the source of most bottlenecks. In fact, companies are so used to the fact that IT decision-making and implementations impede their organization that technology and its limitations often drive business decisions. Service orientation, however, has the potential to change this equation, and enable business decisions to finally drive their technology decisions. On the other hand, building Service-oriented infrastructures is not easy. It requires investment and commitment on the part of enterprises. The long-term business benefits of Service orientation, however, can justify such investments.

To understand Service orientation, it is important to grasp three related concepts: *Service-oriented integration*, Service-oriented architectures (SOA), and *Service-oriented management*, as illustrated in Figure II.2 below. Each of these three concepts plays a critical role in building agile technology infrastructures that provide business agility. For a complete discussion of SOI, see ZapThink's *Service-Oriented Integration* Report (ZTR-WS103), and for a discussion of SOM, see ZapThink's *Service-Oriented Management* Report (ZTR-WS106)

Figure II.2: Service Orientation Relationships



Using a SOA for integration is known simply as Service-Oriented Integration (SOI). Rather than explicitly declaring how systems will interact through low-level protocols and object-oriented architectures, as was the case with previous EAI and B2Bi efforts, SOI provides an abstracted interface with which systems can

interact. Systems merely need to expose their capabilities as Services, and other systems that choose to interact with them can simply discover those services and bind to them either at runtime or design-time. The enterprise also needs to have a management infrastructure in place that can support the performance of the Services as they are being moved into production as well as once they are live. ZapThink calls the management infrastructure needed to support the ongoing functionality of a SOA Service-oriented Management (SOM).

Companies must successfully rearchitect their IT infrastructures into SOAs in order to take advantage of SOI and remove the integration bottleneck, thereby connecting different systems in a flexible, cost-effective manner. Likewise, companies must implement SOM to enable loose coupling and coarse granularity, as well as enabling the integration framework required for SOI. The enterprise also must have a management infrastructure in place that can support the monitoring of Services performance as they are being moved into production as well as once they are available for public consumption.

Furthermore, in order to encapsulate the underlying software components and systems with Web Services interfaces and then compose these fine-grained atomic Web Services into coarse-grained business Services, companies must have a set of management tools that can establish and maintain the connections between the software on the one hand and the Services on the other. By removing the integration bottleneck with SOI and enabling SOAs with SOM, therefore, such companies are able to achieve their desired business agility, as illustrated in Figure II.2 above.

The rearchitecture needed to move to SOAs does not take place in a vacuum; companies must transition their systems from the existing architectures to SOAs in a manner that does not impede the ongoing necessary functionality of the technology. Furthermore, the act of rearchitecting is not sufficient enough by itself to guarantee that the resulting business Services will meet the needs of the business.

III. Foundations of SOA

There are several definitions for *architecture* in the context of software systems, but one that describes the concept quite succinctly comes from the Institute of Electrical and Electronics Engineers (IEEE). The relevant IEEE standard defines architecture as:

> The fundamental organization of a system embodied by its components, their relationships to each other and to the environment and the principles guiding its design and evolution. (IEEE P1471/D5.3)

While the components referred to in the above definition generally refer to hardware and software, the architecture also includes their relationships to their environment—namely, the people that work with the technology and the organizational constructs that form the business.

The term enterprise architecture *consists of an aggregated architecture of all the individual IT systems within an organization.*

The term *enterprise architecture* consists of an aggregated architecture of all the individual IT systems within an organization. In this case, the enterprise architecture environment include not only the human element within the enterprise, but also the systems, people, and organizational constructs at other companies that have relationships with the enterprise, as well as the individual consumers who are that enterprise's customers. Companies typically have relationships with customers, suppliers, and other organizations that have formal or informal interactions with the company. For the purposes of this report, we will describe all those companies and individuals as the enterprise's *partners*.

SOA is a particular approach to enterprise architecture. While SOA does require a shift in perspective regarding the role of IT in an organization, it is nevertheless important to remember that SOA is an evolutionary change to the practice of enterprise architecture; many of the established approaches and principles of the existing practice of enterprise architecture apply to SOA, including the use of modeling, the 4+1 View Model of Architecture, the Zachman Framework, and other practical and proposed approaches. Furthermore, it is also important to point out that in many ways the formal practice of software architecture is still immature in many ways. Engineers still typically build software systems by hand, and comprehensive designs for such systems are often nonexistent or cobbled together in many of today's enterprises. SOA, therefore, is not merely an evolution on a mature practice, but an important step in the maturation of the practice of software architecture.

SOA is an evolutionary change to the practice of enterprise architecture; many of the established approaches and principles of the existing practice of enterprise architecture apply to SOA.

The inherent immaturity in the practice of enterprise architecture places a special burden on SOA, because there are no universally accepted best practices for enterprise architecture to rely upon. Instead, the foundations of SOA this report will call upon are generally recent developments in the software architecture literature. In particular, there are two increasingly popular movements in IT—one architectural, the other methodological—that each have something to offer the Service-oriented architect. The first is *Model-Driven Architecture* (MDA), championed by the Object Management Group (OMG), the same body that looks after CORBA.

3.1. SOA Foundation: Model-Driven Architecture

The core concepts of MDA are *models* and *metamodels*. A model is a formal representation of a system, and a metamodel is a formal representation of a set of models—in essence, a model of models. For example, one of these MDA metamodels states that architects should begin with a formal model of the system being built, ideally in UML (the *Unified Modeling Language*, also shepherded by the OMG). However, that metamodel is only one example of many possible metamodels.

☆ Vendor Focus

Aonix
Borland
Interactive Objects
Popkin Software
Rational Software
Sun Microsystems
Sybase
Telelogic
Visible Systems

There are many software modeling products on the market that enable architects to apply the benefits of modeling, including tools from **Aonix**, **Borland**, **Interactive Objects**, **Popkin Software**, **Rational Software**, **Sun Microsystems**, **Sybase, Telelogic,** and **Visible Systems**. Typically, these tools implement some subset of UML to allow architects to manipulate visual representations of the organization and behavior of the systems they are designing. When architects speak of models, therefore, they often refer to both the formal construct as well as the representation of that construct in the modeling tool. This distinction is important when trying to understand MDA, because while much of the theoretical work behind MDA deals with the formal construct, the intention is to be able to implement MDA within modeling tools that provide concrete representations of the theory.

The MDA actually goes up one additional level of abstraction to the concept of a meta-metamodel, or a formal representation of metamodels that can be used to generate individual metamodels. This meta-metamodel is the *MetaObject Facility* (MOF). The MOF provides a standard way for building metamodels like the *Common Warehouse Metamodel* (CWM), a standard approach to data models, as well as models built with UML, typically of object-oriented systems. The MOF, therefore, can be used to build formal SOA metamodels—although nobody has done so yet. Nevertheless, current MDA metamodels provide a wealth of architectural approaches that form a solid foundation for SOA.

The MDA approach begins with a *platform independent model* that completely represents the functional requirements and use cases of the users of the system. From this platform independent model architects can derive whatever *platform dependent models* they need in order to specify the design of the system under construction. These platform dependent models can be extremely detailed and therefore be used to automatically generate the implementation code itself – the true benefit of MDA.

The core strength of MDA is that architects can fully specify systems in their designs, and thus there is little leeway for misinterpretation when the systems are built. This level of detail seeks to resolve the biggest problem with software models, which is that they are of limited use to developers in real-world situations. There are several reasons for this problem: first, the models are typically not detailed enough, leaving the developers to fill in the implementation details. Second, there is typically no way for developers to update the model when implementation decisions modify the architecture of a system, thus making the models out of date. Third, there is typically cultural resistance among developers to following a model. Developers aren't necessarily school-trained, and are often self-taught. Naturally, in the process, they skip modeling altogether and go straight for the programming jugular – missing the value of modeling entirely

The MDA solves many of these challenges in its modeling approach. Because models built with an MDA metamodel ideally fully specify the platform-specific functionality of a system, developers are no longer required to fill in the blanks in the model. And while MDA doesn't address the cultural issue of model development by programmers directly, it does change the role of the developer, by segmenting them into classes of architect or a lower-level programmer who meets predefined requirements.

However, MDA has some limitations as well. First, MDA assumes that the business requirements are fully specified before the model is built, which is not the case in the typical dynamic business environment. Second, MDA doesn't offer a feedback loop: if developers need to diverge from the models, there's no set way to keep the models up to date. So while it is possible to use the MOF to generate metamodels that provide the feedback and flexibility necessary to address these issues, the MDA as currently constituted falls short. For this reason, this report includes a second foundation for SOA.

3.2. SOA Foundation: Agile Methodologies

The second foundation of SOA is the *Agile Methodology* (AM) movement, most notably represented by *Extreme Programming* (XP). Agile methodologies like XP provide a flexible process for building software systems in environments where requirements are unknown or in flux. XP requires that a user representative work closely with the development team, helping to write tests that guide the daily work of the developers. All members of the team pitch in on the design, which is kept as minimal and informal as possible. All team members are responsible for all of the code, and anyone can rework any part of the project. Developers tackle AM projects iteratively, first writing a test, then implementing just the code necessary to pass all the tests.

The goal of agile methodologies is to build just what the user wants, avoiding extraneous work on artifacts like formal models. AM's focus is on people and working code, not on process or documentation. AM's core strength lies in its agility—the ability to deal with changing requirements. AM's main weakness is its limited scalability. For example, XP works well for small teams and projects of moderate size, but as the project scope grows, it becomes more difficult for the

team members to have a solid grasp of all aspects of the project without a concrete, comprehensive plan to work from. It is true, however, that companies have achieved some success with AM on large projects, typically by breaking those projects into smaller pieces, and also by introducing additional project management overhead that is not part of the agile approach.

One of the most important AM techniques is how the approach recommends adoption of the methodology itself. AMs are not intended to be adopted in an "all or nothing" manner, but rather the agile approach to adopting an AM is to adopt those aspects of the methodology that solve the pressing problems in the existing development process. Therefore, even though there are ideal descriptions of AMs, in reality, teams that implement some form of an AM typically pick and choose those aspects that are appropriate for their needs.

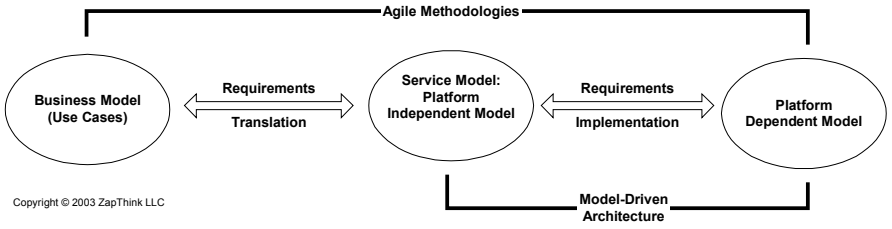
3.3. The SOA Metamodel

On the surface, MDA and AM appear to be contradictory—MDA assumes fixed requirements, while AM assumes the opposite; MDA centers on formal models, while AM eschews them. However, at the risk of being iconoclastic, this report takes certain elements from each of these approaches and puts them together into a coherent architectural practice. In particular, the key elements of an MDA that the SOA metamodel inherits are the platform independent model and the platform dependent models. In the SOA practice, MDA's platform independent model provides a guide for how we can model the business Services that encapsulate the underlying system functionality. The platform dependent models represent the particular implementations that underlie the business Services. The SOA metamodel also leverages the model-driven code capabilities of MDA.

In the SOA practice, MDA's platform independent model provides a guide for how we can model the business Services that encapsulate the underlying system functionality. The platform dependent models represent the particular implementations that underlie the business Services.

AM provides the missing piece that MDA lacks—feedback between the developers and the business users. While the MDA assumes that business requirements are complete in the platform independent model, the SOA metamodel introduces a separate business model that represents the changing business requirements in the enterprise—in other words, the use cases that describe the desired behavior of the architecture. In essence, AM provides for direct interaction between the business model and the platform dependent models, while the SOA metamodel introduces the platform independent Service model between those models. Figure III.1 below shows how MDA and AM conceptually fit into the overall SOA meta-model:

Figure III.1: The SOA Metamodel



The ovals in Figure III.1 represent the three levels of abstraction in an SOA, following the first principle of SOA from Section 2.2 above: the business drives the Services, and the Services drive the technology. AM relates the business model directly to the implementation, as represented in the platform dependent model. MDA, on the other hand, does not separate the business model from the platform independent model, but rather takes the platform independent model

(or Service model) as its starting point. SOA must connect these models, or levels of abstraction, into a single architectural approach.

The SOA metamodel, therefore, has the three models as shown in Figure III.1, connected by two-way arrows that represent the feedback loops. The left arrow represents the translation of business requirements into changes in the Service model that inform the IT organization about the requirements, and the right arrow represents the process of implementing those requirements in code. It is important to note that there is no arrow directly between the business model and the platform dependent models—this feedback always feeds through the Service model. An enterprise must actively maintain the Service model so that it can act as a central clearinghouse for business requirements from each constituency within the organization. The IT organization can then look to the Service model as the source of the business requirements it needs to make changes to the underlying systems, as represented in the platform dependent models. Note that the user is not directly a member of the development team as with AM; instead, the user is represented by the Service model.

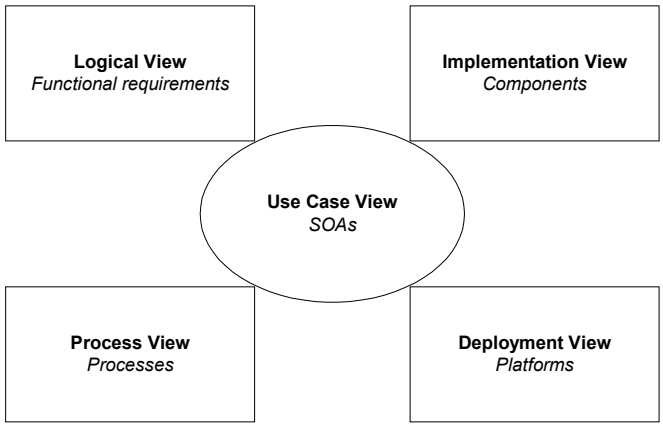
While the SOA metamodel puts together an abstract foundation for SOA, there must be clear connections between the abstract models and the real world of systems, software, processes, and requirements. Different people within the organization are responsible for each of these system elements, and thus the Service-oriented architect must be able to coordinate each participant's view of the enterprise IT environment. This report will provide this coordination with the 4+1 View Model of Architecture.

While the SOA metamodel puts together an abstract foundation for SOA, there must be clear connections between the abstract models and the real world of systems, software, processes, and requirements.

3.4. The 4+1 View Model of SOA

Enterprise architects find their profession both challenging and gratifying because they must consider IT from many perspectives. Philippe Kruchten of **Rational Software** distilled these perspectives into the 4+1 View Model of Architecture, which is applied to SOA in Figure III.2:

Figure III.2: The 4+1 View Model of SOA



The four rectangles represent different ways of looking at an architecture, representing different stakeholders. The fifth view, the use case view, overlaps the other views and plays a special role with regard to the architecture.

- The *deployment view* maps software to the underlying platforms and associated hardware, which is the way that systems specialists view the

architecture. These specialists typically work with the platform independent models that are appropriate to their specialty.

- The *implementation view* describes the organization of the software code, and is the view favored by developers. This view necessarily spans the Service model and the platform dependent models, because it is the role of the developers to provide the code necessary to encapsulate existing functionality into atomic Services and compose those Services into the business Services represented in the Service model.
- Business analysts work with the *process view*, which addresses the runtime issues of the software. It is the role of the business analyst to understand the business requirements, represent them in the business model, and maintain the connection between the business model and the Service model.
- The *logical view* represents the users' functional requirements. This view represents IT in a high-level way that deals with business concepts like customers and orders. Tools that enable the logical view provide a business environment for users to interact with the IT environment.
- In the case of SOA, the Service-oriented architect must be able to connect the users to the Services, and the Services to the underlying technology, following the threads of use cases in the *use case view*. This architect must therefore maintain the "big picture" of what the SOA is doing so that systemic problems can be recognized and dealt with. SOA involves several levels of abstraction, which are both powerful as well as risky. To mitigate this risk, the use case view helps the Service-oriented architect keep tabs on all aspects of the architecture.

Decision Point

In the case of SOA, the Service-oriented architect must be able to connect the users to the Services, and the Services to the underlying technology, following the threads of use cases in the use case view. This architect must therefore maintain the "big picture" of what the SOA is doing so that systemic problems can be recognized and dealt with.

To show how the Service-oriented architect must work with each of these views, let's put them in the context of the SOA metamodel, as shown in Figure III.3:

Figure III.3: The Practice of SOA

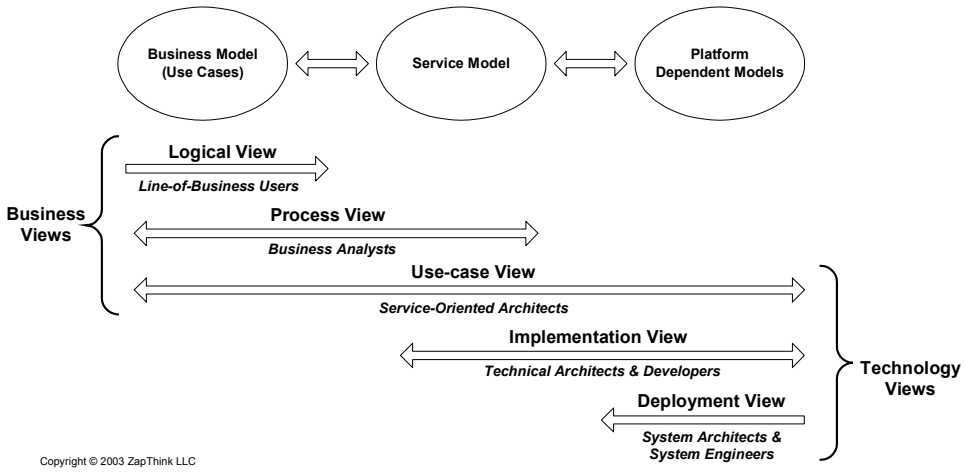


Figure III.3 shows the two overlapping realms of SOA: the business realm represented by the business model and Service model ovals on the left, and the technology realm represented by the Service model and platform dependent model ovals on the right. The Service model thus becomes the point of contact

between the business and technology views, and acts as the conduit for communication across the enterprise. Business users who use the logical and process views work with coarse-grained business Services, orchestrating them into processes as needed depending on the fluctuating requirements of the business. Technologists, on the other hand, work to build and maintain the abstraction layer between the services and the underlying technology. The central model, representing the Services themselves, acts as the axis around which the business moves.

The most important feature of the SOA metamodel is in fact the delineation between the business and technology realms, coupled with the explicit modeling of the two-way interaction between the two realms. This balance between the two realms enables the business to drive the technology in an environment of flux.

The most important feature of the SOA metamodel is in fact this delineation between the business and technology realms, coupled with the explicit modeling of the two-way interaction between the two realms. This balance between the two realms enables the business to drive the technology in an environment of flux. Traditional approaches to software architecture presuppose a traditional software development lifecycle, where users define their needs, and then IT builds and deploys the required system. In reality, this traditional, "waterfall" approach typically does not work, for a variety of reasons that boil down to risks that develop as a result of unknown or changing circumstances. As a result, companies react to the risks of the traditional approach by constraining the expectations of the business, essentially allowing technological risks and limitations to drive the business. SOA reverses this conundrum, providing sufficient flexibility to allow business to drive the technology.

The SOA metamodel achieves this flexibility by inheriting the platform independent and dependent models from MDA, while adding AM's interaction with the user as well as an agile feedback loop, represented by the two-way arrows between the ovals. Likewise, the meta-model solves AM's scalability issue by introducing the intermediate level of abstraction provided by the central Service model. Users can thus deal with the day-to-day concerns of the business, reflecting any changing requirements in the Service model. The technologists then can respond to these changing requirements quickly and effectively, because the underlying technology is model-driven.

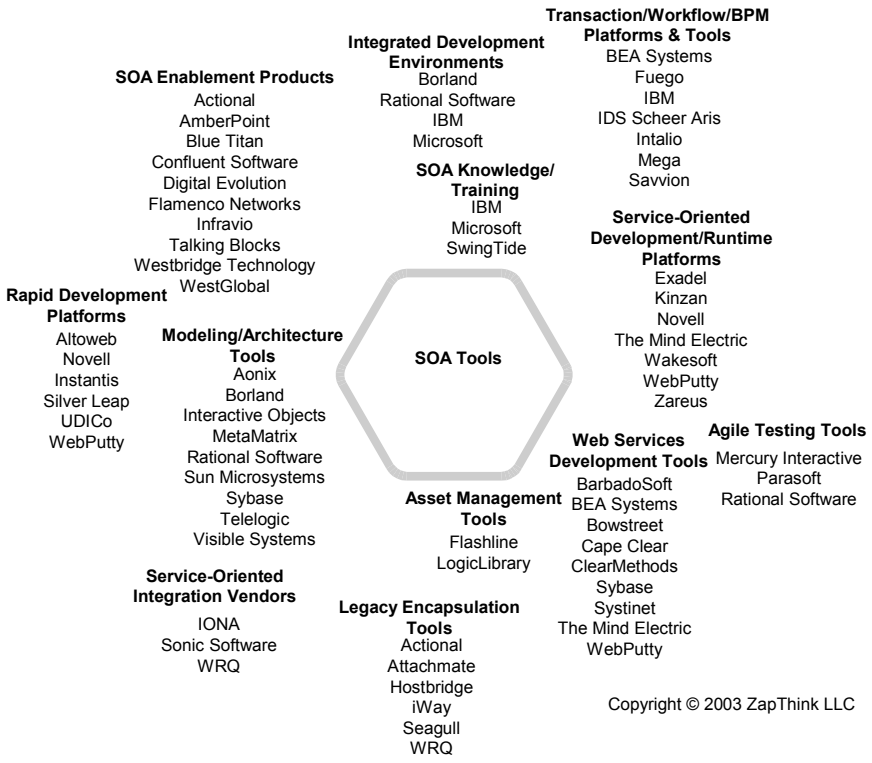
What's different about the practice of SOA and more traditional approaches to enterprise architecture is the way it enables agility. Remember that the third principle of SOA from Section 2.2 states that SOAs are always in flux. This environment of constant change is the cornerstone of the practice of SOA found in Figure III.3. The stakeholders shown in this figure continue to effect changes throughout the architecture on an as-needed basis. The line between design time and runtime blurs as the technologists respond to the changing business requirements as a normal part of day-to-day operations.

IV. Market Segmentation

There is at this time no single, nascent SOA tools market.

The most salient feature of the SOA tools market map shown in Figure IV.1 below is that there is at this time no single, nascent SOA tools market. Instead, the vendors profiled in this report fall more accurately into a variety of separate markets, some of which have a Service orientation focus, but many do not.

Figure IV.1: SOA Tools Market Map



Service orientation itself is not truly a market either, but more of an approach to distributed computing. As a result, there will likely be few vendors in the broader IT market that call their products "Service orientation" products, just as there are few vendors who refer to their products as SOA tools. Nevertheless, Service orientation and SOA tools fall broadly into the category of emerging markets, even though they are not distinct markets in their own right.

The existing market categories considered in this report that offer SOA tools include the following:

- *Service-oriented development and runtime platforms* – The products in this space are typically J2EE platforms that provide an abstraction layer between the underlying J2EE objects and the Services environment that developers work with. **Microsoft**'s .NET environment fits into this category as well.
- *Asset management tools* – Products that track a heterogeneous mix of software-related assets, including code, models, and associated documents for the purpose of facilitating reuse and greater visibility into the development process.
- *Modeling/architecture tools* – Tools that provide modeling environments for architects to design systems. These tools often provide some level of automated code generation, and can often reverse engineer code into models.
- *Web Services development tools* – Tools that specifically help developers build Web Services, but often do not offer additional Service-

orientation capabilities. Includes *Rapid development platforms*, which are tools that provide visual environments for developers that help them to build Service-oriented applications quickly, and *Integrated development environments (IDEs)*, which are traditional all-in-one visual environments for developers.

- *Service-oriented integration vendors* – Vendors that provide integration tools and platforms that abstract the systems being integrated with a Services layer. Please refer to ZapThink's *Service-Oriented Integration* Report (ZTR-WS103) for more information. While facilitating the development of SOAs, this market category is specifically focused on the challenge of heterogeneous, inter-system integration and communications.
- *Service-Oriented Management Products that enable SOAs* – Service-oriented management products that offer encapsulation and composition of Web Services as well as a range of runtime capabilities that enable companies to maintain business services specified in the Service model. See ZapThink's *Service-Oriented Management* Report (ZTR-WS106) for more information about these products.
- *SOA knowledge and training* – Leading vendors in the Service-orientation space who provide a range of resources for understanding and adopting Web Services and SOAs.
- *Legacy encapsulation tools* – Adapters that interface with legacy systems and expose their functionality as Web Services.
- *Agile testing tools* – Software testing tools that are appropriate for the agile test-first approach to development. See ZapThink's Web Services Testing Report (ZTR-WS105) for more information.
- *Transaction/workflow/BPM platforms and tools* – This category actually includes several related market segments. These tools are generally for the business user or the business analyst who is crafting, modifying, or executing business processes. ZapThink will fully explore this category in our upcoming *Service-Oriented Process* Report.

4.1. Current State of the Market

The interesting story about the current state of the SOA tools market is not about the tools, but about the overall market transition from n-Tier/Web architectures to SOAs. At this point in time, relatively few early adopter companies are building SOAs, and those are typically starting with one project. Nevertheless, there is substantial interest among IT executives in SOAs and the benefits they can bring to the business. These executives understand the value of business agility, and yet they are having trouble putting together all the pieces required to embark on an SOA adoption project.

IT executives are just now becoming aware of the potential of Web Services in the enterprise, and they are beginning to establish architecture teams and committees to look into how their enterprises can leverage Web Services strategically.

Looking backward to 2002, ZapThink's research shows that Web Services themselves have become widely accepted in the enterprise, albeit often at a grass-roots level. ZapThink's research shows that there is a disconnect between CIOs and some of the Web Services activity going on in their own IT organizations. Technologists always like learning and using the latest technology, and IT middle managers are realizing that Web Services can lead to dramatic cost savings on integration projects. IT executives are just now becoming aware of the potential of Web Services in the enterprise, and they are beginning to establish architecture teams and committees to look into how their enterprises

can leverage Web Services strategically. The answer, of course, is Service orientation.

V. Business and Technology Trends

As introduced in Section II, ZapThink believes that the transition to Service orientation represents the maturation of the distributed computing market, and hence of the overall IT market, as it moves toward open standards and away from "rip and replace" strategies. Overall, ZapThink sees a rapid transition to Service orientation, characterized by a tempered, measured increase in IT investment as compared to the period before the most recent economic downturn. This shallower annual growth rate is partly due to the overinvestment in technology during the 1990s and the resulting psychological reaction to IT expenditure, but we feel those effects of the dot-com era will diminish rapidly over the next few years. Instead, IT investment will experience a slower rate of increase simply because of the fundamental thrift properties of Service orientation. Or, more concisely put, the "new normal" for business will be thrifty IT expenditure rather than exuberant investment.

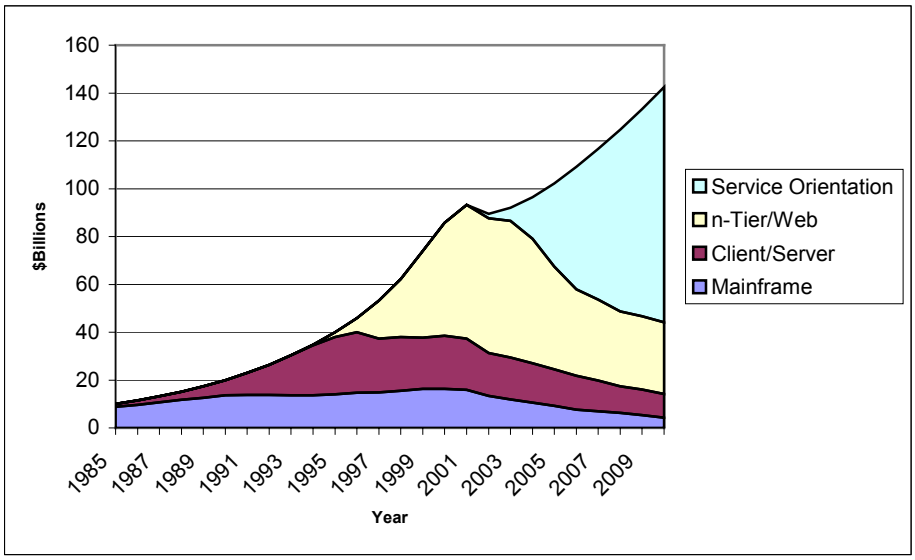
ZapThink sees a rapid transition to Service orientation, characterized by a tempered, measured increase in IT investment as compared to the period before the most recent economic downturn.

5.1. Long Term Trends: A Shift in the Favored Approach to Distributed Computing

As discussed in Section 2.1, ZapThink believes that Service orientation is the latest generation of distributed computing approaches, following the mainframe, client/server, and n-tier/Web approaches to distributed computing. In order to place this evolution in context, this report frames the discussion of distributed computing approaches in the context of the overall enterprise software market.

According to broadly available estimates of the size of the enterprise software market (including application development and deployment, applications, and systems infrastructure software), total revenues for the global enterprise software market grew from approximately $10 billion in 1985 to approximately $80 billion in 2000, with an average compound annual growth rate (CAGR) of about 15%. ZapThink estimates that the CAGR grew to a maximum of approximately 19% in 1999, to be followed by a severe downturn in IT spending, bottoming out in 2002 with a negative CAGR of approximately 4%. ZapThink predicts that the worst is over because of the gradual upturn in IT spending we see in the marketplace, and the IT market will gradually resume its growth pattern, reaching a CAGR of approximately 7% by 2006. The substantial decrease in the CAGR in the future when compared to the 1985-1998 period is due to the thrift effects of Service orientation. This trend can be seen in the topmost curve enclosing the graph in Figure V.1 below:

Figure V.1: Enterprise Software Market Composition



Source: Copyright © 2003 ZapThink, LLC

It is important to note that the top curve in Figure V.1 reflects a CAGR of approximately 14% from 1985 to 1999, but a CAGR of only 7% once the IT market pulls out of the downturn in 2003-2004. ZapThink attributes this substantial decline in the growth rate of the enterprise software market to the thrift advantages of Service orientation—which also explain why the three earlier approaches to distributed computing remain a significant presence throughout the decade.

ZapThink believes that the actual CAGR for the 2003-2010 timeframe for each approach is as likely to be above as below the 7% CAGR prediction. A stronger than expected economy would cause the CAGR to exceed 7%, while an unexpected lengthening of the economic downturn would cause it to be less than this amount.

Within the context of the overall enterprise software market, this report then considers the percentage of the market that generally falls into the four distributed computing approaches. This division is necessarily approximate, because it is often the case that a particular software package may have elements of more than one approach. For the purposes of this prediction, however, the four approaches are considered to be mutually exclusive. The percentage estimates can be found in Table V.1, below:

Table V.1: Proportion of Aggregate Expenditure on Products and Services Supporting Enterprise Computing Approach

	2002	2003	2004	2005	2006	2007	2008	2009	2010
Mainframe	15%	13%	11%	9%	7%	6%	5%	4%	3%
Client/Server	20%	19%	17%	15%	13%	11%	9%	8%	7%
n-Tier/Web	63%	62%	54%	42%	33%	29%	25%	23%	21%
Service Orientation	2%	6%	18%	34%	47%	54%	61%	65%	69%

It is important to note that the older approaches continue to have a significant presence even many years after their peak, and that aggregate expenditure on products and services focused on enabling Service orientation doesn't surpass similar spend on products and services for n-tier/Web until the 2005-2006 timeframe. By 2010, then, ZapThink expects 69% of the total enterprise software market to be Service-oriented. From these percentages, we calculated the relative proportions within the enterprise software market composition reflected in Figure V.1 above. The values for the period 2002-2010 appear in Table V.2 below:

By 2010, ZapThink expects 69% of the total enterprise software market to be Service-oriented.

Table V.2: Enterprise Software Market Composition

	2002	2003	2004	2005	2006	2007	2008	2009	2010
Mainframe	13.42	11.96	10.61	9.19	7.64	7.00	6.23	5.33	4.27
Client/Server	17.89	17.48	16.40	15.32	14.19	12.83	11.22	10.66	9.97
n-Tier/Web	56.35	57.05	52.11	42.91	36.03	33.84	31.17	30.65	29.91
Service Orientation	1.79	5.52	17.37	34.73	51.31	63.01	76.06	86.62	98.27
Total	89.45	92.01	96.50	102.16	109.18	116.68	124.70	133.26	142.42

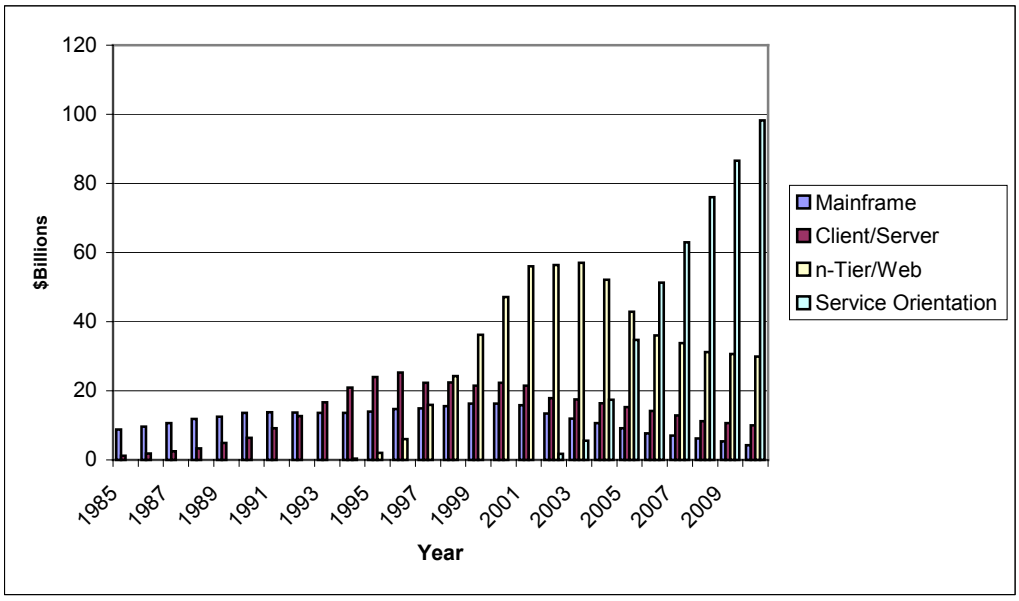
$Billions

ZapThink believes that the mainframe and client/server segments of the market are past their peak during the period reflected in Table V.2, and the declining percentages in this table reflect that fact. The n-Tier/Web segment continues to be strong through the decade, but its peak in 2003 reflects the fact that new investment in n-Tier/Web technologies will rapidly shift to products that offer Service orientation capabilities, putting them into the Service orientation segment.

By 2010, the overall Service Orientation market will produce annual revenues of $98 billion.

Therefore, by 2010, the overall Service Orientation market will produce annual revenues of $98 billion. To see how each distributed computing approach peaks and then gradually diminishes, we break down the total revenues for each approach into bars, as shown in Figure V.2, below:

Figure V.2: Enterprise Computing Approach Market Sizes



Source: Copyright © 2003 ZapThink, LLC

Note in Figure V.2 that while mainframes (which includes minicomputers for the purpose of this exercise) dominated the computing environment until the early 1980s, they remain a dominant force throughout the 1990s, and continue to be a significant force into the future. Client/server, which peaks in the mid-1990s in terms of dollar value, also remains a significant portion of the market. The n-Tier/Web curve peaks in the 2001-2003 period, and is characterized by a steeper ramp-up than the earlier two approaches, due to the dot-com buildout of the late 1990s.

ZapThink predicts that Service orientation will take hold in 2003, and become the dominant distributed computing approach by 2006.

Finally, ZapThink predicts that Service orientation will take hold in 2003, and become the dominant distributed computing approach by 2006, when it passes the n-Tier/Web approach. We expect the ramp-up to be as steep or steeper than the last one, not because of a major IT buildout, but rather because companies will be taking advantage of the thrifty encapsulation and embracing heterogeneity best practices in addition to the fact that new investments in n-Tier/Web solutions will increasingly be for Service oriented applications. Therefore, while existing systems will continue to remain in the IT environment, new investment in Service orientation will come to pervade the IT infrastructure.

5.2. Long-Term Trends: Grid/Utility Computing

The bars representing Service orientation in Figure V.2 continue to go up as they reach the right hand side of the graph. It may seem that we are predicting continuing growth for Service orientation beyond 2010, but that is not the case. In fact, each distributed computing approach ramps up quickly to a peak, and then gradually decreases as the next approach takes hold. We expect the same trend to occur with Service orientation—which begs the question, what comes next after this latest approach?

It is still too early to tell for sure, but there is one category of distributed computing on the horizon, known as *grid* or *utility* computing. There are already a handful of approaches to grid computing, but the main idea behind this new

★ Vendor Focus

IBM
The Mind Electric

approach is to virtualize computing resources both within an enterprise as well as across the Internet. All computers participate in "the grid," which acts like a single, massive computer that exposes all of its functionality via a Services interface. **IBM** is a leader in grid computing technology, and **The Mind Electric** also offers a grid computing project.

There is a good chance that grid computing in retrospect will be considered more a part of the move toward Service-oriented computing.

While it is convenient to consider grid computing to be the next approach after Service orientation, there is a good chance that grid computing in retrospect will be considered more a part of the move toward Service-oriented computing. Therefore, we may not have found the next big thing after all; we may only be glimpsing the full extent of the current trend. Please see ZapThink's upcoming *Grid Computing Technologies and Trends* Report for more information.

5.3. Inhibitors to Growth of Service Orientation and SOAs

This report provides a broad, high-level understanding of the direction enterprise computing is moving in as well as the tools and best practices that will get companies there. Nevertheless, many pieces are still missing from the SOA puzzle—pieces that software vendors and professional services organizations must provide. In the business realm, vendors must offer service-oriented business process, workflow, and service orchestration tools and services. Modeling tools must be available that can adequately reflect business services in an agile, platform independent way. The technologists must have tools that can generate code from models, and update those models when the code changes. And finally, vendors must offer SOA enablement software that allows Service-oriented architects to build and maintain the level of abstraction between the services and the underlying technology in a reliable and scalable way. Fortunately for today's enterprises, such products will be coming to market soon.

Another missing piece is a full representation of the SOA meta-model in terms of MDA. Today's incarnation of MDA presupposes fixed business requirements, but the business model in Figure III.1 represents a flexible, business-oriented interface to the SOA that enables business users to interact with the architecture. This limitation, however, is with the current implementations of MDA, not with the principles underlying it. Theoretically, the MOF provides the flexibility necessary for tools vendors to implement meta-models like the SOA meta-model in software.

Most of today's thinking about Web Services is bottom up: "here's how to build Web Services, now let's use them for integration."

The most important missing piece, however, is the change in mindset needed to take top-down approach to SOA outlined in this article. Most of today's thinking about Web Services is bottom up: "here's how to build Web Services, now let's use them for integration." That approach to using the technology is a great first step, because Web Services can dramatically lower the cost of integration, which is a story today's technology managers love to hear. As the economy improves and IT finally pulls out of the doldrums, however, companies will look to IT to offer increasingly strategic value to the organization. Service-oriented architectures provide the framework that will enable IT to offer this value in the form of business agility.

VI. Conclusions

Service orientation represents the next major trend in enterprise computing, and as such, requires a new perspective, new techniques, and new tools for implementing technology solutions that meet the needs of business. At this point in time, the IT industry stands at a cusp—a tipping point where sporadic applications of Web Services become a movement toward agile, thrifty

computing. When people stand at such a threshold, they often have a difficult time planning for the future, because many of the business patterns that have applied in the past may no longer apply. This difficulty is facing the SOA tools category now. This report refers to SOA tools as a category, because it is not yet a market in its own right. Instead, the vendors profiled in this report fall into several different markets. The common thread that ties them all together, however, is Service orientation—they all have some capability that helps companies build SOAs.

This movement to SOAs is very different from the last transition in enterprise computing approaches—the movement to the Web in 1995-97, primarily because of the completely different economic climate. Business agility and thrift are driving this new transition, which promises to provide value to business without the need for a massive buildout. As a result, there is an identifiable economic force that is acting as the impetus for the changes in IT ahead, and it is that economic force that gives the predictions in this report credibility. After all, nobody simply implements new architectures; companies make investments to achieve business goals. Only if a new architecture can clearly help companies achieve their goals will they invest in the change.

6.1. Key Notes

- Today's business person wants tactical solutions to current problems, in particular when there is money to be saved.
- As marketplaces evolve, there comes a time in their evolution where the benefits of cooperating in a standard manner outweigh the advantages of conducting business in a proprietary manner.
- The world of distributed computing is currently undergoing the transition from proprietary approaches to standards-based approaches.
- *Service-oriented architectures* are an approach to designing distributed computing infrastructures that thinks of software resources as services available on a network.
- DCOM is a Microsoft-only architecture, and CORBA, although intended to provide cross-platform interoperability, in reality is too complex and semantically ambiguous to provide real interoperability without considerable manual integration work.
- The distinction between the practice of architecture and the architectures that result is subtle, but critical.
- Service orientation is the fourth distributed computing approach to affect IT since the mid-twentieth century.
- *Business agility* is the ability of a company to respond quickly and efficiently to change, and to leverage change for competitive advantage.
- Reworking existing brittle, expensive IT infrastructures into flexible, Service-oriented architectures promises substantial cost savings, most dramatically in terms of business agility.
- The term *enterprise architecture* consists of an aggregated architecture of all the individual IT systems within an organization.
- SOA is an evolutionary change to the practice of enterprise architecture; many of the established approaches and principles of the existing practice of enterprise architecture apply to SOA.

- In the SOA practice, the platform independent model models the business Services that encapsulate the underlying system functionality. The platform dependent models represent the particular implementations that underlie the business Services.
- While the SOA metamodel puts together an abstract foundation for SOA, there must be clear connections between the abstract models and the real world of systems, software, processes, and requirements.
- The most important feature of the SOA metamodel is in fact the delineation between the business and technology realms, coupled with the explicit modeling of the two-way interaction between the two realms. This balance between the two realms enables the business to drive the technology in an environment of flux.
- In the Service-oriented environment, large projects with clear lifecycles give way to smaller, *ad hoc* projects that involve creating, updating and maintaining individual Services at different levels of granularity, as well as the consumers for those Services.
- With SOA, the Service model stands in for the user among the technical teams. The Services represented in this model form a critical layer of abstraction between the technology and the business users that forms the core of Service orientation, and thus the users and the technologists do not need to work together directly as long as the Service model is fully operational.
- There is at this time no single, nascent SOA tools market.
- IT executives are just now becoming aware of the potential of Web Services in the enterprise, and they are beginning to establish architecture teams and committees to look into how their enterprises can leverage Web Services strategically.
- ZapThink sees a rapid transition to Service orientation, characterized by a tempered, measured increase in IT investment as compared to the period before the most recent economic downturn.
- By 2010, ZapThink expects 69% of the total enterprise software market to be Service-oriented.
- By 2010, the overall Service orientation market will produce annual revenues of $98 billion.
- ZapThink predicts that Service orientation will take hold in 2003, and become the dominant distributed computing approach by 2006.
- There is a good chance that grid computing in retrospect will be considered more a part of the move toward Service-oriented computing.
- Most of today's thinking about Web Services is bottom up: "here's how to build Web Services, now let's use them for integration."

6.2. Decision Points

- Open standards-based Service-oriented architectures built with Web Services form a matched set of technical principles that can provide companies with the strategic IT direction they need to meet their fundamental business goals.

- Any company that has systems that talk to other systems over a network has a vested interest in conducting business via standards-based distributed computing—in other words, a vested interest in Service orientation.
- In the case of SOA, the Service-oriented architect must be able to connect the users to the Services, and the Services to the underlying technology, following the threads of use cases in the use case view. This architect must therefore maintain the "big picture" of what the SOA is doing so that systemic problems can be recognized and dealt with.
- Enterprises looking to build SOAs are not yet able to take the "buy the tools, then build the architecture" approach. Instead, as early adopters, the approach enterprises should take at this time is to set an architectural goal, and follow the emerging best practices that meet those goals.
- Interoperability drives new IT investment.
- Companies should only consider a rip and replace strategy as a last resort, and then only within a Service orientation context.
- Companies must first take an agile approach to thinking about the problem of SOA—that is, to accept the fact that the business environment is in a state of constant flux, and plan accordingly.

6.3. Best Practices

- Develop a top-down, extended enterprise SOA.
- Build & maintain a platform independent Service model.
- Maintain feedback at all points of the architecture.
- Follow Agile Methodology principles & techniques within the context of the Service model.
- Encapsulate existing/legacy functionality.
- Embrace heterogeneity/follow a federation model of software.
- Compose atomic Services into coarse-grained business Services.
- Build for consumability/broad applicability.
- Perform ad hoc upgrades.
- Prioritize SOA transition activities on the fly.

6.4. Figures

6.5. Tables

- Table II.1: Distributed Computing Approaches
- Table V.1: Proportion of Aggregate Expenditure on Products and Services Supporting Enterprise Computing Approach
- Table V.2: Enterprise Software Market Composition

VII. Profiled Vendors

The twenty vendors profiled in this section fall into several different market segments, as shown in Figure IV.1. The common thread that binds them together, however, is that they each help companies build SOAs. This section briefly describes the relevant products for each vendor, and then places those products in the context of building an SOA.

7.1. AltoWeb

AltoWeb is no longer in business

7.2. Borland

Please see ZapNote ZTZN-1014

7.3. Bowstreet

Please see ZapNote ZTZN-1015

7.4. Exadel

Please see ZapNote ZTZN-1039

7.5. Flashline

Please see ZapNote ZTZN-1043

7.6. IBM

Please see ZapNote ZTZN-1050

7.7. Instantis

Please see ZapNote ZTZN-1051

7.8. Kinzan

Please see ZapNote ZTZN-1059

7.9. LogicLibrary

Please see ZapNote ZTZN-0181

7.10. MetaMatrix

Please see ZapNote ZTZN-1064

7.11. Microsoft

Please see ZapNote ZTZN-1066

7.12. Novell

Please see ZapNote ZTZN-1071

7.13. Rational Software

Please see ZapNote ZTZN-1081

7.14. Sun Microsystems

Please see ZapNote ZTZN-1093

7.15. Sybase

Please see ZapNote ZTZN-1095

7.16. Systinet

Please see ZapNote ZTZN-1096

7.17. The Mind Electric

Please see ZapNote ZTZN-1098

7.18. Wakesoft

Please see ZapNote ZTZN-1105

7.19. WebPutty

WebPutty is no longer in business

7.20. Zareus

Please see ZapNote ZTZN-1111

Related Research

Reports

- *Web Services Technologies and Trends* Report (ZTR-WS100)
- *Service-Oriented Integration* Report (ZTR-WS103)
- *XML and Web Services Security* Report (ZTR-WS104)
- *Testing Web Services* Report (ZTR-WS105)
- *Service-Oriented Management* Report (ZTR-WS106)
- *Service-Oriented Process* Report (ZTR-WS108)
- *SOA Consulting* Report (ZTR-WS109)
- *Grid Computing Technologies and Trends* Report (forthcoming)

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