

Overcoming the Roadblocks to SOA Success

Realizing the Value of Service-Oriented Architectures

Executive Summary

You're probably running fast to accommodate new customers, new products, new business partners, new technologies, new regulations, and new executive leaders with new strategies and new business models. In an effort to overcome the limitations of existing enterprise systems to be adapted to new business needs, you have likely turned to service-oriented architectures (SOAs). SOAs support service-oriented integration and the creation of composite applications from loosely coupled and platform-independent services—typically, XML-based Web services. The services, which incorporate business logic and data handling, represent components of business application functionality.

Composite applications and service-oriented integration enable the development of new solutions from existing applications. In short, they allow companies to do more with what they already have. Additionally, the supporting service-oriented architecture promises to deliver to enterprises the capacity for unprecedented agility. It's a compelling value proposition, but fulfillment of such potential, however, is non-trivial and requires much more than the relatively easy first step of applying Web services veneers on existing applications. Challenges in the way of enterprise SOA success include:

- **Data rationalization.** Semantic incompatibilities, redundancies, and discrepancies in definition may exist in the data that services draw from multiple enterprise systems.
- **Business service enablement.** Many existing services—crudely auto-generated by commodity tools, provided by application vendors as extensions to their platforms or coded by programmers for one-time programmatic use—are mere thin veneers on a non-service-based API, making them unsuitable for the service requirements within an SOA.
- **Abstraction incompatibility.** Mismatches may exist among Web services that have been designed at different levels of abstraction making them difficult to interoperate.
- **Service accessibility.** The need to maximize service reuse necessitates a meta data repository to catalog and enhance the understanding of available services.
- **Constrained innovation.** Personnel inside and outside the core IT function that are closer to the business problems to be solved need user-friendly tools that empower them to create innovative integrated solutions, further increasing service reuse and the organizational return on an SOA investment.

Indeed, without a suite of tools specifically designed to address the needs of an emergent SOA, enterprises will remain distanced

indefinitely, separated from the SOA vision by an unyielding gap between existing applications infrastructure and the innovative service-oriented solutions that deliver the value of an SOA investment.

Service-Oriented Architecture

Accumulated over the last 40 years, the enterprise's packaged, custom and legacy applications remain critical to the business. Indeed, they represent its best practices and competitive advantage. For these reasons, abandonment of these applications is inconceivable. Yet these applications are typically inflexible to changing business requirements. This was never the vision. Rather, the vision was, and remains, the freedom to exploit all the enterprise's IT assets, integrating them easily and cost-effectively as business needs evolve.

The vision now has a name: service-oriented architecture. Less a set of technologies and more an architectural principle, SOA proposes the interconnection of reusable building blocks (software services) via standard interfaces. Analogous to Dell Computer's build-to-order model for the rapid assembly of personal computers from standardized components, SOA enables the rapid assembly of integrated business solutions from standards-based services. These services exploit, non-invasively, the best practices embodied in the heterogeneous applications across the enterprise and, sometimes, beyond the firewall.

SOAs achieve their special brand of integration by exploiting a set of connectivity standards that enable IT assets to access and provide services. Moreover, the connections between the services are loose, a quality effected by abstraction—concealment of as much as possible of the service's underlying implementation below the service's interface. In this way, changes in a service's implementation need impact neither the service's interface nor any connected services.

The Business Value of SOA

Beleaguered CIOs, enterprise architects, and other IT staff responsible for delivering solutions to the business are wasting no time in adopting SOA as the strategic vision for their organization. Indeed, SOA's potential business value is manifold:

Agility. In conventional monolithic architectures, IT systems exchange data and requests via multiple technologies and a tangle of connections. The resulting dependencies between systems are so numerous that system reconfiguration—for example, to support a new business solution—is often extremely time consuming. Indeed, tight coupling between applications may render conventional architectures so fragile that changes become prohibitively complex. The service-oriented architecture, in contrast, is much more agile. Overcoming the paralysis of conventional architectures, it simplifies system reconfiguration

through loose coupling between well-defined services. Indeed, SOAs promise to make today's IT departments as agile as the enterprises they must serve.

Ease of Integration. Integration in a well-designed SOA is as easy as possible. Loose coupling combined with standard interface technologies make integration a virtual snap. Furthermore, the coarse granularity of services frees developers from much of the complexity when connecting disparate IT assets in heterogeneous environments. And the reusability of services reduces duplicative coding.

Lower Development Costs. SOA's ease of integration—specifically, its reuse of proven business functionality and alleviation of much complexity through coarse granularity—considerably lowers development costs. Indeed, it reduces the development of new business solutions to an assembly exercise that doesn't require the deep technical skills when coding solutions from scratch. In these ways, IT functions that deploy SOAs can at last eliminate the backlog of requests for new enterprise solutions.

Alignment. Constrained by conventional architectures, IT functions have traditionally pursued a technology-driven approach to the automation of the enterprise's business processes. Under a service-oriented architecture, the emphasis changes from technology-driven to business solution-driven. Also, SOAs motivate the creation of services that are more meaningful and hence more accessible to business users. With these shifts, enterprises can expect much closer alignment between the IT function and their business units. And IT will look less like a support function and more like a provider of business-solution design expertise.

Innovation. As the costs of developing and integrating applications fall, the IT function is able to progressively deliver more value to the organization. In this way, SOAs promise to transform the IT function into a center of innovation that continually devises new ways of combining software services for competitive advantage. More important, however, is the opportunity created by SOAs to unleash employees closer and closer to the business problem to innovate new solutions to meet their needs directly, and to do so with a set of IT-provided, proven, understandable business components. Examples of such innovation include enabling customer-contact centers to become sales channels, creating new self-help and exception-management applications, providing a broader and deeper view of the customer, and increasing the accountability of managers.

SOA's Compelling Vision Provokes Rapid Adoption

Having spent many years locked inside an information technology Tower of Babel, CIOs are exhibiting fervid enthusiasm for the promise held by service-oriented architectures. Findings from a recent Yankee Group survey of 437 enterprises indicate that 48% have already deployed Web services and 39% expect to deploy them within a year. Whereas a mere 6% expressed no deployment plans, fully 75% of respondents said they would invest in technology and staffing to enable a service-oriented architecture in the next twelve months. Based on such responses, Gartner estimates that more than 60% of enterprises will have adopted service orientation as the guiding principle in the design of mission-critical applications by 2008.

Findings from a recent Yankee Group survey of 437 enterprises indicate that 48% have already deployed Web services and 39% expect to deploy them within a year.

Companies are now reporting success from their SOA initiatives. According to the Yankee Group survey, two-thirds of early adopters say that a service-oriented architecture has reduced complexity in distributed applications, while more than three-quarters believe it has enhanced their ability to collaborate with business partners. So expect SOA adoption to increase, especially in government, where service enablement is the current focus, and in consolidating industries such as telecommunications, financial services, and healthcare, where rampant M&A activity is posing particular integration challenges.

Beware: Roadblocks Ahead!

The SOA vision will undoubtedly induce widespread adoption by IT functions in every industry. However, a sizable gap exists between the vision and the practical reality of building composite applications using today's product offerings. In particular, having seen the proof of service-enablement concept via Web service wrappers around the APIs of existing applications, IT architects must next identify and create the kind of services that will allow them to exploit the full capabilities of a service-oriented architecture. These capabilities far exceed the mere platform independence of a WSDL-defined, SOAP-wrapped API.

Good services must convey business value and attractiveness to the users—the creators of composite applications. And that business value must be easily understandable and accessible. Business meaningful services typically favor coarse-grained services because they are likely to encapsulate more business value. If the service is too fine-grained, it might not appear

meaningful to the specific need of a user who is now tasked with combining it with other services to meet their needs. A service that is too coarse-grained is likely to be overloaded with functionality and data, making them perhaps wasteful of resources, and more difficult to use in certain instances and with too much irrelevant capability for the user. Meaningfulness (and therefore appropriate granularity) is often in the eyes of the user, even within the same functionality domain. Such differing user needs often motivate a selection of services to accommodate various demand scenarios.

Abstraction and interoperability are other qualities of good services. These qualities derive from concealing below the interface of each service as much as possible of the service's implementation. When the implementation is properly concealed, a user of a service that retrieves, say, a customer profile need not care that the source data is actually divided among several application silos. More generally, abstraction—a means of loose coupling—ensures that changes in a service's implementation have no impact on the service's interface and any connected services, thereby engendering IT functions with the agility to rapidly assemble new enterprise solutions from existing software assets.

Other considerations include making available services form a coherent whole in both functional and operational terms, and significant emphasis should be placed on reusability. Good services are a facet of design, not technology. For the present, Web protocols are likely the best technology choice, but that choice might change in particular circumstances or as new technologies emerge. The qualities that constitute good services, in contrast, will likely persist indefinitely.

Business Services: Right-Sizing and Refinement

Services may divide the functionality of applications into smaller components, but these components won't necessarily provide the precise service that a business solution requires. For example, a new sales force automation solution may need a service that validates a customer's eligibility for a particular promotion. Composing such a service might entail aggregating three finer-grained services that access a customer relationship management system (to check that the customer resides in the United States), an accounting system (to confirm that the customer has no outstanding balance), and, beyond the firewall, a business partner's application (to retrieve the customer's credit score).

Interconnection of enterprise services poses yet another challenge for architects of composite applications. Despite the existence of standards-based interfaces, service interconnection may be impracticable because different people have developed the services in different contexts and at different levels of granularity. Without custom tools, developers may have to manually recode services or provide "glue code" so that they can interconnect with one another. Such painstaking labor deprives SOAs of much of their virtue—namely, rapid integration and composite application

assembly. The ability to refine existing services to the appropriate level of granularity to meet interoperability requirements is an important capability within an SOA.

In summary, developers need to be able to right-size and otherwise refine mismatched services. Usually, this entails aggregating raw or finer-grained services into so-called enterprise services—building blocks that interconnect easily to form composite applications. For example, a developer might need to combine three raw Web services—one to parse data from an online order form, another to make calls to a database, and a third to submit messages to the company’s shipping agent—into a single event-based enterprise service that automates multiple business tasks. Business services represent a consistent and meaningful (that is, suitably high) level of abstraction. Once created, they need to be readily accessible for reuse by applications and channels on either side of the firewall. They also need to be semantically interoperable—another obstruction between the creation of services and the rapid assembly of enterprise solutions.

Semantic Interoperability

Through the addition of a thin and transparent veneer to existing software, Web service-enabled components can communicate with each other via a platform-independent messaging protocol. However, the underlying data (for example, customer records) to which the service refers usually remains in a structure and nomenclature unique to the application (for example, a customer relationship management application). This is the problem of semantic interoperability. Just as English and French speakers can use the same alphabet without speaking the same language, a CRM system using a dialect of XML will not necessarily understand the dialect of an order management system. Semantic interoperability enables two Web services to interact with each other despite their semantic differences. Only with semantic interoperability could a travel operator, say, build a composite application that exchanges data with airlines, hotels, rental car companies, weather sites, and other providers.

But the challenge of semantic interoperability is not confined to applications that extend beyond the firewall. It is a large problem inside enterprises. Consider, for example, the difficulty of assembling a composite application that accesses information about customers when the definition of a customer—that is, the associated data and business logic—is distributed across five different enterprise systems. Complicating the task, redundancies and inconsistencies may exist in the data. Separate applications

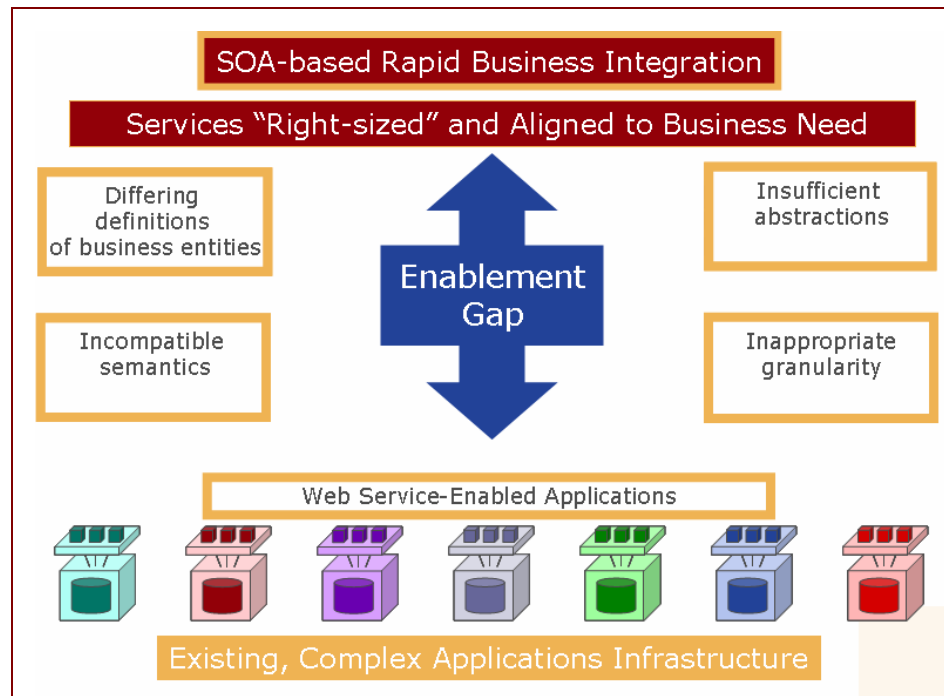
Just as English and French speakers can use the same alphabet without speaking the same language, a CRM system using a dialect of XML will not necessarily understand the dialect of an order management system.

have different ways of defining the same business entities, often with different yet overlapping representations. At the same time, distinct applications may provide unique elements that do not exist in separate applications. And still further impeding semantic interoperability, inconsistencies might exist in the underlying business logic. For example, two enterprise systems might employ different logic to define a customer as preferred status. Rationalizing this chaotic stew so that appropriate services cannot just be defined, but truly interoperate is a prerequisite for an effective SOA.

Enterprise application integration systems and existing Web services applications typically address semantic interoperability in one of two ways: through standard vocabularies (for example, the ebXML standard) or through custom translation code. Neither is adequate. Standards are not always adaptable to broad usage or to new business requirements, widespread adoption of standards is extremely difficult to achieve, and few standards exist for data that resides behind the firewall (for example, in the definition of a customer). Standards do often provide an effective means for industry-specific, business-to-business integration. However, those standards-based vocabularies must then be mapped to the pre-existing internal systems to complete business processes.

Translation code, the alternative to standards, defeats the purpose of using Web services. Indeed, translation deprives services of their loose coupling and therefore their reusability. The coding is also very time consuming and often leads to lengthy backlogs in the delivery of business solutions by the IT function.

Creators of composite applications need a better means to achieve semantic interoperability. They need tools specifically designed to solve these complex challenges and then, solved once, the abstractions and relationships need to be readily reusable by employees without requiring the knowledge of the details behind the simplified representations. Put another way, Siebel, SAP and your legacy system all define a customer differently and each manages certain aspects of the customer such as opportunities, orders and billing history. A common definition of a customer requires rationalizing each system's definition. And creating new applications that combine opportunities, orders and billing history requires navigable relationships between the services fronting each system. These abstractions and relationships—defined through metadata, not code—are themselves reusable building blocks that simplify the assembly of new integrated solutions.



Challenges to realizing benefits of a SOA.

Service Accessibility

As meaningful abstractions of the enterprise's IT functionality, software services are valuable assets in their own right. But qualities aside, services are only of value if they are understood and accessible for use and reuse. Enterprise personnel, including personnel outside the IT function, must know about services, understand their capabilities, and be able to find and use them whenever they need them. Such a requirement entails the provision of a repository and metadata that describes service capabilities in familiar business-object terminology; organizes, perhaps through visual representations, related services in a meaningful context; manages secure access to services; and offers advanced searching features so users can quickly assess the capabilities of available services.

Services also require management accordingly. This management challenge arises and escalates as the number of services and service consumers—including business partners—increases. Indeed, that number can quickly escalate as IT departments break down monolithic legacy applications into smaller service components. For example, an order entry application automates the process of creating orders. The Web services that the IT department creates from that application, however, might include numerous individual functions provided by the original application such as `GetOrderByCustomerId`, `CreateOrder`, `BackOrderItem` and so on. The result: a proliferation of services to manage. Enterprises may also use services that originate outside the firewall, further adding to the service management challenge.

In these ways, a repository promotes maximum reuse of services while opening application assembly to less technical personnel. Enterprises that wish to rapidly assemble composite applications need to deploy instead some sort of repository for the accessibility and management of services.

Solution Assembly & Deployment

SOA is an architectural principle, a design methodology. Without tools to support the rapid assembly of services into new business solutions and deploy those solutions to all applicable channels, the realization of the benefits of service-oriented architectures will remain elusive. These tools must be as easy to use as possible, demanding both minimal technical skills isolating users from programming languages, as well as minimal required knowledge of the details behind the services (whether they are domain or technical details). Such a skills requirement ensures that the broadest possible audience, including business analysts and other less technical personnel, can contribute their innovative potential to the creation of enterprise solutions.

Indeed, tools for the rapid assembly and deployment of composite applications need not be nearly as complicated as integrated development environments (IDEs) such as Microsoft's Visual Studio .NET and Borland's JBuilder, which programmers use to write code and expose application functionality as Web services. Moreover, enterprises that use such IDEs to assemble and deploy services are missing the opportunity of using simple yet powerful tools that have been purpose-built to leverage the services that were created in an IDE.

These tools are not only simple and powerful, but versatile as well. Aspirants to service-oriented architectures will also need tools that are versatile. By 2009, Gartner estimates, almost half of all new interactive applications will be designed for multichannel deployment. Such applications will draw from the same set of reusable business logic, with the particular choice of services for each channel determined by that channel's interface requirements. Consider a customer-service application that's deployed across five channels: call center operators, customer-service managers, account managers, a customer self-service application, and the customer-service application of a reseller. Each of these five channels has different functional requirements. For example, call-center operators need a speedy interface, customer-service managers need access to reports that show operator productivity, and a reseller needs to periodically update its price schedule from the company's product database. In addition, users will access these applications via different devices—thin-clients (call-center operators), notebook computers

By 2009, Gartner estimates, almost half of all new interactive applications will be designed for multichannel deployment.

(customer-service managers), hand-held devices (account managers), and so on—further increasing the complexity of the channel-specific front-end logic. In this way, versatile assembly and deployment tools are critical to the efficient development of multichannel applications.

Realizing the SOA Vision with Above All Software's Composite Application Platform™

Above All Software powers enterprises across the enablement gap so they can realize the vision of a service-oriented architecture. Its Composite Application Platform provides the tools necessary to move beyond low-level service enablement of applications infrastructure to the end game of rapidly assembled composite applications.

A Repository for Your SOA

Above All provides powerful tools and a repository (the Above All Dictionary) that catalog services for easy management, control, and reuse. The tools translate and abstract the often-cryptic information in a service's WSDL into more familiar business-object metadata that describes the service's capabilities and how to use them. Once the tool has extracted the semantics of a particular Web service, it catalogs the service in the Above All Dictionary to make it available for easy reuse. Additional metadata can be captured to further aid less technical users in understanding the service's capabilities and use in new integrated solutions. In addition, Above All allows organizations to establish multiple repositories while controlling access to services through role-based security.

Modeling Services into More Convenient Representations

Above All features a modeling tool to refine Web service capabilities to match business requirements. For example, suppose a company embarks on a new channel-partner strategy. The strategy necessitates creating a more coarse-grained and business-meaningful enterprise service that channel partners can use to manage orders. This service must coordinate several transactions involving the company's order-management application as well as a CRM application (for customer validation) and an accounting application (for account status verification).

Above All's modeling software supports the creation of just such a service. Moreover, the resulting service hides from the service user the details (that is, the finer-grained services, the message formats, the implementation architecture, and so on) behind the transactions. Also of note, Above All's modeling capabilities can be used to create new services that automatically synchronize or rationalize redundant systems, combine multiple services into a more appropriate composite service, or simply refine an overly complex service into a more useful representation.

Semantic Interoperability Among Disparate Services

Disparate services may operate on data that is not directly compatible. For example, services related to customer in a CRM system from Siebel may be semantically incompatible with services related to the customer's order in an ERP system from SAP. This incompatibility can make extraordinarily difficult the assembly of composite applications that source data and business functionality from multiple enterprise systems. Powerful visual tools within the Above All's Composite Application Platform overcome semantic interoperability by allowing users to define relationships between services. For example, the relationship "customers have orders" may be created by joining a customer service to an order service via a common data attribute, a transformation, or a look-up service that maps different customer identification schemes across systems.

Several relationships may in fact be definable between customers and orders—for example, services for back orders, open orders, and so on. Once defined, these many relationships constitute a visual map that becomes part of the reusable definition of a relevant service within the repository. Predefined relationships make the development of composite applications easy. They eliminate the need to figure out relationships and write tedious code to make disparate services semantically interoperable. These abstractions and relationships are defined as metadata defined and managed in the repository and as easily reusable as the services themselves.

Assembly Without Programming

Using Above All's visual tools and wizards, application assemblers can easily and rapidly snap together composite applications. The Above All Dictionary hides the technical intricacies of the underlying implementations and organizes services into familiar business-object terms like customers and orders, so assemblers can focus on the enterprise's business needs. Furthermore, with the capacity to refine services and predefine relationships between services, users can easily overcome nettlesome problems like semantic interoperability and interconnection mismatches between services developed in different contexts. Indeed, Above All's tools empower a broad range of enterprise personnel—from application developers to business analysts to power users—to rapidly assemble composite applications that meet changing business needs on demand.

In light of such of ease of use, power, and versatility, it is no wonder a recent issue of *Web Services Journal*[†] lauds Above All Studio—a component in the Composite Application Platform tools suite—as a "slick point-and-shoot visual environment for building composite applications." And these composite applications can

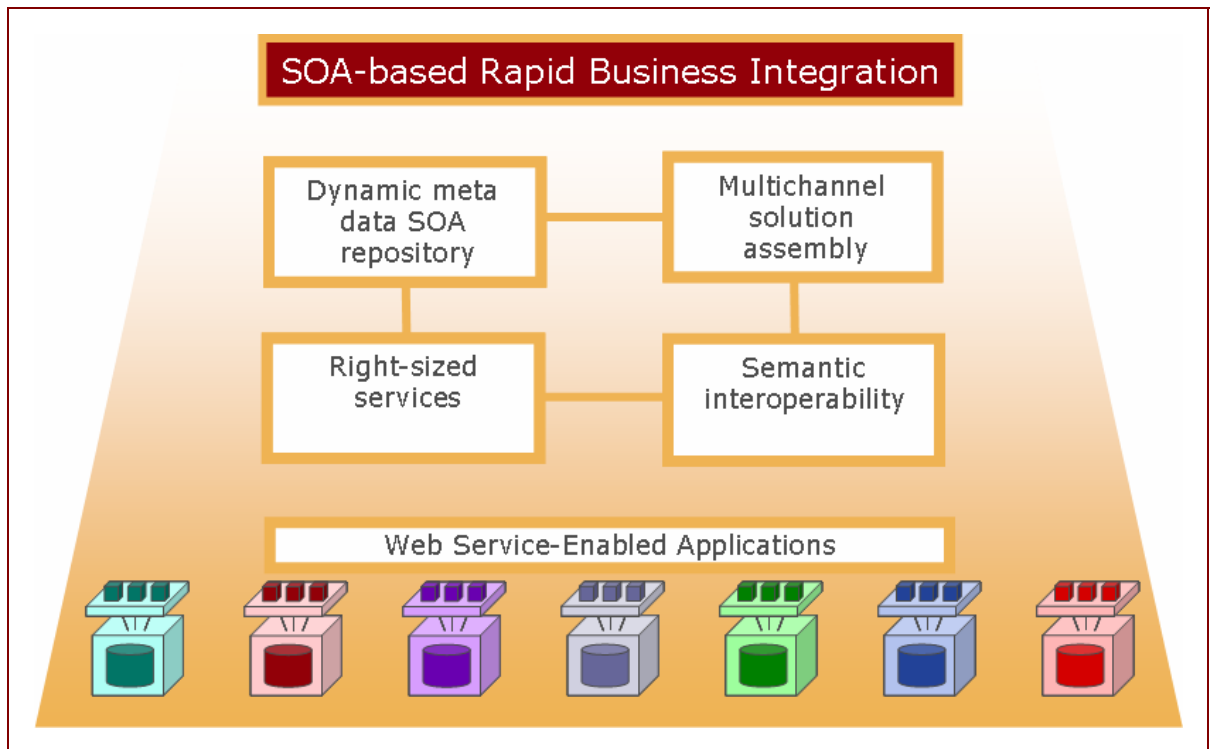
[†] Product Review, *Web Services Journal*, December 2004, pp 38–39

span numerous integration styles from user-centric interactive applications to event-based process integration to batch-style synchronization.

Multichannel Deployment

Agility means more than the rapid assembly of business solutions. It also means deploying those solutions wherever and however needed. In sectors as diverse as banking, insurance, healthcare, government, retail, transportation, media, and manufacturing, enterprises are formulating strategies that reach customers and business partners through a variety of channels including the Web, mobile devices, portals, thin clients, and even embedded within users' primary applications. Moreover, enterprises need to support multichannel deployment with a single integrated view of their customers.

With Above All Software's Composite Application Platform, developers can readily deploy a composite application across multiple channels by creating and reusing enterprise services. They can also assemble an integrated view of the many transactions that take place through these multichannel applications. All this is possible because the composite applications themselves are not coded but are rather defined via metadata, rendering them uniquely suitable for multichannel deployment.



Above All Software bridges the gap between existing applications infrastructure and SOA benefits.

Conclusion

In light of SOA's compelling vision, many companies have already launched SOA initiatives, and even more are about to embark on them. These companies are motivated by carrots and sticks. The stick is the need, the imperative, to create and recreate enterprise solutions at the break-neck speed of today's changing business environment. The carrot is the potential to achieve such agility by reusing an enterprise's existing IT assets—legacy applications, middleware, mobile platforms, mismatched data sets, and so on—in the rapid assembly of composite applications at considerably reduced cost and with considerably reduced technical skill requirements.

Carrots and sticks notwithstanding, enterprises are discovering that the road to a service-oriented architecture is unpaved. Indeed, a sizable enablement gap exists between the practical reality and the vision of a service-oriented architecture. Specifically, you'll need to establish how services with no knowledge of each other will interoperate semantically. You'll need to refine low-level services into the coarser business-oriented granularity of enterprise-class services. You'll need to provide centralized and managed access to the Web services and other software services in your repository. In sum, you'll need a comprehensive suite of tools to power you across the enablement gap. Only with this foundation can you rapidly assemble the high-impact integration solutions known as composite applications.

Whether just embarking on an SOA initiative or further advanced, Above All Software's Composite Application Platform can help you extract the maximum business value from your existing IT assets. Built from the ground up to be an essential element of your SOA, it dramatically simplifies the steps required to leverage your past IT investments to meet continually changing business needs.

Next Steps

Above All Software provides award-winning business integration software that allows customers to leverage service-oriented architectures to meet ever-changing business needs. Service-oriented integration provides a simplified, flexible approach that is faster and more cost effective than traditional methods. With Above All's Composite Application Platform™, customers transform existing IT investments into rapidly assembled, high-impact business solutions. For more information about Above All Software, visit our Web site at www.aboveallsoftware.com or call 800-819-5530.