ALICIA BUELOW

Flexible IT, better strategy

John Seely Brown and John Hagel III

IT's critics say that it lacks strategic importance. So why does technology keep getting in the way of good strategy?

Technology architecture is one subject guaranteed to make a chief executive's eyes glaze over. For many CEOs, the topic is mysterious. Even for those who understand technology better, it is a sore subject because today's IT architectures,[1] arcane as they may be, are the biggest roadblocks most companies face when making strategic moves.

Strategists have largely discredited classical notions of strategy formation. The empty biennial reviews of yesteryear are gone, superseded by "radical incrementalism,"[2] which emphasizes rapid waves of near-term (6- to 12-month) operational and organizational initiatives brought into focus by a shared view of a company's much longer-term (five- to ten-year) strategic direction. A quick sequence of focused incremental shifts can produce

[1] An IT architecture is the overall structure of and interrelationships among the data, business logic, and interfaces of a company's computers and other hardware, applications, databases, operating systems, and networks.

[2] Radical incrementalism is related, but with crucial distinctions, to theories such as strategy by experimentation, the portfolio of initiatives, and time pacing. For radical incrementalism (discussed as "layered strategies"), *see* Chapter 9 of John Hagel III, *Out of the Box: Strategies for Achieving Profits Today and Growth Tomorrow through Web Services*, Boston: Harvard Business School Press, 2002; for strategy by experimentation, *see* Eric D. Beinhocker and Sarah Kaplan, "Tired of strategic planning?" *The McKinsey Quarterly*, 2002 special edition: Risk and resilience, pp. 48–57 (www.mckinseyquarterly.com/links/7473); for the portfolio of initiatives, *see* Lowell L. Bryan, "Just-in-time strategy for a turbulent world," *The McKinsey Quarterly*, 2002 special edition: Risk and resilience, pp. 16–27 (www.mckinseyquarterly.com/links/4916); for time pacing, *see* Kathleen M. Eisenhardt and Shona L. Brown, "Time pacing: Competing in markets that won't stand still," *Harvard Business Review*, March–April 1998, pp. 59–69. The key distinction between radical incrementalism and these other theories is that it emphasizes the need for a clear but high-level view of a company's longer-term (five- to ten-year) strategic direction to put more near-term initiatives, or "experiments," into context. Without this long-term context, the near-term initiatives begin to lose coherence.

cumulative and radical change that isn't easy to copy. Radical incrementalism has helped companies such as Charles Schwab, Dell, Microsoft, and Wal-Mart Stores reshape industries and deliver superior returns to shareholders.

Yet radical incrementalism is notoriously difficult to accomplish. Operational shortcomings and organizational inertia hinder companies from making near-term innovations in business practice and process. Technology can be an even bigger hindrance—in part because it's so deeply embedded in operations and organization, in part because information systems are rigid.

This inflexibility is endemic today. Big suites of enterprise-wide applications like those in enterprise-resource-planning (ERP) suites, designed to integrate disparate corporate information systems, dominate client-server architectures. Unfortunately, the onetime, "big-bang," and tightly defined way in which these applications have been implemented, as well as their massive bodies of difficult-to-modify code, mean that enterprise applications integrate businesses only by limiting the freedom of executives. Introducing a new product or service, adding a new channel partner, or targeting a new customer segment—any of these can present unforeseen costs, complexities, and delays in a business that runs enterprise applications. The expense and difficulty can be so great that some companies abandon new business initiatives rather than attempt one more change to their enterprise applications. Far from promoting aggressive near-term business initiatives, enterprise architectures stand in their way.

The problems can be so great that some companies **abandon** new business initiatives rather than face changing their enterprise apps

The good news is that just as the limitations of the current generation of IT architectures are becoming painfully apparent, new methods of organizing technology resources are appearing. IT is on the verge of a shift to a new generation of "service-oriented" architectures that promise to go a long way toward reducing if not removing current obstacles to new operational initiatives. These new architectures are no panacea; technology in isolation has never created strategic value. But service-oriented architectures *will* enable companies to introduce new business practices and processes more rapidly and at lower cost. Such innovations will accumulate by increments into strategic advantage.

Service-oriented architectures don't require the removal of existing IT resources and in fact were developed specifically to help businesses get more value from them. These architectures (*see* sidebar, "Getting there from

here") are still in their early days, but companies such as Eastman Chemical, General Motors, and Merrill Lynch have already begun to experiment with them. Companies that follow suit will break free from the constraints of today's architectures and become capable of leveraging IT—mostly for the first time—to gain strategic advantage.

Getting there from here

Service-oriented architectures are in the first stages of emergence; the current deployment of Web services technology is a promising early initiative in this direction. In particular, the foundation standard—the Extensible Markup Language (XML)—provides a major advance by creating ubiquitous standards for presenting data and for defining the interfaces that loosely coupled connections require. But other standards and protocols for service-oriented architectures, particularly standards and protocols relating to security and the management of business processes, will have to become more robust before they can fully support mission-critical, long-lived transactions.[1] Even for companies that have started to focus on service-oriented architectures (such as Eastman Chemical, General Motors, and Merrill Lynch), these additional emerging standards and protocols remain largely conceptual. As of now, the use of Web services technology tends to be very limited in scope and targeted at a specific area of a business.

However, early implementations are creating optimism about the business appeal of these architectures, which can not only provide much greater flexibility in supporting business operations but also be put into service in a more incremental fashion than previous generations of IT architectures. Each stage of implementation can be geared to specific business initiatives and, with relatively modest investments and short lead times, deliver tangible business value. Service-oriented architectures thus represent an inflection point shifting enterprises from rigidity to true flexibility and also leveraging vast resources that are already in place by exposing them and making them accessible as services.

Companies don't need to shift their entire IT architecture or to remove existing IT resources to enjoy the business benefits of a service-oriented architecture; they can begin with a focused effort to support specific initiatives. At first, relatively few IT resources will be accessible in a service-oriented architecture—for the farm-equipment maker discussed in the body of this article, only the supply chain applications in the manufacturing plants. But the accessible resources will be those most relevant to near-term operating initiatives. The business benefits (in this case, the operating savings from direct customer access to order-status data and the revenue benefits from more satisfied customers) will help finance this first stage of the transition. Over time, the design principles of service-oriented architectures will lead to the development of entirely new services.

[1] For more about the emerging standards, protocols, and services needed for mission-critical transactions, *see* John Hagel III and John Seely Brown, "Service grids: The missing layer in Web services," *Release 1.0*, December 2002, Volume 20, Number 11, pp. 1–33.

The agony of customized connections

How are today's IT architectures an obstacle to more agile strategies? Let's consider the example of a company that produces and sells farm machinery. Senior management has looked ahead five to ten years and decided that the best way for the company to continue creating value would be to evolve into a customer relationship business, thus deepening its ties with a clientele of large agribusinesses and serving a broader range of needs.

What can the company do during the next 6 to 12 months to accelerate this change in direction? Suppose it decides to focus on two initiatives. The first is intended to provide customers with better, more easily obtained information about the status of their orders—both to improve customer service and to reduce operating costs—since the company maintains a large order-processing staff to answer time-consuming customer inquiries. The second initiative aims to expand the company's range of products by allowing it to source and then resell some complementary ones from third-party manufacturers.

On the face of it, neither initiative seems particularly daunting. Let us say, however, that the company manufactures its products at three US plants, two of them acquired from other companies. Since different companies owned the facilities, each of them uses different enterprise applications to run its operations. Employees checking on the status of orders therefore have to access three separate applications with very different user interfaces and ways of presenting product information. This problem explains why the order-entry staff spends so much time fielding queries.

What would be needed to make the information directly accessible to the purchasing systems of customers? For each of them, the company would have to create three custom-designed connections linking the customer's purchasing system with the enterprise suites of the three manufacturing plants—connections that would have to translate information for product descriptions, shipping instructions and status, and other key data.[3] The custom connections should also provide for adequate security, transaction monitoring, and message routing. Even if these needs were ignored, the connections would demand a deep understanding of the applications at either end and of the features specific to them. The connections would be not only

[3] Electronic-data-interchange (EDI) systems, which preceded service-oriented architectures, permit companies to exchange information in standardized forms, thereby avoiding the need for custom connections. But EDI, focusing mostly on order and shipping information, has very limited application. And while EDI makes it possible to exchange data, custom connections are still required to translate the information for the applications that use it.

expensive to create (because of the coding time required) but also unlikely to be reusable for anything other than their original purpose. Technologists aptly describe custom-designed connections as "hardwired," because they lack flexibility.

As for the second initiative, if the company wanted to resell third-party manufacturers' products and to give customers for them the same level of order-status information, it would need to create custom-designed connections between each customer and every supply chain application that its suppliers happened to run.

Even during the early deployment of the company's first initiative, its complexity, cost, and lead times would mount. If the company later wanted to enhance each of these connections—by giving customers a limited ability to modify orders before they were shipped, for example—that feature would have to be coded into every customized connection. Suppose too that some of the company's first product suppliers didn't work out and had to be replaced. It would then have to create entirely new connections. The more business conditions changed, the greater the complexity, the cost, and the lead times.

Under today's information architectures, if you need to connect two applications, databases, or operating systems—or to connect any of these to human beings—each connection must be specially created for its specific purpose, and even the smallest modification requires it to be recoded. Worse yet, the expense and effort needed to establish connections across technology resources increase exponentially—not linearly—with the number of resources connected.[4] Small wonder that companies spend large portions of their IT budgets on integration as they create new connections and redesign old ones to keep up with changing business conditions.

Creating new kinds of connections

Service-oriented architectures take a different approach to connecting technology resources. Instead of customized, hardwired connections, these architectures rely on "loosely coupled" ones in which IT resources, even if they use incompatible operating systems or different vocabularies in their own operations, can be joined together easily without friction or customization and just as easily disassembled and reassembled. (Of course, this assumes that all participants have agreed on a standardized vocabulary to serve as a common translation overlay.)

[4] This is the infamous "n-squared" problem, which rears its head as companies add participants to a network. *See* John Hagel III, *Out of the Box: Strategies for Achieving Profits Today and Growth Tomorrow through Web Services*, Boston: Harvard Business School Press, 2002.

All of the information required (for instance, which outputs the software can deliver, how they are accessed, and who is authorized to do so) is described and contained in the interface—the electronic description that other applications use to find out how to establish an electronic connection. At the farm-machinery company, the interface to the manufacturing application might specify that it could provide information on a product's manufacturing status and indicate what protocols and standards the "service's" user (another piece of software) would need to access the information.[5]

Of course, the information must be presented in a way that is broadly understandable, and this is the key role that standards and protocols play in supporting loosely coupled connections. Unless the standards and protocols are widely adopted, the range of feasible connections is very limited. The rapid spread of the Extensible Markup Language (XML) as a foundation standard, and the whole series of standards and protocols derived from it, provide an effective framework for broadly understandable and accessible interfaces.

Much more flexible connections can be established once the standards and protocols are in place. Since computers can "read" and understand them, connections can be automatically created as the need arises. The connection focuses on the service's output rather than on the details of how it is generated. Thus, connections can be established much more quickly, without requiring a deep understanding of the underlying features at each end of the connection.

In effect, service-oriented architectures embody a modular approach to organizing IT resources. As with all such approaches, the key requirements are standardized definitions of interfaces so that different technology resources become self-describing, which allows them to be mixed and matched quickly and easily to meet the requirements of the moment. As new operational initiatives are launched, appropriate applications and information resources are accessed dynamically to support those changes.

The business benefits

To understand the full benefits of service-oriented architectures, let's turn again to the farm-machinery manufacturer. How would such an architecture help it achieve the near-term operating flexibility required for radical incrementalism and other new approaches to strategy?

The architecture would expose, or make visible to other applications, the features and capabilities of each of the company's supply-chain-management

[5] Services are created when an application's features—for example, calculating a currency's value in terms of another currency—are "exposed," or made visible. This visibility is generated through an interface that describes what data, features, or both are available in the application and how to access it.

applications. The company's IT department would accomplish this by creating a standardized interface allowing, say, an order's status to be available as a service that could be shared by any application using the same standards and protocols.

Access would come through a loosely coupled connection established only "on the fly" when needed, because all of the necessary information would already exist in the service interfaces. If the consuming application (in this case, the customer's procurement software) didn't already know which manufacturing plant to query for an order's status, a directory service could help identify the appropriate services. A variety of enabling services of this kind could be delivered as loosely coupled services shared across all connections rather than designed into each of them in advance.

The advantages are significant. Traditional approaches require programmers to design a customized new connection in advance for each pair of resources that might need to interact. A service-oriented architecture requires only a onetime investment to write code that allows the service to be accessed by any application with an interface adhering to the same widely available standards and protocols, such as XML and the Web Services Description Language (WSDL).[6] In contrast to hardwired connections, which have less reusable code (so that each new connection represents a substantial programming effort), the initial investment in a service-oriented architecture is amortized further each time a new connection is created.

With services designed to be context free—designed for use by any application or in any business-application environment—companies can more quickly mobilize services that are already available and deploy them in new business contexts. In this way, those companies will enhance their ability to create and test new products, to redesign business processes, and even to implement new business models rapidly, because they will have less concern about the potential disruption of their ongoing business activities.

Let's say that the farm-machinery company wants to streamline its manufacturing operations in one plant and to modify the manufacturing application used there. In architectures dominated by enterprise applications, every custom-designed connection between that manufacturing application and each customer would need to be retested and possibly reconfigured because the proper functioning of the connections depends on assumptions about how information is generated. With loosely coupled connections, which

[6] For more about Web services terminology, *see* Samir Patil and Suneel Saigal, "When computers learn to talk: A Web services primer" (www.mckinseyquarterly.com/links/7475).

don't require such assumptions, the company can try out a new business process in one of its plants without worrying about unanticipated malfunctions in the plant's customer IT systems.

Service-oriented architectures thus make it possible to create more flexible connections across applications, and this is probably how such architectures will first deliver value to the companies using them. But that is only the beginning. As more functionality becomes exposed in the architectures, companies will use this functionality to orchestrate business processes in a much more sophisticated way: the architectures will become the foundation for loosely coupled business processes delivering even more flexibility.[7]

Service-oriented architectures will become the foundation for loosely coupled **business processes**

Consider again the farm-machinery company. At present, to fulfill an order, specific parties must undertake different tasks in a specific sequence, and any change to it or to the parties means that the order-fulfillment application must be modified. With a service-oriented architecture, both the sequence and the parties can be specified at the time of the order. If, for example, the customer needs financing from a particular institution and wants to have the machinery shipped before the financial arrangements are fully confirmed—an exception to standard procedures and thus something the application's original programmers hadn't anticipated—the company can reconfigure its business processes to meet these exceptional needs.

Much as companies have become familiar with modular product design, they will use service-oriented architectures to implement modular business processes. Such processes will make it possible for companies to mix and match tasks and for resource providers to facilitate both process innovation and responsiveness to changing markets on the fly. The early focus on creating more flexible connections will inevitably lead to fundamentally different business processes.

Beyond the firewall

Loose coupling also supports business innovation beyond the boundaries of individual enterprises: this approach can, for example, be very helpful in automating connections with business partners, thus making it easier for a company to add value for its customers by accessing resources that its business partners own. Loose coupling will also accelerate a more fundamental

[7] *See* John Seely Brown, Scott Durchslag, and John Hagel III, "Loosening up: How process networks unlock the power of specialization," *The McKinsey Quarterly*, 2002 special edition: Risk and resilience, pp. 58–69 (www.mckinseyquarterly.com/links/7477).

unbundling of the enterprise, allowing companies to focus more tightly on activities in which they are distinctive and to rebundle other assets and capabilities from a broader range of partners. The farm-machinery manufacturer, for instance, could use service-oriented architectures to move more quickly to its goal of focusing on its core strength: customer relationships. Since the company's automated connections will give it the ability to communicate with contract manufacturers in a more visible and coordinated way, it can off-load more and more of its product-manufacturing activities to specialized companies.

Furthermore, service-oriented architectures automatically generate rich information about the performance of the connections and the outputs at each end, simply as a by-product of managing connections. By contrast, capturing such information from manual connections is time-consuming, error prone, and costly. Automation gives business decision makers much better information about the overall performance of both IT and the business and helps them root out inefficiency. Our farm-machinery manufacturer, for instance, would automatically generate detailed information about customer inquiries into the status of orders. The company could then improve customer service by systematically cataloging the information and using it as the basis of proactive order-status reports for certain types of customers and orders.

The power of emerging service-oriented architectures is hard to ignore. They make it easier to implement radical near-term changes to business practices at the local level—changes that can lead over time to radical changes in overall business practices and business structures. That is the essence of radical incrementalism, and of good strategy. Q

John Seely Brown, the former head of Xerox's Palo Alto Research Center and chief scientist at Xerox, can be reached at jsb@parc.com; **John Hagel**, an alumnus of McKinsey's Silicon Valley office, is now an independent business consultant and can be reached at www.johnhagel.com.