

The SOA Magazine

Feature Article



Modernizing the Mainframe for SOA: Considerations for Transforming Mainframe Assets into Services

by Dan Finerty

Published: April 3, 2007 (SOA Magazine Issue VI: April 2007, Copyright © 2007)

[Download this article as a PDF document.](#)

Abstract: In order to realize the core business imperatives of increasing agility and efficiency SOA demands that organizations reconsider the value of legacy technologies and examine ways to extend and enhance that value rather than taking a “toss the baby out with the bathwater” approach to modernization.

This article discusses how to transform and modernize the mainframe into an active, industry-standard participant of SOA implementations. It provides step-by-step guidelines to ensuring that the mainframe’s strengths are leveraged, allowing it to become a full-fledged member of service-oriented solutions. Available technology approaches that enable the mainframe to both provide and consume Web services are also discussed.

Introduction: The Mainframe as an IT Asset

Business organizations rapidly adopting service-oriented architecture to take advantage of the increased agility and competitive advantages it affords often face tough decisions regarding mainframe systems. Typically decades old, many of these legacy systems were designed before the concepts of Web services or SOA were even thought about. Nevertheless, they can comprise the very heart of an organization’s business processes.

Mainframes usually contain vast stores of data, represent years of development investment, and are central to mission-critical operations. They also typically do what they do exceedingly well and are extremely dependable. It is because of these valued attributes that the mainframe has survived for so long and remains well ensconced in the enterprise today. More commercial transactions are processed on mainframes than on any other platform, and over 90% of Fortune 500 companies still rely on them. In fact, it has been reported that more transactions are currently processed by IBM CICS and IMS systems than by the Internet in its entirety. So it’s easy to see how organizations seeking to modernize are hesitant to replace these environments.

The process of replacing the mainframe platform and moving its functionality to a different platform entails considerable time and expense (and that’s if it is an unqualified success). The probability of disruption and intermittent, partial, or even system-wide failure is also high. Considering the mission-critical nature of many mainframe-based applications, these impacts can have seriously adverse effects on productivity and revenue as well.

The Mainframe and SOA

An alternative to the actual replacement of large mainframe investments is to make SOA a strategic means for drawing further value out of mainframe assets. Given the right tools, non-standard or proprietary assets on the mainframe can be transformed so that they adhere to a reusable, component-based SOA framework. All application development in support of an SOA initiative can then be accomplished via industry-standard Java and/or .NET tools having multi-vendor support and requiring no specialized mainframe programming skills. Without having to tamper with its tried-and-true, mission-critical mainframe software infrastructure, the organization can still transform its legacy system into a more flexible, services-based application that better addresses current and future business needs.

The concept of non-invasiveness touched on here is important beyond the obvious virtue of mitigating disruption and risk to the operation of critical systems. The availability of legacy skill sets is on the wane, and personnel possessing them command high salaries or fees. An invasive solution - one that involves re-writing mainframe application code to make those applications adapt to a service-oriented architecture - consumes those scarce and high-priced resources. And, it furthermore expends those resources on restructuring something that already works without adding anything of real value in and of itself.

Since the whole point of modernizing legacy platforms is to take advantage of new technology in introducing new functions to employees, customers, and/or business partners, this approach involves an unnecessary exercise of effort and expense. The aforementioned resources are better left maintaining the proper operation of the existing mainframe systems, while new application development is relegated to more abundantly available Java and .NET experts.

It's been well documented how SOA derives values from the reuse of existing business functionality in the form of components and Web services for developing new functions. The goal of mainframe modernization is to identify specific business transactions from within multiple legacy systems and expose these transactions as reusable components and Web services, maximizing productivity and increasing the ROI of the organization's legacy technology investments. If portal developers can see mainframe assets not as a proprietary obstacle, but as something they're accustomed to working with every day, they can be build and deploy new applications faster. More importantly, if an organization can reuse non-standard mainframe assets through the use of Web standards without requiring any changes to be made on the mainframe side, that organization will have a sufficiently large resource pool for making mainframe modernization a reality.

The Modernization Process

The first step to making the mainframe a full participant in an SOA is to understand what parts (assets) of the mainframe environment need to be incorporated into this reuse-centric framework. Those parts can be classified under three different categories:

- *Data* - Some 70 percent of the world's "system of record" data resides on mainframe platforms, representing vast volumes of information not necessarily organized in a relational format. The ability to access and manipulate this data in new ways can be invaluable.
- *Business Logic* - Legacy systems typically involve a jumble of different application environments with distinctly different programmatic interfaces. Some of the older applications, in fact, lack any programmatic interface at all, having been designed for a self-contained, monolithic environment where reuse was not part of the picture. Exposing these assets as reusable components can have enormous impact on productivity and ROI.
- *Screen Logic* - For some of the older applications where programmatic interfaces were lacking, organizations have traditionally employed the more or less pseudo-programmatic integration approach known as terminal emulation or "screen scraping." More modern tools are available today that offer dynamic introspection technology, allowing organizations to work with screens in a truly programmatic fashion.

As you might extrapolate from this list, if these mainframe environments are to be effectively extended to distributed platforms within a global, enterprise-scale business it will involve hundreds, if not thousands, of point-to-point integration connections, each with its own attendant hardware, software, and personnel. This redundant layer of complexity adds multiple potential points of failure as well as layers of inefficiencies and hidden costs.

SOA entails a distinct departure from this form of integration architecture. It provides a model capable of fully leveraging the non-proprietary Web services platform, allowing it to be positioned as a foundation for a distributed, service-oriented approach to messaging-based inter-connectivity. In fact, this type of platform forms the basis of enterprise service bus (ESB) products.

By the same token, the heterogeneous mainframe-reliant organization should incorporate into this same model all mainframe connectivity — in essence, taking a "mainframe services bus" approach. As a result, both data and business logic (as well as screen logic) become seamlessly available via industry-standard technologies such as XML, SOAP, and SQL.

For Example

Let's say an organization has decided to transform certain mainframe systems into a set of services in a non-invasive

manner. Next on the agenda is considering how that transformation is to take place. Integrated Development Environments (IDEs) are available (based, for example, on the open source Eclipse framework), which can be employed in administering the transformation.

Ideally designed for the use of administrators rather than programmers, such a utility should provide a single GUI-based development environment to handle any integration requirement without demanding detailed knowledge of the mainframe or distributed technologies. It should be able to expose mainframe applications and data within the IDE as Web services, real-time business events, or as direct SQL calls, essentially transforming the mainframe into an industry-standard enterprise server. Such an environment can enable architects and developers to build new applications quickly and cost-effectively. Equally important to making the mainframe a full participant in a service-oriented enterprise is the use of transformation utilities capable of exposing existing services to the mainframe itself.

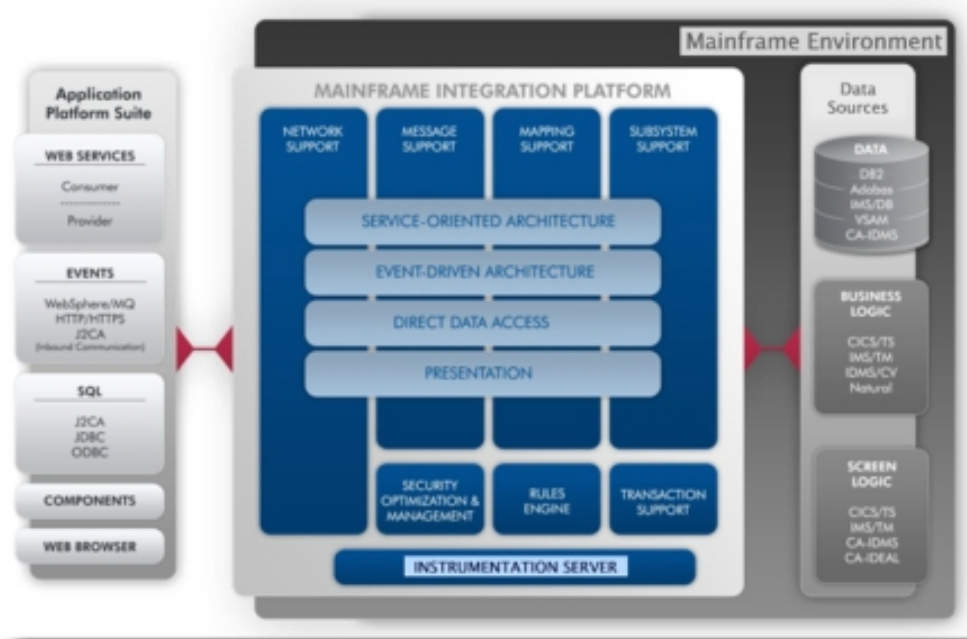


Figure 1: A modernized mainframe environment tailored to support SOA.

Deployment Considerations

To this point the discussion has focused on why and how an organization would go about implementing a robust transformation engine to provide mainframe access within an SOA. But it should be noted here that what really provides value is the way in which this environment is actually deployed and managed. In making the mainframe participate as part of an SOA, it is important to ensure that doing so does not compromise the very attributes that make the mainframe such a valued part of the enterprise in the first place - that is, that the new extended environment inherit as many of the following vital attributes as possible:

- Predictable Performance
- Scalability
- Security
- Reliability

Furthermore, any transformation engine ought to allow services to leverage some of the more sophisticated hardware and software features available in today's mainframes. For example, the transformation engine should be fully mainframe-aware so as to enable communication between logical partitions (LPARs).

Management Considerations

As previously noted, two of the valued attributes of mainframe systems are predictable performance and scalability. Maintaining these attributes from an SOA standpoint requires visibility into the transformation engine, which can reveal events that take place before, during, and after unforeseen errors occur.

Because the mainframe typically handles an extremely large volume of transactions, managing an SOA environment that includes a mainframe system can be a real challenge. If and when an error condition arises, it is frequently not because of what a user did per se, but is instead due to a particular chain of occurrences of which the user's action played merely a (usually small) contributing role. With the nearly infinite permutations of events possible due to the mainframe's handling thousands of things for thousands of users, activities that ordinarily work flawlessly can suddenly encounter unexpected combinations of conditions in which they fail.

In a point-to-point architecture with one solution for each data source, one for each business logic routine, etc., large-volume management would be all but impossible. For an organization integrating mainframe assets into an SOA environment, then, what is called for is a single console solution. This type of solution provides for real-time, in-production monitoring of all activities in the transformation engine that enables the organization to manage its operations (including problem diagnosis) while supporting thousands or even millions of production transactions per hour.

The ability to administrate a mainframe as part of a service-oriented enterprise in a manner that carries over the reliability and performance of the mainframe itself is critical. Sharing and exposing an organization's resources on that scale increases the risk to the business simply because any mishap will impact so many users. It is essential to know what is going on and, more importantly, that sufficient information is readily available to resolve any problem in a timely manner.

Security Considerations

The sheer volume of data and other resources on the mainframe also make it necessary to ensure that integration of these assets into an SOA environment includes another of the aforementioned attributes: security. A common challenge that is likely to arise with services implemented as Web services is related to the fact that SOAP messages (which are generally delivered via the stateless HTTP protocol) are authenticated and authorized with every SOAP transmission. This raises a real concern: you don't want users accessing a core system without authenticating themselves and receiving authorization to perform a given function. However, legacy platforms impose security in an entirely different manner.

The mainframe traditionally operates in terms of sessions: a user logs in at a terminal to begin a session, performs numerous transactions, and logs off at the end of the day. Let's say a user performs a thousand transactions in that session. If this same security approach were now applied to Web services-based service-oriented solutions, those thousand transactions would need a thousand authentications and authorizations. This is because users are no longer in a session but are part of an independent, stateless SOAP environment, where any number of users can request access at any given time.

This harks back to what was said earlier about the transformation engine being mainframe-aware in order to leverage advantageous attributes. The IBM z/OS operating system, for example, offers something called SAF, a security API that lets the OS interface into a security infrastructure in a different environment such as an SOA. A robust mainframe transformation engine for SOA must therefore be capable of addressing the new security workload, while maintaining the existing security infrastructure.

Reliability Considerations

Another key mainframe attribute that needs to be taken into account is reliability. A corollary of reliability is risk mitigation, which was discussed earlier as one of the factors driving the need for a non-invasive, mainframe-SOA solution: if something is not broken and you don't change it, it won't break. Any change that is made introduces the risk that what was changed will not be as good as yesterday (or may cease working altogether). But one of the great values of SOA is its ability to enable rapid change by adapting to shifting business requirements.

Therefore, a final management-related consideration for implementing a solution that incorporates the mainframe into an SOA is how it handles change management over the software development lifecycle. Organizations should be able to automate, control, and manage the promotion of services from the development environment to QA and right through production. They should avoid any tool that lets them build a service which, when they try to move it from one environment to another, requires them to essentially rebuild it (a consideration especially relevant to services implemented as Web services). Organizations essentially want to be able to integrate into those existing piece management processes in order to minimize risk as they incorporate the mainframe within the SOA.

Conclusion

Given the rapid industry adoption of SOA, the viability of mainframe systems needs to be re-evaluated. Modernization strategies can be employed to realize mainframe enablement specifically within service-oriented environments. Quality of service attributes must be extended on the part of the mainframe in support of SOA development initiatives. Organizations have at their disposal industry standard technologies to transform the mainframe into a reliable, malleable, and adaptive resource suitable for service encapsulation.

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[home](#) [past issues](#) [contributors](#) [official book site](#) [legal](#) [rss](#)

Copyright © 2006-2007 SOA Systems Inc. All Rights Reserved