Commentary

A Service-Oriented Architecture Can Save the Mainframe

Enterprises that implement service-oriented architectures on their mainframes can extend the life cycles of their systems; however, many have been reluctant to incur the costs required to attain high levels of granularity.

Large enterprises have tried a number of approaches to eliminate rigid legacy systems from their portfolios, but they have met with limited success. Despite packaged conversion or platform transformation, many find themselves left with a significant portfolio of legacy systems. Although some applications are simply too archaic to save in their current implementations, others are more flexible and can be leveraged by exposing the business rules embedded within.

Gartner espouses a development strategy that employs a service-oriented architecture (SOA). The service-oriented development of applications (SODA) framework provides the development mechanisms to implement an SOA. However, with little interest in anything invasive, enterprises have used legacy extension technologies to wrap current transactions and expose them through alternative presentation mechanisms (such as Java or HTML), or as programmatic objects (including the Component Object Model and Java).

This approach works when the granularity of the business function (as represented by the "screen") is acceptable. By using programmatic integration servers, for example, you can package a flow of screens into a coarser granularity service than a simple screen — or you can expose only a subset of the screen through a service interface. While this "pseudo-SOA" approach works, true flexibility and "maintainability" are provided via finer-grain business functions that do not map to their traditional legacy presentation layers.

SODA for Mainframes

The SOA approach describes the framework for exposing business functions as "services" — that is, those services that can be accessed by applications that need them. In traditional mainframe legacy systems, these business functions are written to be exposed to individuals via a 3270-style presentation layer.

A new development-environment category is emerging that enables the assembly of composite applications based on services. "Web services producer platforms" are a set of tools, frameworks,
engines and services that enable this style of construction (see "Producer Platforms and SODA Will Shift the AD Approach"). These platforms are focused primarily on Web services, but they can also support composition without using the Web services approach. The keys to these platforms are automation and flexibility, through the ability to quickly maintain or enhance applications that are assembled using this approach. Producer platforms for the mainframe are slowly evolving and will require a level of tool integration by the user to provide these functions. However, they are possible and we recommend using them.

Software services are components that do not require a developer to use a specific component model. They enable the developer to create programmatic interfaces that can access any underlying platform model — such as .NET or Java 2 Platform, Enterprise Edition (J2EE) — without having to be specific to any single model. When building applications that rely on services, developers must recognize that services are not static components. Rather, software services come with a consistent behavioral aspect — that is, a service will behave in different ways, but over time, this behavior becomes a pattern. This is because a service is an operational set of functions that an enterprise must manage for quality control and availability. There is no better environment in which to source services than the installed base of working, time-tested business functions that are implemented in legacy systems.

Legacy Systems — "Pandora's Box"?

Wrapping current business functions is possible; however, to provide the greatest flexibility in reusing current systems, finer-grain services must be exposed. This requires a willingness to restructure current applications, isolating (as much as possible) the presentation layer from the business logic. It is unnecessary to completely re-engineer these systems or isolate the presentation layer, the business logic and the data access layer. However, it is beneficial to restructure these systems to enable service-oriented access as well as the current presentation mechanism (3270).

Continued refinement of these systems into increasingly finer-grain business logic is an evolutionary process that should occur over time, balancing the need for ultimate flexibility with the performance requirements necessary for these systems. "Balance" is something that enterprises must discover for themselves.

Although restructuring current systems into fine-grain components increases the flexibility when developing new applications, it can also have a negative effect on system performance. Also, established systems were not designed for reuse, and they often contain behaviors whose meaning has been lost over time. Enterprises that successfully implement SOAs on the mainframe will use legacy understanding and legacy transformation tools to create "right-grain" services (that is, services that are not too small and do not negatively affect performance, but neither are they too big to hinder reasonable use) and identify the lost meanings.

Web Services on the Mainframe

We have previously addressed the applicability of exposing current business functions on the mainframe as Web services (see "Web Services and the Mainframe: Reality or Ridiculous?"). Enterprises that accept SODA as their future application strategy must also include current mainframe applications. While options exist to transform applications off the mainframe into J2EE or .NET platforms, these options are most appropriate for those enterprises at the lower end of the mainframe MIPS scale (that is, fewer than 200 MIPS). Those enterprises with a greater investment in mainframe MIPS must begin to address the role that mainframes will play in their application strategies in the next five to 10 years.
We do not underestimate the difficulty of this decision, but we believe an SOA enables these enterprises to leverage their current applications and platforms while evolving to a J2EE or .NET architecture. This is actually the single-biggest application development decision that enterprises need to make. These architectures are maturing, but they have not shown the ability to completely replace the huge, mission-critical, industrial-strength applications that continue to execute on the mainframe in Customer Information Control System (CICS) or Information Management System (IMS).

While we believe that most large enterprises will continue to have both J2EE and .NET platforms in their portfolios, they must decide how to integrate these traditional mainframe systems into their enterprise architectures. A service-oriented approach — particularly using Web services — provides a mechanism to integrate these applications with either platform, since they both can consume Web services.

**Bottom Line:** Mainframe-based enterprises can increase their flexibility and reduce their risk by restructuring established systems around a service-oriented architecture (SOA). However, an SOA requires a willingness to undertake invasive "re-architecting" of these systems. In the past, most enterprises have shown little interest in doing this, leaving them with tactical extension or highly risky transformation approaches. The amount of change required is a function of the granularity desired from established systems. The greatest gain comes from the finest grain, but this requires the most restructuring and the highest expense.