The Missing Business Model

Web sites have a hard time interoperating between themselves. Most are the same architecturally, but components developed in one technology cannot invoke methods on components developed in another technology. Another complication: The firewalls in every site disallow binary calls. Developers today handcraft interoperability solutions between businesses. These solutions are proprietary, provide singular relief, and are expensive and time-consuming to implement.

Web Services address these problems in an elegant and simple fashion. Web Services use HTTP as a transport protocol to transport calls from one Web site to the next, and they use a higher-level protocol such as Simple Object Access Protocol (SOAP) to represent the call payload. Web Services development platforms, such as .NET, enable you to expose and consume Web Services with great ease, almost as if the interacting components were in the same Web site.

Microsoft promotes actively the "software as a service" notion. It envisions businesses exposing functionality over the Web in the form of Web Services, where interested parties can consume them as part of their applications. Here's the critical, unanswered question: What is the business model for Web Services? In other words, how can companies make money from Web Services? I see a few possible business models. Companies can charge a per-use fee every time somebody uses their services. Or they can run a subscription program where they collect a flat fee and grant unlimited access to services. If the service facilitates some business transaction, the service provider can collect a percentile commission of the transaction itself. Of course, companies can also expose their services for free, then attempt to collect a reward for their efforts in some other fashion, whether in the form of recognition, market penetration, or access to other services. Companies can also use a combination of these business models, such as giving a basic set of services free (for market penetration) and charging only for the premium services.

But all these business models have a flaw that makes committing to Web Services risky from the service consumer's point of view: What prevents the service provider from changing the service definition? Changing a published service (its syntax, semantics, performance, error handling, and so on) will likely break all existing consumers of the service. Imagine a service that collects fees per use from its consumers. What should the service provider do if it wants to change the service and that change is welcomed (or even required) by all but one of the consumers? If the change is allowed, that one consumer will have to rewrite, retest, and redeploy the application. If the change isn't allowed, the service provider might lose the business of the rest of the consumers.

I believe this risk will prevent companies from adopting the "software as a service" notion until some mechanisms are in place to address this problem.

But you can use Web Services in another way. Imagine a situation where a company wants to access its own application across the Internet at a remote customer site. Such access is useful for support, maintenance, and upgrade purposes. In this scenario, the company controls both the service definition and its use, and is therefore not susceptible to a third party changing the service definition. The company uses Web Services only to access its application remotely through the customer's firewall. Developing such remote-access solutions without Web Services is difficult because of the reasons I discussed already. However, Web Services make developing such solutions easy. I think this is where Web Services will stake their first major claim—in situations where companies hold both ends of the stick.

Web Services have the potential to be the next information technology revolution. They promise to change the nature of our interaction with the Web and the services it provides. .NET and Web Services will also affect the way each and every one of us develops applications, even if only to accommodate exposing the application as a Web Service in the future. But, at least for the short term, Web Services won't be used the way Microsoft hopes. **VSM**

Juval Lowy

Agree? Disagree? State your opinion on the Talk to the Editors discussion area on The Development Exchange (http://news.devx.com).

The opinions expressed in this editorial are those of the author and not necessarily the opinions of *VSM*.

Juval Lowy is a seasoned software architect and the founder of IDesign, a consulting firm focused on COM and .NET design. Juval also conducts training classes and conference talks on component-oriented design and development process, and he authored the book *COM and .NET Component Services* [O'Reilly, August 2001]. Reach him at www.componentware.net.