The Web Services Debate

As the articles in this section attest, the future of Web services is as certain as it is unclear. That is, the Web services arena is most certainly the next technological wave; what is not so clear is what direction (of many) that wave will flow. The challenge of selecting the tools to successfully pull all the components together is particularly daunting.

Two of the leading companies in the high-tech field are competing for that vote with architectures that promise to build and orchestrate distributed applications. There's been much speculation regarding Microsoft's .NET and Sun Microsystems' J2EE approaches to Web services solutions. Here, Joseph Williams, Master Architect, Sun Professional Services, and Gerry Miller, chief technology officer for Microsoft's U.S. Central Region, argue the strengths, advantages, and implications of these systems.

by JOSEPH WILLIAMS

J2EE vs. .NET

The mere utterance of the word "integration" in a meeting of experienced IT professionals can clear the room faster than a Paul Anka song at a college rave. Hardly a *Dilbert* cartoon storyline doesn't at some point mock the integration claims of some weasel of a vendor or of a hapless engineer (or both!). For all the skepticism, derision, fear, and loathing directed at integration as a concept, a widely used rule of thumb is that 60% of IT costs are absorbed by integration efforts. Integration is staggeringly important but remains a numbingly difficult activity.

One of the many very good reasons that Web services have been given a lot of attention is because they have the potential to dramatically mitigate the complexities and the costs of integration projects. Formal analyses of these claims are still in the offing, but anecdotal evidence suggests that significant

Illustration by Jean-François Podevin

In point of fact, the realization of Web services as a common communications infrastructure requires cooperation between the J2EE and .NET factions.

reductions in lead times and investments are being achieved by Web services deployments. The now-famous example of the Southwest Airlines/Dollar-Rent-a-Car Web services integration achieved reduced transaction costs by 80%.[1] Merrill Lynch reported they recently brought in a budgeted $800,000 project for $30,000 when they switched to a Web services solution.[2]

At the bottom line, Web services are of interest in large part because they promise a common communications infrastructure that will be supported by all of the major application vendors and development shops. There are still a number of major shortcomings to Web services that need addressing before they can be anointed as a *linqua franca* for integration (security and workflow management are just two obstacles) and it is not at all clear at this point that SOAP, XML, HTTP, UDDI, and WSDL will be the ultimate enabling platforms for Web services.[3] However, there is considerable momentum behind Web services; although Web services may not yet be a mature technology, they have clearly arrived. Unlike previous attempts at interoperability (for example, DCOM, CORBA), Web services take a loosely coupled approach that is appealing at many levels.

Development and deployment of Web services requires no specific underlying technology platform. However, the 600-pound gorillas on the scene are Microsoft's .NET framework and the Java Community Process' (JCP) J2EE specification. Since Sun Microsystems invented the Java programming language and is one of its most vocal proponents, the Web services platform debate is often viewed as a Sun vs. Microsoft or J2EE vs. .NET confrontation. Indeed, there is some philosophical and business animus but it does not detract from the core commitments both companies (and, indeed, virtually all companies) have to making Web services successful as common communications infrastructure. In point of fact, the realization of Web services as a common communications infrastructure requires cooperation between the J2EE and .NET factions.

.NET is a Microsoft-centric approach to Web services. It runs on a single platform (Windows) but it does natively support multiple languages (although each is a value-added purchase for the .NET development platform and C# is still the language of choice). J2EE, on the other hand, is a platform-independent solution deployed in a single language (Java), although it does have support for other languages. J2EE is the more mature and proven technology. Microsoft has made a massive investment in its .NET framework and, in many respects, has staked the credibility and future of the company on .NET. Both J2EE and .NET present different challenges and opportunities in the application environment.

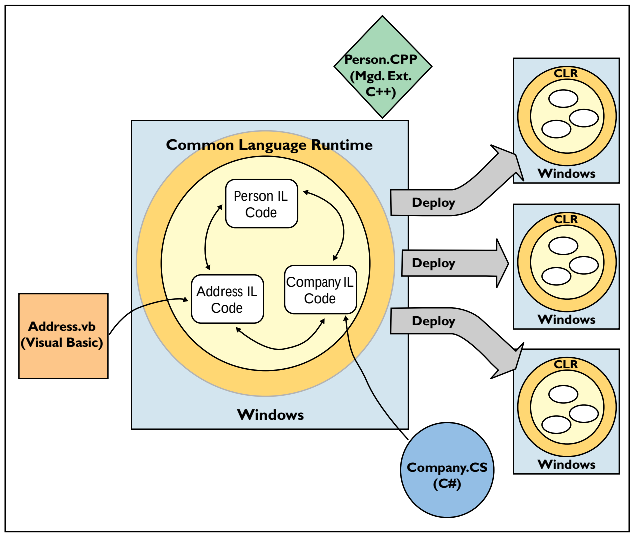

Figure 1. The .NET platform.

The development platform that should be adopted will depend on a number of factors, including the development/business environments, the exigencies being addressed, and the enterprise's objectives for the project. As with any tools choice, J2EE vs. .NET is an "it depends" proposition (see "It Depends" sidebar).

In the final analysis, however, J2EE has much more going for it than does Microsoft's .NET. By most accounts J2EE is the platform of choice for 55–70% of Web services deployments.[4] It has broad support from the major vendors (for example, SAP, Oracle, IBM, HP, PeopleSoft, BEA) and provides the ISVs with the greatest platform flexibility.

[1]www.business2.com/articles/Web/0,1653,35461,FF.html.
[2]Gillmor, S. Show me the model. Infoworld (Dec. 6, 2002).
[3]For a solid introduction to Web services see: java.sun.com/Webservices/docs/1.0/tutorial/doc/IntroWS.html

[4]www.altoWeb.com/events/buzz/zapthinkShowsWS.html; also, www.freeroller.net/page/ceperez/20030118?catname=101%20List

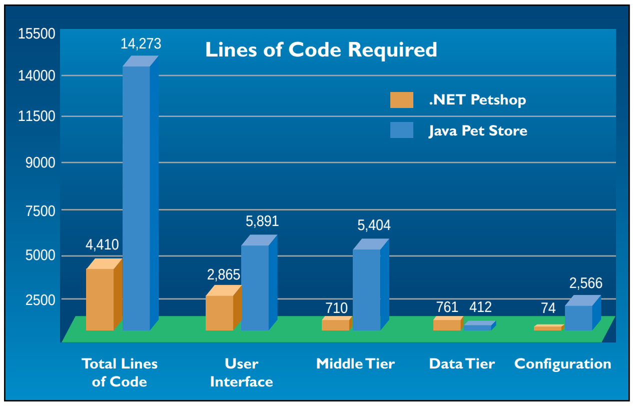

Figure 2. The bogus Java Pet Store metrics.

Microsoft and Sun Microsystems are basically in accord on what Web services are, how they work, and their value to the development community. The issue isn't whether Web services are valuable—we all agree (including IBM, HP, and virtually the rest of the industry) they are. The issue is how to implement them. Thus, it truly is a J2EE vs. .NET debate. And, although Sun Microsystems is the flag bearer for J2EE, it is Microsoft against the entire Java community when it comes to deploying Web services. And Java has a serious head start that all of Microsoft's investment dollars cannot easily overcome.

Microsoft's .NET

.NET is Microsoft's attempt to develop a comprehensive execution environment that includes everything a company would need to implement Web services on Windows. In many ways .NET is Microsoft's Windows-centric response to J2EE. Much of the hype around .NET is just rebranding of existing products. The current centerpiece of .NET is the .NET development platform. A huge product (the zipped file is 1.8GB and expands to more than 24,000 files), Visual Studio .NET is an update to Visual Studio 6 that directly supports the new languages Microsoft is promoting for developing Web services: C#, which is a new language that looks a lot like Java; J#, an even closer implementation of Java syntax and libraries; and VB.NET, a substantial update to Visual Basic.

The bottom line on .NET is that it only runs on Windows servers. Thus, it is completely a Microsoft-centric technology. Yes, it does support dozens of different languages so it is a very rich development platform (of course, each language module must be individually licensed at considerable cost). However, the deployment platform is pure Microsoft, meaning the technology has limited reach in the kinds of heterogeneous environments that persist at most enterprises.

Microsoft has been running a series of attacks on J2EE that feature their .NET rewrite of the Java Pet Store study.[5] Quoting The Middleware Company (TMC) in October 2002, Microsoft stated:

The TMC team spent four months performing this benchmark. They worked 100 hours per week on this and worked every weekend, skipping holidays. Trust me, they were trying very hard to make J2EE win. But in the end, J2EE did not come out on top. TMC had a lot of heartache seeing the results of this benchmark. We internally debated about whether we should post this or not. In the end, we decided to go forward and publish this report.

In fact, TMC is obscure and certainly not a definitive source for information on J2EE (that is, it's all FUD—fear, uncertainty, and doubt). Microsoft will also cite a VeriTest report saying that .NET is 10 times faster than J2EE.[6] This report is also little more than FUD and has been countered effectively by Oracle (otn.oracle.com/tech/java/oc4j/pdf/9ias_net_bench.pdf). And, since the announcement of its results TMC is considering a rematch based on feedback it received on flaws to its approach.[7]

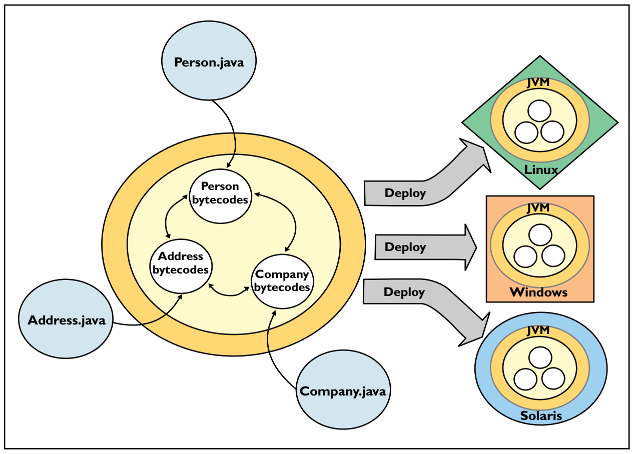

Figure 3. The J2EE platform.

	.NET	J2EE
Programming Language	C#	Java
Interpreted Language	MSIL	Java Bytecode
Runtime Environment	CLR	JVM/JRE
Rich-Client	VB.NET	Swing
Web Presentation	ASP.NET	JSP/Servlets
Business Services	????	EJBs
DB Integration	ADO.NET	EJB-SQL/JDBC
Messaging Integration	MSMQ	Msg EJBs/JMS
Legacy Integration	COM TI	JCA

Figure 4. Feature comparisons.

[5] www.onjava.com/pub/a/onjava/2001/11/28/catfight.html
[6] gotdotnet.com/team/compare/
[7] www.middleware-company.com/j2eedonetbench/

The Java Pet Store was never designed to be a benchmarking platform, as all but Microsoft acknowledge. It isn't even clear yet how to make meaningful performance comparisons between .NET and J2EE that involve large-scale, highly concurrent, high-availability applications. Thus, at this early stage all the Microsoft performance FUD falls under the broad category of "lies, statistics, and benchmarks."

Even so, Microsoft's commitment to .NET should not be underestimated. In addition to rebranding their entire product line as ".NET" they have reportedly invested four years and millions of hours in producing its .NET development environment.[8] They have supposedly invested more than $6 billion in new money already in developing the .NET story (by contrast, Sun Microsystem's total R&D budget is estimated to be about $1.8 billion per year). To spread the .NET gospel, Microsoft and Hewlett-Packard have jointly invested $50 million to train 5,000 HP sales professionals on .NET, to get 3,000 HP Services professionals certified on .NET, and in forming a new group of .NET solution architects in HP's services group.[9]

It Depends

Remember, Web services do not rely on either J2EE or .NET—they ride on SOAP, XML, and other independent components. The J2EE vs. .NET decision is more about in which application development environment Web services will be deployed.

The selection of J2EE or .NET will probably hinge on two or three factors. Although the Java Community does not concede the desktop to Microsoft, analysts have been quick to point out that .NET will probably become the de facto standard for client-side application development because Microsoft is so dominant there. Similarly, J2EE is already entrenched in most enterprise-grade and large-scale Web deployments so J2EE will continue to be the incumbent there. The area of greatest contention will be in the mid-tier application space where J2EE and .NET will contend with C++, Java, and other development environments for enabling Web services.

The deployment platform is also likely to be an important factor. If the application is to be developed on Windows and is to run only in a Windows environment, then .NET will have a certain appeal. For most companies, however, the deployment environment will be heterogeneous and J2EE will have greater appeal because of the extensibility of its write-once/run-anywhere core.

The other issues are likely to be economic and philosophical. An entrenched J2EE or .NET shop would face considerable expense in migrating to the other platform. There are also issues for many companies associated with Microsoft's monopolistic practices and the impacts they are having on licensing costs and product quality. On the other hand, some companies are attracted to Microsoft's solutions (for reasons which they can better articulate) and .NET will be an easy choice for them. Experienced shops will probably find J2EE's product maturity to be compelling.

In shops that run both J2EE and .NET the choice of Web services platform would go through the same process as any other development project, including an analysis of the performance, supportability, and economics of the solution. It's difficult to say what those will be, which is why the tools selection answer is "it depends." C

J2EE

The Web services promise of loosely coupled components existing in a build once/use-many environment finds its natural expression in J2EE. Although J2EE is Java-centric it does have the ability to run on any operating system; thus it represents the flip side of .NET. J2EE is one language, many platforms. .NET is many languages, one platform (although, arguably, that actually isn't true since most of the powerful tools for .NET are designed specifically for C#; thus, it really is one full-featured language on one platform).

The bottom line on J2EE is that on a feature-by-feature basis it compares very favorably with .NET (see Figure 4).

Moreover, all of the specifications that define the J2EE platform are published and reviewed publicly, and numerous vendors offer compliant products and development environments. This simply isn't true with .NET.

J2EE is more mature than .NET and even though XML is not yet native (it will be in JDK 1.5) much of the vision implemented by Web services was presaged in J2EE.

.NET Claims Against Sun and J2EE

The software community expected Sun to step right up and announce a well-articulated vision to counter Microsoft's marketing machine around .NET. For whatever reason, analysts now claim it didn't happen:

"To the disappointment of many, Sun announced only some XML interfaces to Java and little else. No

[8] www.microsoft.com/presspass/press/2002/Apr02/04-10UmbrellaPR.asp
[9] www.Webservices.org/index.php/article/articleview/655/1/10

overarching vision, no clever sound bites. Nothing. As can be seen from the companion interview[10] with Marge Breya, vice-president in charge of Sun ONE, there is still no vision beyond promoting Solaris, the iPlanet application server, and the Liberty Alliance. (DevX Special Report)."[11]

This perception, unfortunately, is widely held and a number of influential commentators have stated that Sun Microsystems missed the Web services wave. This simply isn't true (after all, Sun's own John Bosak was on the forefront of XML's invention), but the misperception lingers. J2EE support for Web services has always been implemented in the J2EE Web Services Pack and, as mentioned earlier, Web services will be native to J2EE in JDK 1.5.

Microsoft claims the default interpretive mode of Java is a liability (it has stated this in response to several lawsuits filed against the company) in that bytecodes designed for a virtual machine do not lend themselves as well to native optimization. There is no hard data to prove or disprove that claim, either generally (bytecodes vs. native-compiled languages) or specifically (Java vs. C#) so this claim is also little more than marketing FUD.

Conclusion

Both Sun and Microsoft are articulating clear visions around Web services. J2EE and .NET are different tools embodying different strategies for implementing Web services. It is pointless to argue currently that one tool is superior to the other. Instead, the focus needs to be on ensuring the right tool is deployed; for most enterprise shops J2EE offers more flexibility, greater robustness, and a proven pedigree. C

[10]archive.devx.com/javaSR/articles/interview/breya-1.asp
[11]archive.devx.com/javaSR/articles/binstock/binstock-2.asp. See also, www.informationweek.com/story/IWK20020208S0037 and www.economist.com/search/search.cfm?cb=46&page=index&qr=Let+Battle+Commence&area=1

Joseph Williams (joseph.williams@sun.com) is Master Architect, Sun Professional Services, Fort Collins, CO.

The opinions expressed within are the author's own and do not represent those of his employer.