WebSphere vs. .Net:
Choosing an e-Business Platform in the Enterprise

by Robert Frances Group, Inc.

October 2004

ROBERT FRANCES GROUP
Business Advisors to IT Executives

Table of Contents

Executive Summary

IT executives today are faced with several choices when re-engineering legacy systems or deploying new applications. Both IBM Corp.'s WebSphere and Microsoft Corp.'s .NET are promising options, but discussions about both environments dissolve into polarized debates that rarely account for enterprise development requirements. In this paper, RFG presents an enterprise-focused comparison between WebSphere and .NET, with particular attention paid to the unique needs of enterprise customers. This comparison is broken down into the following categories:

1. **Strategic Position** – The overall value of each development environment.
2. **Product Acquisition** – Licensing costs and purchase options for the products involved.
3. **Development Cycle** – Architect and developer experiences with each environment.
4. **Application Deployment** – Deploying the stack products and the developed applications.
5. **Management and Support** – Tasks such as patch management and performance monitoring.
6. **Long-term Effects** – The ramifications of each choice over a multi-year ownership period.

Platform Scores

	WS	.NET
Strategic Position	9	4
Acquisition	8	7
Development	7	6
Deployment	8	6
Management	9	6
Long-term Effects	8	5
Totals	**8**	**6**

Overall, RFG found that both products were relatively easy to acquire and deploy, and addressed Web services development requirements. However, this is where the similarities end. .NET is plagued by security management issues, scalability limitations, and an utter lack of hardware architecture, operating system, and vendor portability, all of which increase total cost of ownership (TCO) and decrease enterprise elasticity. RFG also believes mistrust over Microsoft's recent licensing changes and competitive tactics have underscored the weakness of becoming locked into the vendor's product stack and road map.

On the other hand, Java has become one of the most extensively deployed development architectures in the world, and boasts hundreds of sub-technologies that address specific development needs. WebSphere deployments may involve more architectural elements and higher developer salaries than Visual Basic(VB).NET. However, better out-of-the-box security levels, higher functionality, higher server-to-administrator ratios, and more extensive management interfaces reduce long-term ownership costs, and make the platform a much better choice in the long run.

WebSphere also addresses enterprise elasticity because it provides customers with choices should a hardware platform or operating system change be required. Strategic elements such as these can provide long-term benefits that can help a company meet aggressive growth schedules, weather tight economies, and address other business challenges. RFG thus believes WebSphere is generally the best choice for enterprise development projects.

Introduction

IT executives today are faced with several choices when re-engineering legacy systems or deploying new applications. IBM's WebSphere provides a mature and rich target environment for these efforts, and support for the industry-standard Java 2 Enterprise Edition (J2EE). On the other hand, Microsoft's .NET architecture provides tight integration with various Microsoft products and a migration path for VB applications, which are heavily deployed in many enterprises. Unfortunately, discussions about both environments have dissolved into polarized debates that often provide little value to IT executives in this decision-making process.

Enterprise development needs are complex and unique, and cannot be addressed by simple technical feature comparisons. Most companies have a variety of systems, including both legacy and new environments, distributed and vertically scaled systems, and low-end to high-end hardware platforms. Enterprises also tend to deploy more complex applications than smaller businesses, so cost differentials are multiplied by longer ownership periods, and long-term planning horizons are much more critical because selection decisions are not easily changed.

In this paper, RFG presents an enterprise-focused comparison between WebSphere and .NET, with particular attention paid to the unique needs of enterprise customers. This comparison is broken down into the categories listed below. In each category, RFG examined a variety of characteristics and scored each on a scale from 1 (worst) to 10 (best). Characteristic scores were averaged to produce an overall score for each category.

1. **Strategic Position** – The overall value of each development environment.
2. **Product Acquisition** – Licensing costs and purchase options for the products involved.
3. **Development Cycle** – Architect and developer experiences with each environment.
4. **Application Deployment** – Administrator experiences in deploying the software products in each stack, as well as the applications created by the architects and developers.
5. **Management and Support** – Management of each platform, patch deployment, and other support tasks.
6. **Long-term Effects** – The ramifications each choice would have based on a multi-year ownership period.

Both WebSphere and .NET may be deployed in a variety of product combinations, and certain comparisons such as licensing costs thus require specific product selections. Where this occurs, the Microsoft stack will consist of Windows Server 2003 and SQL Server 2000. The IBM stack will consist of WebSphere Application Server and DB2 Universal Database (UDB). Where specific product versions are referenced, these will be described in further detail.

Strategic Position

Both WebSphere and .NET strive to provide similar functionality, but application development tasks cannot be accomplished equally in either environment. The differences begin at a strategic level, and in this section, RFG examined the maturity of each platform, availability and quality of developer skill sets, portability issues, and adherence to industry standards.

Maturity

J2EE has a head start on .NET in terms of maturity. Sun Microsystems, Inc. released the initial specifications for the Java development language in 1995, and it has since become a standard choice for new development projects. In 1999, Sun created J2EE, a framework for developing Java-based, multi-tiered enterprise applications, and within a year, BEA Systems, Inc., IBM, Sun, and other vendors announced application servers that supported the standard. These vendors and other companies also participate in a community review process for changes to the Java architecture and language. There are now dozens of cooperatively developed J2EE sub-technologies that provide specific functions such as directory services integration, all of which are based on industry standards.

In contrast, .NET began with a late and confusing start when Microsoft announced the .NET platform in 2000. The announcement was followed by significant confusion as, over the next three years, Microsoft redefined the goals and product compositions for its .NET strategy. These redefinitions were in part driven by the market's refusal to accept certain key components in the original strategy, such as Hailstorm. Hailstorm not only raised security and privacy concerns, but also raised criticism that Microsoft was setting itself up to collect tolls on Internet transactions.

In contrast to J2EE, .NET is comprised of several languages, including the traditional Visual Basic and Visual C++. Born out of the bitter war with Sun over claims of "extend and embrace" tactics to create an incompatible version of Java, Microsoft added a new language to its development suite called C#. With similar object structures and hierarchies, C# was billed as the next language evolution for Visual C++ developers, but is in fact Microsoft's object-oriented alternative to Java.

Microsoft has traditionally catered to the small and medium-sized business (SMB) market, and the extensibility of its product offerings serve as a testament to this. Despite inroads in technologies, such as component object model (COM) and distributed component object model (DCOM), which aim to address n-tier architectures, the development platform remains immature and not as well suited to enterprise development as J2EE.

For example, VB.NET is arguably one of the more mature Microsoft languages, but is not structured to handle enterprise application development. C# lacks the maturity of VB, but offers no strategic advantage over Java. A decision to employ C# as the basis for an enterprise application inherently locks in a reliance on the Windows platform, and denies portability to alternative operating systems such as Linux.

Although Mono and the DotGNU Project provide an Open Source C# interpreter, they lack the commercial interest to give them enough momentum to significantly penetrate the enterprise, and neither is endorsed or supported by Microsoft. Furthermore, the development tools available for C# or VB are limited to what Microsoft provides, unlike Java, which has a far greater number of options.

Stability

As previously discussed, WebSphere is designed to run J2EE applications, and J2EE is a mature and entrenched standard created and accepted by a community of participants rather than a single entity. Changes to the architecture tend to be evolutionary, such as the recent introduction of new standards to define application workflows. In fact, J2EE's road map predates the advent of Web services, and while a significant number of new APIs and standards were developed to support them, J2EE itself did not

significantly change to support them. To the contrary, J2EE's object-oriented nature lends itself well to Web services development.

On the other hand, .NET developers have already seen significant shifts in their APIs that have required redevelopment. When VB.NET was released to replace VB version 6, developers were forced to redevelop many applications before they could take advantage of the new environment. Microsoft also recently announced the planned discontinuation of Web Storage System (WSS) and WinForms, APIs that were introduced with Exchange 2000 and Windows .NET, respectively. Further, for Web services applications written in .NET, the code will need to be rewritten if customers want to take advantage of the new features of Longhorn when it arrives. These moves leave customers wondering which APIs to use, and may force redevelopment efforts, as customers attempt to keep up with Microsoft's new products.

Developer Skills

This maturity also extends to the availability of developers. Employers generally have little difficulty filling Java and VB.NET positions, while C# skills are less common and can also be more expensive as a result. Because an object-oriented language would often be the preferred choice for enterprise application development efforts, this conflict makes .NET somewhat less suitable than J2EE. Further, because J2EE is more mature than .NET, it is possible to find developers with several years more experience writing J2EE applications, and thus increase the likelihood of success of a J2EE development project.

Traditionally, VB has been used to create user interfaces and front-end databases. Adding and communicating with the middle logic layer for enterprise applications is a task that VB developers are not often asked to perform, and thus may not be prepared to address.

Vendor Portability

J2EE application servers and middleware products may be obtained from a number of vendors, including BEA, IBM, Oracle Corp., SAP AG, and Sun. There are also several Open Source options available. Competition among these vendors drives innovation, and has generated an array of development and platform products from which customers can choose. Customers may thus choose vendors and products based on business requirements or value received, such as consulting offerings, pricing, and support channels. Should it become necessary to select a new vendor, a customer would have the option of moving to a competing product, in many cases with little to no redevelopment involved.

Microsoft has a single-vendor philosophy wherein Microsoft is the primary provider of all core technologies and development tools. The vendor's only concession is in the area of third-party, add-on products, such as universal modeling language (UML) tools. There are no alternative vendors for key .NET components, including BizTalk Server, Windows Server, and Visual Studio.

There are advantages and disadvantages to a company relying on technologies driven by a single vendor for innovation. Speed of changes to market is perhaps the greatest advantage. However, it can also be a huge challenge. If a vendor comes out with programming language innovations or significant upgrades every two years, the market may not be prepared to incorporate them at the same rate. Many times, such upgrades can require extant code to be rewritten extensively. Further, it makes the decision of which version of a technology to use for development projects almost impossible for development projects of any length. With Open Source, technologies benefit from a greater quantity of community input, and therefore have a greater continuity of structure and approach.

Java has the added benefit of the competition between the J2EE vendors, who also have to answer to the needs of customers. Microsoft does not have the benefit of competing with another vendor, and instead has only to respond to the needs of its customer base. As a result, users rely solely on Microsoft for innovation, and thus cannot benefit from the choice and innovation resulting from competition.

Platform Portability

Hardware architecture and platform portability have high strategic value in enterprise environments because they allow customers to run each application on the most appropriate platform combination. It is common to perform development tasks in a shared server environment, deploy pilot applications on low-end hardware, then scale vertically or horizontally as application workloads increase. Availability, performance, and scalability requirements also factor into the decision-making process.

IBM's WebSphere is the most extensively ported application server available. It runs on a wide range of hardware architectures, including Hewlett Packard Co. (HP) RISC, IBM POWER, IBM zSeries, Intel Corp. 32-bit, Intel 64-bit, and Sun SPARC, on AIX, Linux, Netware, OS/400, Solaris, Windows, and zOS operating systems. It can also run on embedded systems, such as cellular telephones or other mobile devices.

.NET offers very limited platform portability. As mentioned above, DotGNU and Mono both offer limited .NET support on other platforms, but neither of these projects is certified, endorsed, or supported by Microsoft. Microsoft itself only offers .NET on Windows, and Windows is only available for Intel-based systems. With the introduction of Itanium, Windows customers gained a 64-bit target platform, so customers can now address scalability requirements to a point. However, as long as .NET remains tied to Windows it will continue to suffer from the limitations inherent in a single-architecture strategy.

Standards Support

Finally, both WebSphere and .NET have a heritage of support for industry standards, including data definitions, language specifications, and protocol formats. WebSphere currently supports a larger number of standards, and there are early indications that Java itself may be Open-Sourced at some point, a move that would benefit all users.

Strategic Position

	WS	.NET
Maturity	8	5
Stability	9	4
Developer Skills	7	5
Vendor Portability	9	3
Platform Portability	10	2
Standards Support	9	5
Final Score	**9**	**4**

.NET also has a basis in standards, and Microsoft's new C# language has been approved by Ecma International, an industry standards organization. However, the rest of .NET is not standardized, and Microsoft has a track record of locking customers into its higher-level platform and technologies. Many of its products continue to show the earmarks of the vendor's "extend and embrace" strategies. Microsoft also has a history of creating closed, proprietary protocols and APIs that make it more difficult for third parties to integrate with Microsoft products, especially from other platforms.

The Bottom Line

From a strategic planning perspective, WebSphere offers a significantly more attractive platform for enterprise application development efforts. IT executives should take care when hiring developers to look for J2EE experience in addition to pure Java knowledge, and should add their voices to encouraging Sun to completely Open Source Java.

However, .NET poses significant challenges to enterprise development efforts. Language choices do not meet all mission-critical application requirements, product lock-in makes it difficult and costly to migrate to alternatives, and a lack of hardware architecture and platform portability prevent customers from managing and scaling workloads as required.

Product Acquisition

Development projects require a range of support products in two key areas. The first set includes the application infrastructure – the servers, operating systems, and other software products required to deploy a WebSphere or .NET application. The second category is the set of development, modeling, and testing tools used to create the business applications. Costs for products in the second category will be evaluated in the Development Cycle section below.

To calculate software purchase costs, RFG considered four hypothetical customer environments running e-commerce applications. The first two environments simulate a "basic" deployment to handle low-end application workloads and include a single server each. The first system is configured with Red Hat, Inc. Enterprise Linux ES Basic plus WebSphere Application Server Express, and the second system is configured with Windows Server 2003 Web Edition plus SQL Server 2000 Standard Edition.

The third and fourth environments are intended for mission-critical, high-volume workloads. These workloads would be deployed in a multi-tiered environment. In this case a three-tier environment was selected for comparison, consisting of database, business logic, and presentation layers, for a total of three servers.

The third environment thus includes three copies of Red Hat Enterprise Linux ES Standard, a copy of WebSphere Application Server, and a copy of DB2 UDB Workgroup Server Unlimited Edition. The fourth environment includes three copies of Windows Server 2003 Standard Edition and one copy of SQL Server 2000 Enterprise Edition. The reader should note that Microsoft does not offer a mid-level database product that directly competes with DB2's Workgroup Edition. Customers that require more functionality than SQL Server 2000 Standard Edition must purchase Enterprise Edition.

All prices are retail and are based on per-processor licensing. Web server pricing is not included because Microsoft's Internet Information Server (IIS) is included for free with Windows and Apache is Open Source.

Purchase Cost Summary **"Low End"**

Component	Red Hat and WebSphere Express	Windows 2003 and SQL Server
Operating System	$249/year	$399
Application Server	$2,000	Included
Database	$624	$4,999
Three-Year Total	$3,671	$5,398

Purchase Cost Summary "High End"

Component	Red Hat and WebSphere	Windows 2003 and SQL Enterprise
Operating System	$2,397/year	$3,597
Application Server	$10,000	Included
Database	$9,375	$19,999
Three-Year Total	$26,566	$23,595

Source: Robert Frances Group

Microsoft products are often thought of as less expensive up front, especially at smaller scales, but RFG found this assumption to be untrue; a WebSphere deployment was 32 percent cheaper. At this scale, the high cost of SQL Server more than offsets the additional application server product in the WebSphere case.

At the high end, .NET is nominally less expensive than WebSphere, although this pricing fails to take into account the increased functionality provided by WebSphere, including enhanced service monitoring, service level controls, and easier end-to-end performance management. It is difficult to normalize the value of these features for each customer, but the fact remains that additional value is received as part of WebSphere.

Purchase Costs		
	WS	.NET
Options Available	8	7
"Low End" Costs	8	6
"High End" Costs	7	8
Final Score	**8**	**7**

The Bottom Line

Overall, WebSphere is less expensive than .NET at the low end and only marginally more expensive at the high end, while offering more flexibility and functionality. However, RFG believes software acquisition costs constitute only a small percentage of TCO for most application deployments, and IT executives should ultimately focus on strategic factors, human costs, and long-term support costs rather than up-front expenses.

Development Cycle

The development cycle includes both development costs and qualitative differences between J2EE and .NET. RFG examined developer skill sets and salaries, tool costs, and the relative complexity of several key development tasks.

Development Costs

To estimate development costs, RFG reused the hypothetical target environments given above. Attempting to determine actual headcount and time requirements produces heated arguments among

WebSphere and .NET supporters, and RFG believes application engineering issues dwarf cost savings and timesaving claims based on features or functionality in each environment.

RFG thus based these estimates on simplified cases and salary data collected during past interactions with RFG clients. The salaries used were $71,000 for Java developers, $67,000 for .NET developers, $78,000 for Java architects, and $75,000 for .NET architects. Database administrator salaries were based on $67,000 for DB2 and $62,000 for SQL Server. These salaries were each increased by 30 percent to cover employee overhead.

The "Low End" environments will thus be compared based on two developers and a development period of two months. At this time, WebSphere customers have more development tool options than those using .NET. WebSphere Application Server Express includes development tools for each license. A licensed copy must be obtained for the second developer. VB.NET Professional will be installed on the .NET developers' systems.

The "High End" environments are based on three developers and six months of development time. Further, these environments will include one architect as a project leader, who will be supplied with the appropriate Enterprise version of WebSphere Studio or Visual Studio. Finally, the "High End" environments include two weeks of database administrator time to address data migration requirements from a hypothetical legacy system.

Development Cost Summary

Component	"Low End" WebSphere	"Low End" .NET	"High End" WebSphere	"High End" .NET
Developers	$30,767	$29,033	$138,450	$130,650
Architect	N/A	N/A	$50,700	$48,750
DBA	N/A	N/A	$44,200	$40,300
Development Tools	$100	$2,158	$1,000	$3,237
Architect's Tools	N/A	N/A	$5,500	$1,799
Migration Costs	N/A	N/A	$3,630	$3,196
Totals	$30,867	$31,191	$243,480	$227,932

Source: Robert Frances Group

Development costs are thus relatively similar between WebSphere and .NET, with higher Java salaries contributing to about a six-percent increase in the "High End" environments. As with the product purchase costs, this comparison is a simplification that does not take into account factors, such as language differences, time reducing code generation tools, and wizards. In the case of J2EE, such timesaving factors could result in lower implementation costs in the end.

Skill Sets

The calculations above are based on VB developers. While using C# developers in the above scenario would even out personnel costs, as they often command higher salaries than J2EE developers, they are harder to find. It could be pointed out that this choice serves as a point in favor of J2EE because there are developers available to develop enterprise applications.

IT executives who choose WebSphere need only seek personnel with J2EE skills, while those considering .NET projects must either choose between VB.NET and C#, or attempt to employ individuals with both skill sets. Otherwise, Java and VB.NET skills are both readily available, though the pool of VB skills tends to occupy presentation layer development predominantly. IT executives should simply be careful in both environments to seek out individuals with J2EE and .NET framework skills, rather than just experience with the underlying language.

IT executives may find that filling the enterprise development positions is easier when using J2EE, since there are currently more skilled enterprise Java programmers than enterprise VB or C# programmers.

Task Complexity

It is difficult to include task complexities in development cost estimates because not all applications are affected to the same degree by each issue. However, qualitative differences do exist between J2EE and .NET, and RFG examined these differences in each of the following areas:

1. Creating a traditional multi-tiered application
2. Creating a Web service
3. Testing component behavior and benchmarking performance
4. Performance
5. Developing secure code
6. Porting applications to alternate architectures and platforms

Multi-tiered Applications

Creating multi-tiered applications is complex in both environments, but both Microsoft and IBM provide reference applications and sample code that developers can use as a basis for their own projects. The main delta here is in performance monitoring and management to meet service level requirements. WebSphere includes specific functionality to provide visibility into the operational characteristics of J2EE Enterprise JavaBean (EJB) objects on a server as well as hooks for external monitoring tools. Although Windows provides some of the same functionality, its features are not as extensive. WebSphere developers and architects can thus more readily obtain an end-to-end view of performance characteristics, making it easier to identify bottlenecks and meet service level agreements (SLAs).

With the advent of .NET, Microsoft added a new means for communicating between tiers of an enterprise application. .NET now permits and encourages the use of .NET Remoting, instead of DCOM. .NET Remoting uses channels over which messages pass between the different layers. Developers have the option of using HTTP or Transmission Control Protocol (TCP) as a choice for communicating over channels. With HTTP, user context information such as security identification and role membership can be passed between tiers automatically, however HTTP is the slowest of the possible choices.

Alternatively, developers can opt for using the TCP channel for .NET Remoting, which when compared to the HTTP protocol, is substantially faster. However, it has no means by which to automatically pass

user context information, relying instead on developers to write the necessary context code. Furthermore, there is no native support for managing transactions. To achieve this, it is possible to fall back to DCOM/COM+, and doing so would likely provide faster throughput than .NET Remoting.

To enable .NET code to interact with DCOM/COM+ requires extra layers of code "interop wrappers" for translation, thereby slowing communication. However the wrappers are necessary in order to take advantage of object pooling and participation in automatic or distributed transactions.

J2EE takes advantage of a communication protocol called Remote Method Invocation over Internet Inter-ORB Protocol (RMI/IIOP). By communicating over the standard Transmission Control Protocol/Internet Protocol (TCP/IP), J2EE can make calls to application servers. The communication protocol provides for authentication, location services via Java Naming and Directory Interface (JNDI), and object transfer from client to server. J2EE has an advantage over .NET in providing the services of authentication, object pooling, and transactions because it does not have to go across the extra "wrapper" layers that are required to translate between .NET and COM.

Web Services

Web services development benefits and drawbacks are hotly contested by J2EE and .NET supporters, and RFG believes many of the arguments from both communities have little bearing on enterprise development efforts. The real problem with Web services is that many of the specifications on which they are based have only recently been standardized. Thought leaders in the space are still discovering both best practices and opportunities for new development concepts based on these standards, such as grid computing and server-managed desktop applications. Thus, while creating Web services is relatively straightforward in both environments, it can also be difficult for IT executives to determine when and where Web services should replace traditional applications.

Testing

Testing processes follow traditional testing and debugging methodologies in .NET, but WebSphere Studio generates testing code for developers automatically, making this a much easier process in J2EE. Further, IBM's recent acquisition of Rational Software gives customers a single-source vendor for both an application server and higher-level development and testing tools.

Performance

The available third-party studies on the performance differences between J2EE and .NET often dissolve into polarizing debates pitting vendors against one another. However, due to the advantages outlined above in the multi-tiered application section, RFG believes that J2EE will perform better in disparate enterprise environments.

Developing Secure Code

There are two distinct concerns when security in developed products is considered. The first is in the platforms and products with which applications are created and on which applications are deployed. Microsoft has a very poor track record in platform-level security compared to IBM's WebSphere, and although the vendor has made some progress in the past year, the continued high rate of vulnerability discovery and patch distribution is disappointing. Further, Microsoft as an entity has sufficiently irritated the creators of vulnerability exploits that the bulk of these attacks are generally targeted at Microsoft's

products. WebSphere is not immune to occasional vulnerability discoveries, but the rate at which these are discovered is far lower. Moreover, most patches do not require a server reboot to complete installation, and these vulnerabilities almost never become attack vectors for malicious agents.

Second, developers face the task of creating secure applications; that is, the code they write may also be subverted to malicious ends if not properly written. In practice, developers require sufficient education in appropriate measures to avoid creating vulnerabilities, regardless of which development environment they use. However, WebSphere also provides security elements that control access to and the behavior of components without requiring additional effort by developers, so it is easier to provide for authentication, authorization, and auditing in WebSphere.

Beyond inherent security aspects, both J2EE and .NET have the ability to incorporate security into the code. To protect resources that are external to the Java virtual machine, J2EE can make use of the classes from the SecurityManager class. Doing so provides a mechanism for each unsafe action to require permission before proceeding. .NET also has a class named SecurityManager, which implements the similar functionality of role-based permission checking.

Task Complexities

	WS	.NET
Multi-tier	8	6
Web services	6	6
Testing	9	6
Performance	8	7
Security	8	5
Porting	8	1

VB developers are not accustomed to writing code for securing access to applications, as there were no means to do so prior to the .NET development platform. Traditionally, securing applications has been the responsibility of the network administrator, who would apply permissions to directories and files. Therefore, as stated above, the shift to .NET requires the additional security consideration and education of developers using Microsoft languages.

Legacy Enablement

For applications that need connections to legacy applications residing on mainframes, J2EE has a distinct advantage over .NET. For any workable degree of programmatic access to mainframes, .NET requires the existence of an additional product called the Host Integration Server (HIS). For an additional $2,600, calls to databases and files on a mainframe can be facilitated through HIS. Unfortunately, even doing this goes against a design goal of .NET in that HIS provides mainframe access by wrapping it in COM objects. Furthermore, HIS relies on IBM's distributed data management architecture (DDM) to communicate with mainframes.

Development Cycle

	WS	.NET
"Low End" Costs	7	7
"High End" Costs	6	6
Skills Availability	8	7
Task Complexities	8	5
Final Score	**7**	**6**

J2EE provides a native method for interacting with mainframes though J2EE Connection Architecture (CA). J2EE CA is a standard architecture for connecting the J2EE platform to different enterprise information systems (EIS), such as Oracle or SAP. Each EIS vendor provides a resource adapter that plugs into an application server, and provides communication support for the application server, the EIS, and the applications that interface it.

Cross Vendor Porting
Porting applications between different vendors' J2EE environments is straightforward. Because J2EE is a component-oriented architecture, legacy applications can be wrapped in J2EE components that allow Web services and other new applications to communicate with them. By wrapping legacy applications, the time and money of a complete rewrite is saved. However, as development tools vendors continue to encourage customers to migrate their application development cycles to service-oriented architectures (SOAs), it may not always be possible to componentize legacy applications. In this case, they will need to be rewritten before the new architecture will yield its greatest benefits.

The same component-architecture comments apply to .NET, but where migrating from the products of one J2EE vendor to another's is straightforward, there is no migration path for .NET customers. At some point, Mono and/or DotGNU may be feasible alternatives, but portability is a weak area for .NET at this time, and RFG believes there will always be a demand for commercially supported products in mission-critical environments.

The Bottom Line
Development costs are similar in both environments, although they continue to dwarf both purchase and support costs. .NET has a slight cost lead in "High End" environments due to lower average developer salaries, but J2EE makes up for hourly costs in enterprise tools capabilities. Further, J2EE is a better choice in terms of skills set availability because IT executives need only seek one skills set, and those skills are ubiquitous. Finally, both environments present technical challenges for development tasks such as developing secure code, but J2EE receives a slight edge because .NET is hampered by a nearly total lack of product and vendor portability.

Application Deployment

Deploying .NET itself is theoretically easier than J2EE because fewer products are involved – the core functionality behind .NET applications is now built into Windows, so most deployments only require a single product. In contrast, J2EE stacks require two products, an operating system and an application server. Both environments may also include additional middleware components, such as database and directory servers, eXtensible Markup Language (XML) transaction engines, message queues, and workflow engines.

Application Deployment		
	WS	.NET
Deploying Stack	7	8
Securing Stack	9	3
Deploying Apps	8	8
Final Score	**8**	**6**

For deployment of a .NET application, the executable and associated components can be compiled into an .MSI package, and published or assigned to the relevant client machines or users via Active Directory Group Policy. All patch deployments and uninstalls can also be handled in a similar fashion. Application deployment for Java is most often accomplished through Sun's Java Web Start, a free product that allows the download of an application to clients. By relying on Java Network Launching Protocol (JNLP), Web Start will keep track of versioning, and regularly checks for updates to be downloaded automatically.

Provided that appropriate tools are used, deploying business applications and components is relatively straightforward in both WebSphere and .NET. Eclipse, Visual Studio, and WebSphere Studio all include plug-ins and functions that allow developers to directly export objects to servers, or they may be routed through change management and revision control systems and deployed following a functionality review.

Security

In practice, however, security differences often significantly increase administrator "touch" time with .NET deployments. A WebSphere deployment on a Linux server generally requires little administrator overhead to secure, there are fewer security patches to apply, and most security patches do not require system reboots. With .NET, Windows security patches are released more frequently and most require system reboots, further increasing administrator involvement and adding an additional element of system downtime.

Further, RFG believes certain recent security "enhancements" from Microsoft give administrators a false sense of security, and lead to lower overall security postures in .NET deployments. For example, Microsoft recently introduced the concept of "server roles" in Windows Server 2003, and changed the default setting to disable all roles, making Windows deployments appear as secure by default as Unix systems. However, even with no roles assigned, a Windows server still provides many core networking services, including Remote Procedure Call (RPC), for which several "Critical" level vulnerabilities have been discovered, and in some cases, exploited.

Much of the disparity in security between WebSphere and .NET has less to do with product vulnerabilities than with Microsoft's general goal of hiding system-level functions behind graphical administrative interfaces, and in a poorly documented binary "registry." Windows administrators often do not understand clearly how their systems work on a fundamental level to the same degree as Unix administrators. This makes it much more difficult for them to be proactive regarding security by adding their own judgments to Microsoft's own lock-down recommendations, or in responding to new and unexpected forms of attack.

The Bottom Line

Both J2EE and .NET are relatively easy to deploy, with .NET receiving a slight advantage because only a single product is involved. Deploying applications is straightforward in both environments. However, J2EE is significantly easier to secure than .NET, especially because customers may choose an underlying platform that is more secure by default than Windows.

Management and Support

Management costs are impacted by the underlying platform choice. To simplify the comparison, RFG used data collected in previous TCO studies to determine platform support, administrative, and security costs. Vendor support offerings are retail options for "standard" or equivalent support services; in the case of .NET, this includes Software Assurance. IBM and Red Hat include support as part of their acquisition prices.

Management and Support Cost Summary

Component	"Low End" WebSphere	"Low End" .NET	"High End" WebSphere	"High End" .NET
Vendor Support	$612 (Avg)	$1,350	$4,428 (Avg)	$5,899
Management	$1,295	$1,677	$3,885	$5,031
Security Costs	$867	$2,310	$2,601	$6,930
Three-Year Totals	$8,322	$16,011	$33,052	$53,580

Source: Robert Frances Group

There is a significant delta in support costs between WebSphere and .NET for three reasons. First, the hypothetical cases place WebSphere on Linux servers, and RFG has found that Linux environments typically provide higher server-to-administrator ratios than Windows. Second, patch release rates and deployment issues make security administration a time-consuming task on Windows systems, in turn driving up security management costs. Third, Unix systems offer more opportunities for administrative and systems cost savings through server consolidation by supporting architectures designed for these efforts, such as mainframes and high-end RISC systems.

Management and Support		
	WS	**.NET**
Maintenance	8	6
Security	9	4
Management	9	8
Final Score	**9**	**6**

Finally, Software Assurance makes up nearly 25 percent of the .NET support costs, highlighting the impact this new licensing program has had on customers since its introduction. Customers that choose not to adopt Software Assurance have also seen cost increases, because Microsoft eliminated all other upgrade pricing vehicles in an attempt to encourage customers to adopt Software Assurance.

Both environments provide basic management facilities in the form of console- and Web-based administrative interfaces, as well as Simple Network Management Protocol (SNMP) hooks that may be used to integrate third-party management tools. WebSphere's interfaces are somewhat more sophisticated, reflecting the product's heritage – mainframes have long been very tightly controlled environments, and IBM has a long history as a mainframe vendor.

The Bottom Line

WebSphere is clearly significantly less expensive to operate than .NET, especially when deployed on Linux systems. RFG believes future security efforts by Microsoft will help close this gap, but IT executives should continue to push the vendor to improve its patch deployment services, and to question the value Software Assurance provides. Although paying retail pricing for future product releases is maddening, it may still be less expensive in the long run.

Long-Term Effects

Development environment choices have long-term effects on enterprise elasticity, which RFG defines as "the ability of the infrastructure to bend, contract, and expand without breaking, in timely, agile response to changing business conditions." As mentioned above, .NET offers poor hardware architecture and operating system portability. Customers that develop .NET applications generally become "locked into" the decision because changing any of the three elements would otherwise require a costly and time-consuming redevelopment process. In contrast, WebSphere is available for a variety of operating system and hardware platform combinations, which gives IT executives much more flexibility when future architecture planning decisions must be made.

Long-Term Effects	WS	.NET
Platform Lock-In	9	1
Vendor Trust	7	3
Vendor Stability	9	9
Vision/Road Maps	8	6
Final Score	**8**	**5**

Vendor trust is a related issue. Microsoft's recently introduced Software Assurance program angered many customers when Microsoft attempted to force them to adopt it by eliminating all other upgrade pricing options. No commercial software vendor is immune to criticism by paying customers, especially when IT budgets are shrinking rather than growing. While WebSphere is also a commercial product, IBM has successfully avoided the ire of customers by introducing beneficial programs such as "Express" product offerings rather than attempting to force them to adopt subscription models. In this, IBM has clearly learned from its mainframe experience, while Microsoft is showing its immaturity in the enterprise market.

Platform Scores	WS	.NET
Strategic Position	9	4
Acquisition	8	7
Development	7	6
Deployment	8	6
Management	9	6
Long-term Effects	8	5
Totals	**8**	**6**

Fortunately, IT executives generally do not need to worry about vendor stability with either environment. Even if IBM's or Microsoft's cash flow ceased this year, both vendors have enough operating capital to continue operating for years to come. More importantly, these vendors are dedicated to their strategies. It is highly unlikely that .NET will cease to exist within the next five years, although numerous changes are expected to the core APIs. And, because J2EE is a standard supported by numerous vendors, it is virtually impossible that J2EE will cease to exist in the same time frame.

Finally, both vendors have long-term visions for their products, but .NET is based on revolutionary changes that may require significant redevelopment efforts as new .NET versions are announced. In contrast, J2EE has evolved in manageable steps to support Web services and other new concepts without forcing developers to rewrite their applications. Thus, while Microsoft is the only vendor driving .NET, and can thus implement changes more quickly than J2EE vendors who work collaboratively on new J2EE standards, this has ultimately produced drawbacks rather than benefits

for enterprise customers. The resulting redevelopment efforts lead to significantly higher software development and maintenance costs.

The Bottom Line

While both WebSphere and .NET are available from stable vendors with long-term vision, .NET offers significantly higher business risk due to platform lock-in and vendor trust issues. For dedicated Microsoft customers this point may be moot, especially where those companies are further leveraging technologies already deployed. However, in general, RFG believes WebSphere is a better long-term choice because it provides the most flexibility in future choices, keeping the IT department, and thus the enterprise, elastic.

Conclusion

Overall, RFG found that both WebSphere and .NET offered significant value in enterprise environments, whether used for new applications or to re-engineer legacy deployments. Both environments are also relatively easy to acquire and deploy, and although Microsoft was late to the game with .NET, both vendors have moved rapidly to address Web services development tasks.

However, this is where the similarities end. .NET is plagued by security management issues, scalability limitations, and an utter lack of hardware architecture, operating system, and vendor portability, all of which increase TCO costs and decrease enterprise elasticity. RFG also believes mistrust over the vendor's recent licensing changes and competitive tactics have underscored the weakness of becoming locked into the vendor's product stack and road map.

On the other hand, Java has become one of the most extensively deployed development architectures in the world, and boasts hundreds of sub-technologies that address specific development needs. WebSphere deployments may involve more architectural elements and higher developer salaries than VB.NET. However, better out-of-the-box security levels, higher functionality, higher server-to-administrator ratios, and more extensive management interfaces reduce long-term ownership costs, and make the platform a much better choice in the long run.

WebSphere also addresses enterprise elasticity because it provides customers with choices should a hardware platform or operating system change be required. Strategic elements such as these can provide long-term benefits that can help a company meet aggressive growth schedules, weather tight economies, and address other business challenges. RFG thus believes WebSphere is generally the best choice for enterprise development projects.

This report was developed by Robert Frances Group, Inc. with IBM assistance and funding. This report may utilize information, including publicly available data, provided by various companies and sources, including IBM. The opinions in this document are those of RFG, and do not necessarily reflect IBM's position.