

**Note from The Middleware Company:** The following was as emailed as a Word document to The Middleware Company on our research@MiddlewareRESEARCH.com alias on October 21st, 2004 by an individual that will remain anonymous. It is attributed to IBM but carries no copyrights in it. Since then, other publications such as Syscon Publications Inc. have ran stories about it claiming the document is original and was 'leaked' from IBM.

**IBM Response**

**to the study entitled**

**“Comparing Microsoft .NET and IBM  
WebSphere/J2EE”**

**by The Middleware Company**

*October 2004  
IBM Competitive Technology Lab*

## Another Flawed Study

The latest Middleware Company study is flawed and does not accurately reflect the capability of WebSphere J2EE vs. Microsoft .NET. Like two previous discredited Middleware Company studies, this study was funded by Microsoft. While expert Microsoft programmers were allowed to contribute to the .NET side, neither IBM nor other WebSphere J2EE product experts were invited to contribute to the testing.

The Middleware Company publicly retracted the results of its previous J2EE vs. .NET study because of serious errors in its testing methodologies and its failure to invite J2EE vendors to participate.<sup>1</sup> This study has similar problems. The products and toolsets compared in the testing were mismatched, and relatively crude WebSphere J2EE programming techniques were used rather than the most optimal tools and programming methodologies. Statements made within the study show that the choice of tools and methodologies were purposeful.

The cost comparison between IBM and Microsoft also is inaccurate and does not represent a real-world environment. In addition, there were important cost considerations missing on the Microsoft side.

The Middleware Company's WebSphere J2EE programmers failed to use development practices that would have yielded more favorable results for WebSphere J2EE—for example, practices that would enable software reuse across enterprise architectures. The study also did not attempt to measure the advantages of team development across a business with multiple team members involved. Rather, the study timed the work of three relatively inexperienced WebSphere developers. The results are misleading in each of the categories the tests aimed to measure--Development Productivity, Tuning Productivity, Performance, Manageability, and Reliability.

In its apology and retraction letter for its last J2EE vs. .NET test, The Middleware Company wrote: "We also admit to having made an error in process judgment by inviting Microsoft to participate in this benchmark, but not inviting the J2EE vendors to participate in this benchmark."<sup>2</sup>

Inviting an expert Microsoft partner to participate while not inviting IBM or an IBM partner to participate was once again a fatal flaw. Using outside experts for the Microsoft .NET programming team while using its own employees for the WebSphere J2EE programming team is an obvious conflict of interest in a study funded by Microsoft.

We believe that if J2EE development best practices had been used, the developer productivity would have exceeded that of Visual Studio.NET. The performance and manageability of the WebSphere J2EE application also would have exceeded the published numbers.

Microsoft's track record on these types of studies is not good, and users should generally approach them with caution. Besides two previous Middleware Company studies that were retracted,<sup>3</sup> Forrester Research issued a disclaimer of its Microsoft-sponsored study on the cost of ownership of Linux vs. Windows,<sup>4</sup> and a similar IDC Linux/Windows study was discredited in *Business Week* by one of its authors.<sup>5</sup>

A closer look at how the tests were conducted and the results framed reveals why The Middleware Company study is without merit.

## Wrong "WebSphere" Tool

---

<sup>1</sup> In the Appendix is the complete text of The Middleware Company retraction letter published in November 2002.

<sup>2</sup> The Middleware Company, "A Message to the J2EE and .NET Communities," November 8, 2002.

<sup>3</sup> The initial Middleware Company Pet Shop study and the J2EE vs. .NET study cited in the retraction in the Appendix.

<sup>4</sup> "Forrester Alters Integrity Policy," CNET News.com, October 7, 2003. "Forrester published a new policy that forbids vendors that commission studies from publicizing their results. The new policy follows the release last month of a study that concluded that companies spent less when developing certain programs with the Windows operating system than with Linux. Giga polled just 12 companies for the study, which led some critics to question the report's statistical relevance."

<sup>5</sup> "Pecked by Penguins," *Business Week*, March 3, 2003. Says the report: "One of the study's authors accuses Microsoft of stacking the deck. IDC analyst Dan Kusnetzky says the company selected scenarios that would inevitably be more costly using Linux."

One of the largest faults of the study is the focus on Rational Rapid Developer (RRD) as the primary J2EE development tool from IBM. The results for RRD are represented as “WebSphere” and RRD is characterized as the main WebSphere development tool. However, IBM has never marketed RRD as a full-blown WebSphere toolset with all the functionality of WebSphere Studio Application Developer (WSAD). WSAD is the tool intended specifically for building integrated enterprise J2EE applications.

Comparing VS.NET to RRD is not a fair comparison, nor is it the comparison IBM would have recommended. Why then did The Middleware Company characterize Rational Rapid Developer as the primary tool for WebSphere J2EE environment? The choice is particularly odd in light of the fact that they were aware that WSAD was the “mainstream” tool for WebSphere development (and that they achieved better results with WSAD). Articles in the trade press reported that the WSAD results were “unofficial” because they were not as controlled as the RRD measurements. This explanation was provided by The Middleware Company and Microsoft.

The WSAD results in this study were comparable to VS.NET in developer time spent on the project. The approach to application development in RRD is very different from WSAD, so the theory postulated in the study that the WSAD results were boosted by the experience the team gained using RRD are not credible.

The bottom line is that The Middleware Company chose to emphasize and give more weight to a tool that would not be recommended by IBM as the primary tool for WebSphere J2EE development. By focusing on RRD, the study draws attention away from the more favorable results posted by WSAD, which is IBM's premier application development tool and the direct competitor to VS.NET. Representing RRD in this way is inaccurate and misleading.

The table below summarizes the development tool comparisons made in the study:

Development Productivity	.NET beats RRD	.NET = WSAD	WSAD beats RRD
Tuning Productivity	.NET beats RRD	.NET = WSAD	“uncertain”
Performance	.NET beats RRD	.NET = WSAD	WSAD beats RRD
Manageability	.NET beats RRD	.NET beats WSAD	RRD beats WSAD

**RRD = Rational Rapid Developer**  
**WSAD = WebSphere Studio Application Developer**

In order to have a fair comparison, you should make more of an attempt at having a level playing field. Here are the specific problems in this study which invalidate the results.

### Unequal Programming Teams

Among the most questionable claims made in the study were that the .NET and J2EE programming teams were equal in programming experience and knowledge of their respective platforms. Having programming teams of equal skills is critical to the credibility of the end results.

The report states that three Microsoft “senior developers” were sourced from a Microsoft Solution Provider (Vertigo Software). The J2EE developers were sourced from The Middleware Company. Why were the J2EE developers not sourced from an IBM business partner that would have had some knowledge about IBM products?

There is an obvious disparity in the skill level of the .NET and WebSphere team exhibited during the actual development work. The Microsoft team exhibits amazing experience and depth of knowledge. An example is the Microsoft team’s decision to code distributed transactions using “ServiceDomain classes” instead of the “COM+ catalog.” ServiceDomain classes are very new, very complex, and not that well understood and documented.

Another example of the high skill level of the .NET developers was their use of “Data Access Application Block” (DAAB) for enhancement of performance. This is not a standard part of .NET and one must go out explicitly and

download and install it. To decide to code the distributed transactions with ServiceDomain classes and to add DAAB for performance enhancement, and to have it work the first time, shows a deep amount of technical skill.

The WebSphere team, on the other hand, exhibited little or no experience with WebSphere Studio Application Developer. Their inexperience is obvious and is easily traced throughout the study:

- The study reports (pg.37) that "the .NET team was undoubtedly more experienced with their tool than the J2EE team with theirs."
- The J2EE team decided to install the Sun JVM rather than the IBM-supplied JVM 1.4.1. WebSphere Studio product requirements state to use the IBM JVM or unpredictable results can occur. The unpredictable results that occurred later (pg. 59-60) were an unrecognized parameter during garbage collection. The time it took to troubleshoot and fix the problem counted against the J2EE team. The substitution of the Sun JVM can only be explained lack of experience with WebSphere Studio. If the intent was to test how WebSphere Studio performed, why would you introduce a non-IBM JVM?
- The J2EE team decided not to do any source code management (pg. 32), even though ClearCase LT and CVS were available. "Instead they divided their work carefully and copied changed source files between their machines." The .NET team used Visual SourceSafe for source control. Does the J2EE decision indicate experienced developers? Problems resulted (pg. 45) for the J2EE developers that increased their development time. It appears again that lack of experience played a role in this poor decision.
- The J2EE team decided to use the Sun J2ME wireless toolkit in Sun One Studio to create an application for a mobile device that was part of the specification. The code generated from the Sun wireless toolkit did not work with the RRD-generated application. Again, this displays a lack of experience in not using the WebSphere Studio tools to interface with the mobile device.
- It took the J2EE team extra time to discover that they had a SIMPLE rights (SELECT rights) user error when they went to access the Oracle database. This counted as developer productivity time and again showed lack of experience.
- The J2EE developer team used POJO (Plain Old Java Object) instead of what best practice might have indicated here – EJBs. It was noted later in the report that the team created a "framework" to handle the JDBC access. If they had used EJB's, they would have had the container handle the pooled requests and it can be argued that the performance would have been better. In the POJO scenario, a new connection had to be created for EVERY call to the database which is very expensive. In addition, they spent extra time developing a "crude" framework to handle the requests, which counted against them.
- The J2EE team chose not to use (or did not seem aware of) the more productive tools available such as Java Server Faces and Service Data Objects.
- The J2EE team specifically chose not to use the Model View Controller approach such as STRUTS technology.

Despite the fact that the three J2EE assigned developers from The Middleware Company had little or no experience with WebSphere Studio, and did not use recommended best practices, **WebSphere Studio was still robust enough to allow the J2EE team to equal the results of the expert Microsoft team using VS.NET in the areas of developer productivity, configuration, tuning and performance.**

## Software Platform Disparity

For a fair performance comparison, the software platforms being tested should be equal. However, in addition to the lack of J2EE developer experience, different databases and operating systems were used in this test. Consider the following:

- Although an attempt was made to make the hardware equivalent, that did not hold true for the base development systems. The Microsoft development team's system consisted of Windows XP and a pre-loaded SQL Server database. The J2EE team's system had Windows XP and a pre-loaded Oracle database. These two different databases were used in the production environment as well. Why were two different databases used by the two teams? At the very least, the databases should have been tuned, optimized, tested, and certified that they were performing equally. We have no information or data to indicate how they compared.

**This flaw by itself should be enough to call the entire performance results into question. Before a single line of code was written, the entire study was suspect.**

- Linux was mandated to run the performance tests. A real performance test should compare one competitor to another with as much commonality as possible. The WebSphere J2EE tests should have been run on the same platform (Windows Server 2003) as the .NET tests.

## Single-Tier Restrictions

The configurations in The Middleware Company study were restricted, which benefited Microsoft in performance vs. the J2EE application. The study did not allow for multiple-tier implementations—configurations that would mirror real-world usage. [Customers like to separate physical tiers of Web applications because it gives them more flexibility for allocation of resources on a tier-by-tier basis and the ability to separate tiers for security and/or geographical concerns.](#)

There were a number of other factors in The Middleware Company study that impeded the J2EE application's performance:

- The Edge server is not needed and adds another layer that is not required. Using the IBM HTTP Server where the Edge Server is, and deleting the two IBM HTTP Servers from the Application Server machine would suffice. This would decrease the complexity and total path length of the application, and allow more processing power for the application servers.
- The teams used different patterns to design the applications. The Microsoft team used a stored procedures pattern while the J2EE team did not. It is quite possible that better performance had been achieved on the J2EE side if it had had the same playing field.
- [Heap size affects the amount of memory allocated to the Java Virtual Machine. One would want to dedicate as much memory as possible to the heap size. The heap sizes were allocated stating that the servers had 1 GB of memory each. This appears to be incorrect.](#) The diagram on page 26 states the servers are 2 GB each, as does the CN2 Technology auditor report. A higher heap size should have been used, something like a minimum heap size of 1,256 MB and a maximum of 1,536 MB.
- The standard method of storing data in the HTTP sessions was not used. The standard method of storing session data for both should have been used.
- The use of two datasources in the J2EE case affects performance negatively. [The two datasources allocated were one, an XA compliant for two-phase commit; the other one was not XA compliant. Their claim was that by splitting the requests to each of the data sources, performance would improve. In theory that might be true; however, it basically precludes sharing of data connections between the two datasources, and all the database requests going through the XA datasource would lock the tables for other transactions. If they had a single XA datasource, they could satisfy the two-phase commit and the connection pooling. Performance might have improved because of the ability to cache the connection pools.](#)
- [In general, the performance runs were done based on achieving a maximum response time of one second. The servers were not driven to maximum capacity.](#) It is never stated what the limiting resource was on the system. A more realistic view of performance and stability is a stress run, where there is no "think time" or wait time between interactions, and the only constrained resource is the application server under test.
- On the integrated scenario, the testers conveniently chose to implement 75% of the application where Microsoft is slightly better than IBM, and only 25% of the application where IBM is better than Microsoft on a 2 to 1 ratio, when run together. If one does the math on the separate scenarios, IBM should have

been 650 vs. 550 for Microsoft, yet with integrated scenario IBM is 482 vs. 855 for Microsoft. There are no details on the throughput for each application in the integrated scenario, and there is no attempt to ascertain the limiting resource(s) during the runs, yet we are expected to believe that the applications themselves ran slower because they were run concurrently on separate servers.

## Cost Manipulations

The study's cost assessment is inaccurate—adding costs to the IBM side while excluding costs that should have been associated to Microsoft. One such extraneous cost was the use of high-end 4-way servers, which was unnecessary and raised the WebSphere configuration cost significantly. For example, a typical configuration like the one in the study could have been deployed on three 2-way servers instead of the 4-way servers that were used.

The Middleware Company's J2EE team initially used a three-node configuration (pg. 28) but added a fourth server to run WebSphere MQ Series. The developers' lack of experience added unnecessary cost to the configuration because a JMS server ships native with the WebSphere Application server at no extra cost. This unnecessary use of hardware and software increased the cost of the WebSphere configuration by more than \$200,000.

While IBM's costs were inflated, we believe the Microsoft configuration costs were understated. The Microsoft configuration was missing an extra year of Software Assurance for the Windows Server 2003 Enterprise Server license. This was due to The Middleware Company's misinterpretation of the terms of the Open Value Agreement, which is three years, not two. The Microsoft configuration should be increased by \$595 per server to cover the cost of an extra year of software assurance.

In a typical configuration, users are required to authenticate before gaining access to Web applications. Each client device would require a Microsoft client access license (CAL). Client access licenses are fees Microsoft charges customers for each user or device that accesses a Microsoft Windows server and Microsoft's middleware server. These fees are costly and are in addition to the base server pricing. In Microsoft's Software Assurance licensing scheme, users pay an additional 25% of the CAL cost per year for three years. The WebSphere configuration doesn't require users to pay an extra cost for client access and also allows unlimited customer and external customer access to the middleware servers. As shown in the charts below, the inclusion of CAL cost increases the overall price of the Microsoft configuration.

Another omission was the MSDN enterprise subscription cost that reflected only one year rather than the three-year terms of the Open Value Agreement. This would add an additional \$5,000 to the Microsoft pricing.

The study's assessment of pricing is skewed in other ways that favor Microsoft. For example, the Microsoft configuration is based on a nonsecure environment that would be considered a security risk in a real-world deployment. The WebSphere architecture provides native reliability and security as a standard and does not require users to pay an extra cost for secure client access.

The Microsoft configuration does not require users to authenticate against Active Directory, allowing anonymous users access to the studies' Web applications. Not using authentication or security simplified the development of the .NET application and left the environment open to the types of security risks that dominate the news headlines.

The Middleware Company's cost model understated the full Microsoft costs required for the scenario. A more thorough study of the total costs of ownership shows that Linux and WAS Express have costs slightly lower than the Microsoft solution. This is in strong contrast to their claim that WebSphere would cost ten times more than the Microsoft solution. Below is our assessment of IBM and Microsoft pricing based on a typical customer deployment.

## WebSphere Configuration with a Secure Customer Environment

Item	Product	Price/Unit		Units		Years of Subscription		3 Year Total Pricing	Annual Cost
Base Server OS	Red Hat Enterprise Linux AS Standard Subscription/ per server pricing	\$1,499	x	3 servers	x	3	=	\$13,491.00	\$4,497.00
Application Server Function	WebSphere Application Server Express/ per processor pricing	\$2,000	x	2 processors per server; 3 servers	x	n/a	=	\$12,000.00	\$4,000.00
Annual Maintenance Agreement	WebSphere Application Server Express/ per processor pricing	\$400	x	2 processors per server; 3 servers	x	2 (1st year of 3 included in license)	=	\$4,800.00	\$1,600.00
Developer Tool	WebSphere Studio Site Developer v. 5.1/ per user pricing	\$1,000	x	2 seats	x	n/a	=	\$2,000.00	\$666.67
Annual Maintenance Agreement	WebSphere Studio Site Developer v. 5.1/ per user pricing	\$200	x	2 seats	x	2 (1st year of 3 included in license)	=	\$800.00	\$266.67
<b>Total</b>								<b>\$33,091.00</b>	<b>\$11,030.34</b>

### Assumptions:

- IBM pricing based on Passport Advantage express pricing
- IBM base middleware pricing includes 1 year of support and software maintenance
- The IBM solution provides 24x7 telephone support for Severity 1 problems. Anyone on the IT staff can call-in.
- IBM server configuration is based on three 2 way servers; four servers are not required
- RedHat pricing is based on a annual subscription that includes software license, maintenance and support

## Microsoft Configuration with a Secure Customer Environment

Item	Product	Price/Unit		Units		Years of Subscription		3 Year Total Pricing	Annual Cost
Base Server OS	Windows Server 2003 / per server pricing	\$2,384.00	x	4 servers	x	n/a	=	\$9,536.00	\$3,178.67
Software Assurance (SA)	Windows Server 2003 SA/ annual pricing	\$596.00	x	4 servers	x	3	=	\$7,152.00	\$2,384.00
Client Access License	Windows CAL/ per user pricing	\$29.00	x	40 users	x	n/a	=	\$1,160.00	\$386.67
Client Access License Software Assurance	Windows CAL SA/ annual per user pricing	\$7.25	x	40 users	x	3	=	\$870.00	\$290.00
Developer Tool	MSDN Enterprise Subscription/ per user pricing	\$1,992.00	x	2 seats	x	3	=	\$11,952.00	\$3,984.00
Developer Tool Software Assurance	MSDN Enterprise Subscription SA/ annual per user pricing	\$498.00	x	2 seats	x	3	=	\$2,988.00	\$996.00
Media	Media Kit	\$23.00	x	1 kit	x	1	=	\$23.00	\$7.67
<b>Total Cost of Solution</b>								<b>\$33,681.00</b>	<b>\$11,227.00</b>

**Assumptions:**

- Microsoft pricing data is based on the Middleware Group pricing provided in Appendix18 of the study.
- Microsoft pricing is based on Microsoft Open Value. The agreement has a 3 year term period.
- The MS pricing only covers customer support during normal business hours and web support 24 x7. Only one authorized IT caller is allowed.

# APPENDIX

## **Apology and Retraction Letter from The Middleware Company regarding the J2EE vs. .NET Pet Store benchmark that was completed in October 2002.**

### **A message to the J2EE and .NET Communities**

November 8th, 2002

The feedback we have received about the J2EE verses .NET benchmark we posted was overwhelming to say the least. We would like to thank those who have spent the time to review our benchmark, and have provided extensive feedback on the benchmark. In response to the feedback we would like to make a few public statements about the benchmark:

- Our original intention for this benchmark was to put forth a more realistic comparison of J2EE and .NET to replace the original Pet Store comparison that Microsoft published. The TMC team worked to make extensive optimizations to Sun's implementation, which resulted in a 17x performance increase. Further information about the technical decisions we made in this project are available in our [report](#) as well as our [FAQ](#).
- However, since producing the report, we received valid feedback about additional J2EE optimizations that could be performed. We also received suggestions about using alternative J2EE architectures than the architecture we chose to test (for example, using CMP instead of BMP, or using no EJB at all).
- We have reviewed this feedback and have concluded that testing these suggestions are important and necessary to be fair to J2EE. The alternative architectures in particular may show materially different performance results. We have learned a very costly lesson here -- that even though a great deal of time, effort, and care went into this benchmark, it is important to give a complete view of how J2EE and .NET compare for the J2EE community to accept the results.

- We also admit to having made an error in process judgment by inviting Microsoft to participate in this benchmark, but not inviting the J2EE vendors to participate in this benchmark. We now realize that following this procedure is critical for the J2EE community to accept benchmark results and perceive them as credible.
- Because of the above issues, definitive conclusions about how J2EE verses .NET compares cannot be drawn unless a 2nd test is conducted which takes the above into account.

We do not currently have a public position yet about whether a round 2 will be conducted, especially given the public beating we just took. We have heard everything from accusing us of having sold our opinion to Microsoft, to that our parent company, Precise Software, is a strategic partner of Microsoft and that somehow tainted the results.

The above accusations are far from the truth, and we are disappointed that this has turned so ugly. We have done a lot of hard work for the J2EE community over the past few years, building TheServerSide.com community, writing the leading books on the technology, and much more. In fact, the reason we started this company was to help customers succeed. That is what gets us excited about our work. We tried to make J2EE perform well to show as realistic a comparison as possible, because we knew that one day we would need to post the results. The irony is that the perception right now is completely the opposite -- which this was a rush job, and that we sold our opinion to Microsoft as a marketing gimmick. This really hurts.

Going forward, we have an important decision to make. Despite our wounds, as engineers, a fair and complete performance comparison of J2EE verses .NET remains extremely interesting to us. A round 2 held in a fair and equitable fashion would be useful to get the facts out there about how J2EE and .NET compare, so people can make informed decisions. It would also be a chance to see how different J2EE architectures compare, such as BMP verses CMP, and EJB verses no EJB. We are willing to completely recode Pet Store with modern design patterns if necessary to achieve this, so that it is more benchmark-worthy -- something we did not have time to do in round 1.

A 2nd round test would fully involve vendors from both the Microsoft side and the J2EE side. It would also involve you -- the J2EE community at large. The process would be public, open, and auditable. The community would have the chance to review code as it is developed, and provide feedback to ensure J2EE is well represented. We would use many of the common performance and architectural suggestions received based on the first study. We have not made a final decision about whether to hold a round 2 benchmark, since this is a very controversial subject, with many faces. We are interested in hearing your opinion on the matter, to help us make the best decision for the community. The two questions we have for you are:

A) Do you think we should conduct a 2nd benchmark?

B) How do you think that benchmark should differ from the current benchmark?

We will review your feedback, take it into serious consideration, and then post our decision.

Please post your feedback here:

[http://www.theserverside.com/home/thread.jsp?thread\\_id=16398](http://www.theserverside.com/home/thread.jsp?thread_id=16398).