

On The Relative Advantages of Teaching Web Services in J2EE vs. .NET

Sandeep Kachru
Blackbaud, Inc,
2000 Daniel Island Drive
Charleston, SC 29492-7541
+1 843-216-6200
Sandeep.Kachru@blackbaud.com

Edward F. Gehringer
Dept. of Computer Science
North Carolina State University,
Raleigh, NC 27695-8206
+1 919-515-2066
efg@ncsu.edu

ABSTRACT

.NET and J2EE are the two leading technologies in enterprise-level application development. They are also the platforms of choice for developing Web services. We compare the two platforms using parameters such as features present in each platform, tools and resources offered by the two and compatibility with the rest of the curriculum. .NET offers integrated, native support for various phases of Web services development, while the Java platform achieves this with several new libraries. Finally, we compare the Web-services development process in IBM's Websphere (for J2EE) and Microsoft's Visual Studio .NET and find them remarkably similar. Both the tools provide comparable features to develop Web services easily.

Arguments in favor of J2EE are platform independence, multiple vendor support, the popularity of Java in universities, and a larger number of tools and resources from which to choose. Points favoring .NET include support for multiple languages, and integrated (rather than add-on) support for Web services. The disadvantage of single-vendor support in .NET must be weighed against J2EE's single-language support. We conclude that there is no clear winner; the choice will depend primarily on local factors. Finally, we provide a road map to help educators choose between the platforms.

Categories and Subject Descriptors

J.m [MISCELLANEOUS]

General Terms

Languages/ Theory

Keywords

.NET, J2EE, Web services, Java, C#

1. INTRODUCTION

The emergence of Web services has created excitement in industry. Not a day goes by without news of some company saving millions of dollars by integrating Web services into its business model. A famous example of this is the partnership between Southwest Airlines and Dollar Rent-A-Car [10]. The research firm IDC predicts [16] that Web services will become the dominant

distributed computing architecture in the next 10 years, and that Web services will drive software, services and hardware sales of \$21 billion in the U.S. by 2007 and reach \$27 billion in 2010. This will obviously create a demand for computing professionals with expertise in this field. In the near future, universities will begin teaching Web services in their distributed programming course, or as a separate course. This will require educators to choose a platform for teaching Web services. Currently, two platforms dominate the market: Java Enterprise Edition (J2EE) and Microsoft's .NET. Competition between them has produced a lot of debate in industry and in the IT press. A recent example is the two papers published [1, 2] in the June 2003 *Communications of the ACM*, one by proponents of each technology. While such papers discuss the merits of the platforms, they do not directly address the needs of the academic community.

1.1 Introduction to Web services

The World Wide Consortium (W3C) defines a Web Service as “a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols.” URI means “Uniform Resource Identifier,” and is considered to be globally unique. XML is the acronym for eXtensible Markup Language, a language designed to describe data using custom tags. In simple terms, a Web service is just a subroutine on a server that can be called remotely by the client.

Any of the various available protocols can be used for the description, discovery and invocation of Web services as long as they are XML-based. However, three protocols—SOAP, WSDL and UDDI have gained industry-wide acceptance and are now considered to be the de facto standards for Web services.

1.2 Introduction to J2EE

J2EE is a set of specifications, created by the Java Community Process (JCP), for developing enterprise-level applications. As a framework for the development of enterprise-level multi-tier applications, it simplifies the task of developing applications for multi-tier architecture by providing “containers.” Containers provide certain complex functionality so that software developers can concentrate on writing the business logic. Figure 1 illustrates the J2EE architecture. The four levels are discussed in Table 1, on the next page.

The latest version of the J2EE specification has been augmented with the addition of several libraries to support Web services. The two primary APIs are as follows:

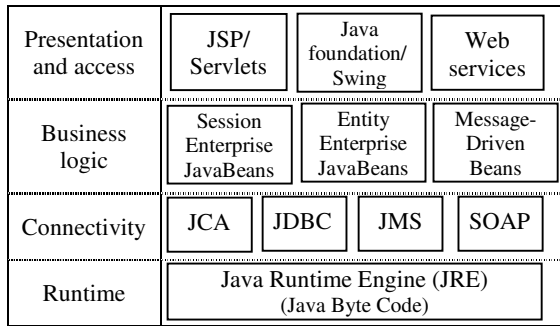


Figure 1: J2EE Architecture [3]

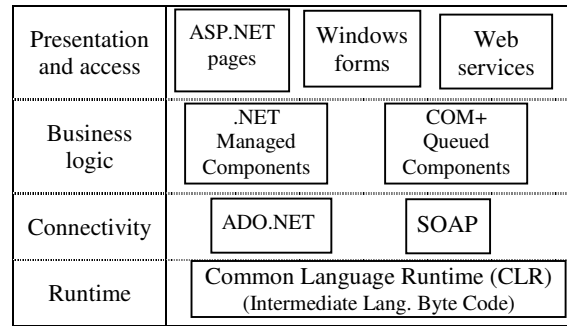


Figure 2: .NET Architecture [3]

- Java API for XML-Based RPC (JAX-RPC) is an API that enables developers to develop and deploy Web services.
- Java API for XML Registries (JAXR) provides a uniform and standard API to access different kinds of XML registries.

Several other APIs provide functionalities like sending and receiving XML-based messages (JAXM), processing XML (JAXP), and binding Java objects to XML documents (JAXB). J2EE is currently the market leader in enterprise application development. It is an attractive choice as a development platform for the companies. The main benefits of using J2EE are:

- Platform independence: Java technology essentially works independently of any single hardware architecture or operating system. The development platform is available for Windows, Mac and various flavors of Unix including Solaris, Linux, HP-UX.
- Multiple-vendor support: Sun Microsystems supplies a comprehensive J2EE Compatibility Test Suite (CTS) to the J2EE licensees. The J2EE CTS helps ensure compatibility among the application vendors which helps ensure portability for the applications and components written for J2EE. J2EE licensees include many heavyweights of the software industry like IBM, Borland and Oracle.

1.3 Introduction to .NET

.NET is a Microsoft product tied closely to the Windows operating system. Microsoft describes it as software that connects information, people, systems, and devices. .NET provides a development framework similar to J2EE for multi-tier enterprise application development. Figure 2 illustrates the .NET development platform. The main benefits of using .NET are language independence and integrated support for Web services.

.NET is a successor to the older Microsoft technologies like COM and DCOM. It has been substantially improved with the addition of several new features. The main benefits of using .NET are as follows:

- Language independence: A developer using the .NET platform has a wide range of choice in terms of programming language. He can use the languages part of .NET, e.g., VB.NET, C#, JScript.NET and C++ with managed extensions or variant of languages like Fortran, Smalltalk, Cobol etc. which have been modified to target CLR.
- ♦ Integrated Web services support: .NET has inbuilt support for developing and deploying Web services. Developing, publishing and discovering a Web service is very simple in .NET. Web services appear just like other objects; there is no visible difference to the programmer once the reference to the component is set.

Table 1: Levels of functionality in the two platforms

Level	J2EE	.NET
1. Presentation and access	Java Server Pages (JSPs) are used to build tag-oriented dynamic Web pages for accessing remote objects. Dynamic pages can also be built programmatically using servlets. Swing is used to build rich interactive GUIs.	.NET uses ASP.NET for dynamic HTML pages. Windows Forms are used for building rich and complex GUIs, and Web services are used for programmatic access to remote business logic.
2. Business logic	Enterprise JavaBeans (EJBs) hold the application's <i>business logic</i> —the code that implements the functionality of the application.	.NET Managed Components are made for the .NET environment, and unlike COM components, are not registered in the registry. COM Queued components work asynchronously, e.g. in scenarios where the server is not online all the time.
3. Connectivity	Java Database Connectivity (JDBC) provides access to tabular data. Java Connector Architecture (JCA) allows J2EE components to access different enterprise information systems. JMS is a messaging standard that allows J2EE components to send and receive messages asynchronously. An extensive API is provided for mapping between Java and XML protocols.	ADO.NET is used for accessing relational databases and provides integration with XML. An XML API is provided for mapping .NET Components to XML protocols such as SOAP and WSDL.
4. Runtime	Java Runtime Engine (JRE), which includes the Java Virtual Machine (JVM), core Java classes and supporting files.	All .NET applications use a single runtime engine, the Common Language Runtime (CLR). Applications can be written in multiple languages, compiled to Microsoft Intermediate Language byte code, and executed in the CLR.

2. PLATFORM VS. LANGUAGE INDEPENDENCE

J2EE is a platform-independent technology while .NET is currently available only for Windows operating system. On the other hand, .NET supports development in a number of languages whereas J2EE is a Java-only technology.

2.1 Platform Independence

A technology can be said to be platform independent if it can be ported to different hardware architectures or operating systems without requiring changes.

J2EE is a platform-independent technology. It works on several operating systems including those for embedded devices. Java embodies Sun's "write once, run everywhere" philosophy. Its architectural neutrality comes from the fact that the Java compiler generates byte-code instructions targeting the Java Virtual Machine (JVM) instead of executable machine code. A JVM is available for hardware devices such as handheld computers, cellular phones and operating systems like Windows, UNIX, MacOS, Solaris and Linux. For a program to be platform independent, the developer must avoid native methods and make sure that the required libraries are present on the target computer.

Being a Microsoft product, .NET is usually considered to be "Windows only." However, there have been efforts to make it available on other platforms. Microsoft, along with Intel and HP, submitted parts of .NET – the programming language C# and the Common Language Infrastructure (CLI) to ECMA for standardization [11]. This opens the way for the implementation of these two fundamental elements of .NET framework in non-Windows systems. Microsoft has provided a shared source implementation of CLI, codenamed Rotor [12], as an implementation of these standards in source-code form. Rotor currently can be built and run on Windows XP, Free BSD and Mac OS. An open-source implementation of .NET called Mono is being created for the UNIX environment by Ximian, a company recently acquired by Novell Inc. Its code can be used for commercial purposes.

2.2 Language Independence

Language independence means that a technology is not dependent on any particular programming language, i.e., that a developer can use any supported language to develop code on that platform. Though the Java platform has been built with the Java language in mind, it can also be used with other languages. Limited language independence has been achieved in the Java platform using the following approach:

1. Using the Java Native Interface. A program can use JNI to make calls from Java code to methods written in languages other than Java, e.g., C or C++.
2. Using Java as an intermediate language. Source code written in some other language is first translated into Java code and then compiled using the Java compiler.
3. Compiling to Java byte code: Using this method, languages other than Java target the JVM, i.e., on compilation they produce a `.class` file which can be run on the Java platform.

In spite of these options, very few commercial projects actually use Java with the approaches outlined above.

.NET takes language independence to a whole new level. It allows complete language integration. All languages supported by the .NET framework share a common API provided by the framework. A class written in one language can be extended by a

class in another language and used in a third language. Exceptions raised in a method of a class can be caught by the calling method in another language. Compilers for .NET-supported languages compile source code into an intermediate form composed of code and metadata. As all compilers produce code in the same format, there is no difference between code written in Cobol and code written C# (as long as both target the CLR).

Thus, .NET is language independent and supports limited platform independence. On the other hand, J2EE is platform independent and supports limited language independence. Which of these two factors is more important in an academic setting? A language-independent platform allows students to use any language to solve a given problem. Using such a platform in a Web-services course would allow students to program in the language in which they are most proficient. This would allow them to concentrate on learning the fundamentals of Web services instead of learning a new language. Platform independence is also important, as it expose students to different platforms and help increase their career opportunities. J2EE, being platform independent and standards based, allows students to choose between tools and resources provided by many vendors.

3. WHICH PLATFORM HAS BETTER WEB-SERVICES SUPPORT?

In this section we will compare the process of developing a simple Web service, deploying it on a server, publishing it on a UDDI registry and invoking it from a client application using the two platforms. Microsoft provides an integrated solution for all the above tasks in the form of Microsoft .NET Framework SDK. For the Java platform, we will use the reference implementation provided by Sun Microsystems and also third-party products, like Apache's Axis and UDDI4J from various vendors.

3.1 Developing and deploying a Web service

We developed a simple Web service that is similar to the ubiquitous "Hello World" program used to teach any new programming language. Our Web service takes a name of a person as a parameter and returns the greeting "Hello", appended with the person's name. For example, a person who calls the service with the parameter "David" will be greeted, "Hello, David." This program just illustrates the process of building and deploying a Web service using the two platforms. It does not serve any useful purpose. The same process would be used to build and deploy complex Web services.

On comparing the Web services coded in Java and C# [8], we find that they use almost the same number of lines of code (Table 2). To deploy the service on the Axis server, we write a descriptor file in XML that lists the methods of the Java class that need to be exposed as Web services. Deploying the C# service on the server does not require writing any extra code, and can be done by copying the compiled code to the server. This is possible because .NET provides attributes like "WebMethod" and "Web service" which can identify methods that need to be exposed. On the other hand, Axis descriptors allow us to expose or hide methods without making any change in the source code. This is not possible in .NET. To test the Web service deployed on Axis, we need to write a client program that uses the Axis libraries. A Web service can be tested using Internet Explorer, which provides an interface to invoke the Web service and display the results. This feature is very useful to developers in testing the Web service instantly.

3.2 Invoking a Web service

To compare the code required for invoking a Web service, we wrote client programs in Java and C# [8] to invoke the Web service provided by Amazon.com to search for books. To invoke this Web service, we first generate client stubs to make the calls to the Web service transparent to the client.

Both Axis and .NET provide a tool to take the WSDL file of the Web service and generate client stubs. The client programs written in Java and C# are almost identical. The C# code has slightly fewer lines than Java, because C# has a “**for each ... next**” statement to loop through collection objects. While this difference is small in our sample application, it could be significant in a large project using many collection classes.

3.3 Publishing and discovering Web services

After the Web service has been developed and deployed on the Web server, the next step is to publicize it so that other developers can use it. The most popular way to publicize a Web service is to publish it in a UDDI registry. Users can then search the registry to find a Web service that meets their requirement.

.NET technology includes an SDK for interacting with the UDDI registries. For J2EE, we used the UDDI4J API instead of the JAXR implementation provided. We wrote sample programs to publish and search for businesses using the platforms [8] and again find that the two programs are comparable in lines of code (LOC) and identifiers used (Table 2). This suggests that the two platforms are about equally easy to code for.

Table 2: Comparing simple J2EE and .NET programs

Sample	.NET			J2EE		
	LOC	Identifiers		LOC	Identifiers	
		Dis-tinct	Total refs		Dis-tinct	Total refs
Building a Web service (includes the descriptor file for Java)	10	6	8	12	5	6
Building client for Amazon Web service	38	35	80	35	38	94
Publishing business on UDDI registry	10	19	27	13	19	30

Thus, the two platforms provide similar support for development of Web services. .NET currently provides an integrated, native solution. Native support for Web services is also expected in the next Java specification.

3.4 Performance benchmarks

Sun Microsystems introduced the Java Pet Store as a demonstration implementation for J2EE-based Web applications. It illustrates various best practices in application development and is provided as a design pattern for customers to follow when building their own enterprise Web applications. Microsoft re-implemented the Java Pet Store using .NET and C# to demonstrate the superiority of the .NET platform over J2EE. They released benchmark information [4] showing .NET Pet Shop performance to be significantly better under high user loads than the Java equivalent. In October 2002, The Middleware Company, which

provides Java training and also maintains online developer resources for the Java community, performed its own benchmarks on the Java Pet Store and .NET PetShop applications [5]. Three tests were performed: a Web application test, a reliability test, and a Web service throughput test. The results showed that .NET based application outperformed J2EE application by a wide margin. Because of the controversy generated by the benchmark tests, the company decided to incorporate suggestions of Java developers and perform another set of tests [6]. Results released in June 2003 showed that the optimized Java Pet Store performed as well as the .NET application in the Web application throughput and reliability test. However, the .NET application still outperformed the J2EE application in the Web services throughput test. Table 3 presents selected results from the case study.

Table 3: Performance of .NET vs. J2EE-based application [6]

Test	.NET-based app	J2EE-based app
Web application peak throughput using Oracle database	1586.54 Web pages per second	1585.74 Web pages per second
Average transactions (Web pages)/sec. processed over 24 hrs.	1136 avg Web pages per second	1150 avg Web pages per second
Peak throughput	1245 Web services requests/sec.	359 Web services requests/sec.

3.5 Which will be more useful in the future?

As educators, our goal is to teach students principles that will outlast specific technologies. But, given two otherwise appropriate technologies, commercial staying power might be a reason to choose one over the other. A survey [18] conducted in June 2002 revealed interesting insights on the adoption of .NET by industry. Completed by 633 development managers, the survey showed that .NET had already gained a substantial amount of support from industry. It showed while currently the ratio of projects using .Net to J2EE is just 28% to 48.8%, in future this will increase to 52.6% to 51.8%. Another survey [17] conducted in October 2002 asked more than 600 developers about the development platform they are using and will be using in the future. Similar results were obtained, with 63% saying that they will be targeting .NET platform and 61% saying they will be targeting the Java platform. From this, we can infer that either choice is viable for our students’ short-term career prospects.

4. COMPARING COMPATIBILITY WITH EXISTING CURRICULUM

An important factor in choosing the platform for teaching Web services is compatibility with courses already being taught in the department. Most CS departments have one or two programming primary teaching languages. If the primary language is Java, then it is obvious that students will be more familiar with the Java technology. Similarly, if students have used .NET based languages like C# or VB.NET in previous courses they will have more familiarity with .NET platform.

An annual survey [7] of computer science departments reports on their primary teaching languages. Forty-five computer science departments were surveyed with various questions on curriculum, faculty and students. The following table shows how Java has gained at the expense of C and C++ in recent years.

Table 4: C/C++ vs. Java as a primary teaching language [7]

Year	C	C++	Java
1998-99	20%	50%	22%
1999-00	19%	54%	22%
2001-02	11%	40%	49%
2002-03 (Projected)	9%	40%	56%

Further, a short survey of the primary teaching languages in the top 25 US computer-science departments [8] shows that 13 of them teach Java, compared to 6 that use C and 4 that teach C++. Java is thus the most popular language used for teaching in universities today. This suggests that in the immediate future, J2EE may take the lead over .NET as a teaching platform.

Another important factor in this discussion is the rising popularity of C#. Many major universities have started using C# and .NET to teach courses in object-oriented programming, Web programming, etc. CMU uses C# and .NET to teach a course in Web-application development [13] and Cornell had an introductory programming course in C# in Fall 2002 [14]. Yale uses C# as an introductory programming language [15]. Several features of C# may make it a more attractive language than Java for CS1 and CS2 [9].

- Reading from standard input is much simpler in C# than in Java.
- The object model is more consistent. Variables with primitive data type don't need to be boxed and unboxed when the language is expecting an object.
- C# introduces the concept of properties, which removes the need to have get and set methods like in Java.

- C# allows passing of parameter values by reference, which is not possible in Java.

Though C# offers some advantages over Java, they are not significant enough to justify changing the primary language from Java to C#. Universities which use Java will also be reluctant to move over to C# because of other reasons like availability of free implementations of Java-related products, maturity of the language and general goodwill for the open standard of Java. However, universities using languages like C++ and Scheme and wanting to move to a better language now have a choice between Java and C#. They may find C# to be an attractive choice because of the reasons outlined above. That said, it has been four years since C# was announced. The inroads it has made into curricula are far less than Java achieved in the first four years after its public release in 1995. This suggests that C# is unlikely to replace Java as a market leader in education, and J2EE will remain the platform of choice for many universities desiring homogeneity in their curriculum.

5. COMPARING TOOLS & RESOURCES

Building a simple Web service in .NET and J2EE is very easy. It can be done using a text editor and command-line tools. However, building complex industrial-strength Web services requires the use of specialized tools, such as an integrated development environment (IDE). Generally, in industry such IDEs are used to build complex applications, as they improve developer productivity by providing various in-built features that save time. They are especially useful for providing context-sensitive help, debugging and testing support. It is important for students to learn about such tools so that they can use them in their job. Familiarity with such tools will boost students' résumés.

Table 5: Comparison of WSAD 5.0 with Visual Studio .NET

Metric	Websphere Studio Application Developer Ver 5.0	Microsoft Visual Studio .NET
Support for creating Web services	Web services can be created from existing Java files, EJB etc. The developer only needs to select the functions that need to be exposed as a Web service, and the server on which the service needs to be deployed, and the wizard takes care of rest.	Creating a Web service is similar to creating any other application. A project of type "Web service" is created and deployed on the server. The functions that are to be exposed as Web service are marked with the attribute "Web Method."
Support for creating Web services client	A wizard to create the proxy files is available in WSAD. The wizard also creates a test client in form of a JSP page from which the Web service can be invoked.	To invoke a Web service, the user needs to add a "Web reference." The user can then call the Web service functions just like any other function.
Support for publishing Web services and exploring the UDDI registry	Web services can be published programmatically by using the UDDI library. WSAD also provides an interface to publish the Web service on various registries.	Web services can be published programmatically using the UDDI SDK. Visual Studio .NET does not provide an interface to publish the Web service in different registries.
Support for building database queries	No support for building stored procedures. Requires a separate database tool.	Support for building stored database queries is integrated.
Cost	WSAD is available for evaluation for 60 days. It is also available to faculty members of IBM Scholar Program [40]. The services can be deployed on a free server like JBoss and Tomcat.	Visual Studio .NET is available for free for members of Microsoft Academic Alliance [41] program. The server IIS is integrated with the latest operating systems like Windows 2000 and Windows XP Pro

A number of tools are available for development in J2EE. Large companies like Sun, Oracle, IBM and BEA provide IDEs that make development in J2EE simple. Notably, IBM's WebSphere Application Developer is available to faculty members through the IBM Scholar Program. The services can be deployed on a free server like JBoss and Tomcat. Fewer tools are available for .NET. However, Microsoft provides an excellent development tool—Microsoft Visual Studio .NET—for developing .NET applications. It is free to members of the Microsoft Academic Alliance.

Table 5 (previous page) compares Websphere with Visual Studio .NET. A similar comparison has been done by GotDotNet.com [19]—a site affiliated with Microsoft—and by IBM [20]. Since then, a new version of WSAD has been released which provides better support for Web services. We will use metrics from both the comparisons to compare the latest version of WSAD with Visual Studio .NET. We see that the two development environments are very similar to each other, providing similar functionalities and resources to the developers.

Neither platform suffers for lack of books. A summer 2003 search of the Amazon and Barnes & Noble Web sites revealed 1770 books with the keyword “Java” and 217 with the keyword “C#”. However, when the term “Web services” is thrown into the search, more books were found for .NET (46) than for “Java” and “Web services” (21). However, even the smaller number is more than adequate for teaching.

Among other resources, Microsoft's MSDN is a huge repository of articles and tutorials. It also provides message boards on which students can post their problems. Most, if not all, J2EE vendors maintain Web sites providing resources for developers. The Developerworks Web site maintained by IBM is a rich source of very helpful resources. Similarly, the Web sites of Sun, Oracle and Borland also contain a number of online tutorials, code samples etc. Again, being an older platform with support from many vendors, J2EE is more fully covered than .NET.

6. CONCLUSION

We examined a number of factors of importance to the educators in choosing a platform to teach Web services. We have seen that .NET and J2EE provide comparable support for the development of Web services through rich and extensive APIs and powerful development tools. Either one is adequate for teaching Web services. The choice is likely to be based primarily on local factors. We provide a road map in form of a flowchart in Figure 3 to help educators make an informed decision.

7. REFERENCES

1. G. Miller, “The Web services debate: J2EE vs. .NET,” *Communications of the ACM* 46:6 (June 2003), Pages 64-67.
2. J. Williams, “The Web services debate: J2EE vs. .NET”, *Communications of the ACM* 46:6 (June 2003), Pages 58-63
3. M. Lehmann. (2002) “J2EE and Microsoft .NET,” Oracle Whitepaper.
http://www.oracle.com/ip/develop/ids/jdevdocs/J2EEandNET_wp.pdf
4. Microsoft (2001) "Microsoft .NET vs. Sun Microsystem's J2EE™: The Nile Ecommerce Application Server Benchmark," Microsoft Corporation Whitepaper.
<http://www.gotdotnet.com/team/compare/Nile%20Benchmark%20Results.doc>

5. Middleware Company (2002) "J2EE and .NET Application Server and Web Services Benchmark," Middleware Company Benchmark Report.
<http://www.middleware-company.com/documents/j2eedotnetbenchmark.pdf>
6. Middleware Company Case Study Team (2003) "J2EE and .NET (RELOADED) Yet Another Performance Case Study," Middleware Company Case Study Report.
<http://www.middleware-company.com/casestudy/tmc-performance-study-jul-2003.pdf>
7. R. A. McCauley and B. Manaris. (2002) Comprehensive Report on the 2001 Survey of Departments Offering CAC -Accredited Degree Programs. Technical report CoC/CS TR# 2002-9-1, Department of Computer Science, College of Charleston.
8. S. Kachru (2003) "On the relative advantages of teaching Web services in .NET Vs. J2EE," Master's Thesis, NCSU.
<http://www.lib.ncsu.edu/theses/available/etd-08172003-193313/>
9. S. Reges. (2002) "Can C# Replace Java in CS1 and CS2?" *ACM SIGCSE Bulletin*, 34: 3 (September 2002).
10. The Stencil Group. (2001) "How Web Services Will Beat the 'New New Thing' Rap," An Analysis Memo from The Stencil Group.
http://www.stencilgroup.com/ideas_scope_200106newnew.html
11. <http://msdn.microsoft.com/net/ecma/>
12. <http://research.microsoft.com/programs/europe/rotor/>
13. <http://www.cs.cmu.edu/education/courses/>
14. <http://www.cs.cornell.edu/courses/cs215/2003sp/>
15. <http://flint.cs.yale.edu/cs112/index.html>
16. http://www.idcresearch.com/getdoc.jhtml?containerId=pr2003_02_03_130651
17. http://www.infoworld.com/article/02/10/09/021009hndevsurvey_1.html
18. <http://www.sdtimes.com/news/057/story7.htm>
19. Building and Integrating XML Web Services with Visual Studio .NET vs. IBM Websphere 4.0.
<http://gotdotnet.com/team/compare/webservicecompare.aspx>
20. How IBM WebSphere Studio Application Developer Compares with Microsoft Visual Studio .NET.
http://www7b.boulder.ibm.com/wsdd/techjournal/0202_kraft/kraft.html

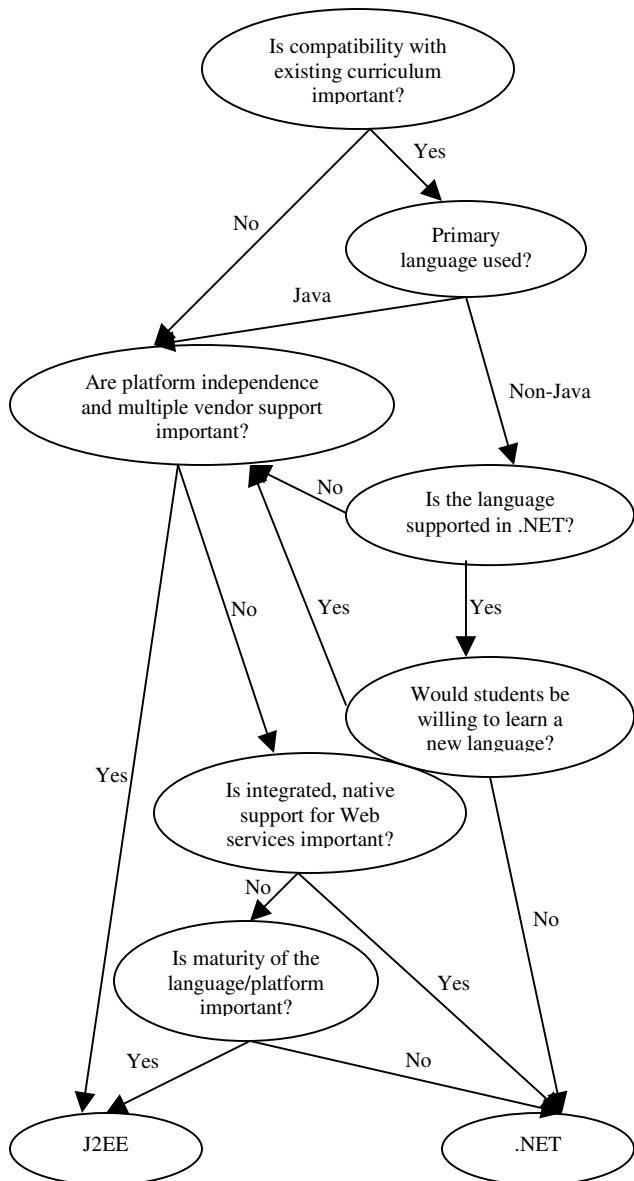


Figure 3: Roadmap for choosing a platform