International Journal of Engineering Research and Development
e-ISSN: 2278-067X, **p**-ISSN: 2278-800X, www.ijerd.com
Volume 7, Issue 1 (May 2013), PP. 116-121

A Comparison of J2EE and .NET as Platforms for Developing E-Government Applications

Dr.R. Udayakumar[1], Dr.K.P. Thooyamani[2], Dr.V. Khanaa[3]
[1]Asst.Prof, IT, Bharath University, [2]Prof,Bharath University, [3]Dean, IT, Bharath University

Abstract:- .NET and J2EE are the two leading technologies in enterprise-level application development. They are also the platforms of choice for developing e-Governance Application. We compare the two platforms using parameters such as Platform Independence, Language independence, Web services, Support for existing system, scalability, system cost, Migration from the previous platforms and Tools. .NET offers integrated, native support for various phases of Web services development, while the Java platform achieves this with several new libraries. We compare the Web-services development process in IBM's Websphere (for J2EE) and Microsoft's Visual Studio.NET and find them remarkably similar. Arguments in favor of J2EE are platform independence, multiple vendor support 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 disadvantages of single-vendor support in .NET must be weighed against J2EE's single-language support.

Index Terms:- NET, J2EE, Web Services, Java, C#, e-Governance.

I. INTRODUCTION

Governments all over the world are moving towards the electronic governance or e Governance. The purpose of implementing e governance is to enhance good governance. Good governance is generally characterized by participation, transparency and accountability. The recent advances in communication technologies and the Internet provide opportunities to transform the relationship between governments and citizens in a new way, "thus contributing to the achievement of good governance goals. The use of information technology can increase the broad involvement of citizens in the process of governance at all levels by providing the possibility of on-line discussion groups and by enhancing the rapid-development and effectiveness of pressure groups. Advantages for the government involve that the government may provide better service in terms of time, making governance more efficient and more effective. In addition, the transaction costs can be lowered and government services become more accessible. Hence many governments are moving towards e Governance and it involves bigger level application development. This will require developers to choose a platform for developing e Governance applications. Currently, two platforms dominate the marker : java Enterprise Edition (J2EE) and Microsoft's .NET. Competition between them has produced a lot of debate in the industry.

This paper compares the J2EE and .NET, two popular technologies involved in e governance application development under e governance application development environment standards [7].

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 1.1 illustrates the J2EE architecture. The four levels are discussed in Table 2.1. The latest version of the J2EE specification libraties to support Web services. The two primary APIs are as follows:

- 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 (JAXM), 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).

Presentation and access	ISP/Servlets.	Java fbtmda&on /Swing	Web Services
Business logic	Session Enterprise Java Beans	Entity Enterprise JavaBeans	Message Driven
Connectivity	JCA JDBC JMS		SOAP
Runtime	Java Runtime Engine (JKE) (Java Byte Code)		

Figure 1.1:J2EE Architecure [1]

Presentation and access	ISP/Servlets.	Java fbtmda&on /Swing	Web Services
Business logic	Session Enterprise Java Beans	Entity Enterprise JavaBeans	Message Drivefl
Connectivity	JCA U3BC JMS		SOAP
Runtime	Java Runtime Engine (JRE) (Java Byte Code)		

Figure 1.1: J2EE Architecture [1]

Level	J2EE	.NET
I/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 business logic – 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.	AH. 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.

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 multitier enterprise… application, development, Figure 1.2 illustrates the .NET development platform. The main benefits of using .NET development platform. The main benefits of using .NET are language independence and integrated support for Web Services.

Presentation and access	ASP.NET pages	Windows Web forms services	
Business logic	1.NET Managed component		COM-Queued compon ents
Connectivity	ADO.NETSOAP		
Runtime	Common Language Runtime (CLR) (Inierrofrdiate Lang.Byte Code)		

Figure 1.2 : .NET Architecture [1]

II. PLATFORM VS. LANGUAGE INDEPENDENCE

2 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++.
J2EE is a platform – independent technology white .NET is currently available only far windows operating system. Or 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 it it can be ported to different hardware architecture 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 [2]. This opens the way for the implementation of these two fundamental elements of .NET framework in non-windows implementation of these standards in source-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 : 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 meta data. As all compile 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 languages independence.

III. WHICH PLATFORM HAS BETTER WEB-SERVICES SUPPORT?

3.1. Performance Benchmarks

Sun Microsystems introduced the Java Pet Store as a demonstration implementation for J2 EE-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 J2EE. They released benchmark information [3] 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.

t/5 £	.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 24hrs.	1136 avg Web pages per second	1150 avg Web pages per second.
Peak throughput	1245 Web services request/sec.	359 Web services requests/sec.

Table : 3.2

Performance of .NET VS. J2EE – based application [6] 4.0
Supports for Existing Systems

Most large corporations have existing code written in a variety of languages, and have a number of legacy systems, such as CICS/COBOL, C++, SAP R/3, and Siebel. It is vital that corporations be given an efficient, rapid path to preserve and reuse these investments. After all, it is likely that business will have neither the funds nor the time to reinvent all existing systems. This legacy integration often is one of the most challenging (if not the most challenging) tasks to overcome when building a web service.

There are several ways to achieve legacy integration using J2EE, including

- **The Java Message Service (JMS) to integrate with existing messaging systems**
- **Web services to integrate with any system**
- **CORBA for interfacing with code written in other languages that may exist on remote machines.**
- **JNI for loading native libraries and calling them locally.**

But by far, the most important part of the J2EE vision for integration is the J2EE connector Architecture (JCA). The JCA is a specification for plugging in resource adapters that understand how to communicate with existing systems, such as SAP R/3, CICS/COBOL, Siebel, and so-on. If such adapters are not available, you can write your own adapter. These adapters are reusable in any container that supports the JCA. The major vendors of existing systems are bought into the JCA.

.NET also offers legacy integration through the Host Integration Server 2000. COM Transaction Integrator (COM TI) can be used for collaborating transactions across mainframe systems. Microsoft Message Queue (MSMQ) can integrate with legacy systems built using IBM MQ series. Finally, BizTalk Server 2000 can be used to integrate with systems based on B2B protocols, such as Electronics Data Interchange (EDI) (the reader should note, however, that Biz Talk does not serve as an access point to a proprietary network on which EDI takes place).

In conclusion, we believe that the legacy integration features offered by J2EE are superior to those offered by .NET. the JCA market is producing a market place of adapters that will greatly ease enterprise application integration. Integration with packaged applications and legacy systems will become much easier – imagine integrating with a system such as Siebel, Oracle, or SAP without every leaving the Java Programming environment. There is no analog to this in the Microsoft domain; rather, there is limited connectivity to select systems provided off-the-shelf through the Host Integration Server.

IV. MIGRATION FROM PREVIOUS PLATFORM

For organization that have an existing deployment using either J2EE-based technologies or Windows DNA-based technologies, an interesting discussion is the ease of –migration from the previous platform to the new platform.

J2EE does not impose many migration problems. As previously mentioned, the Java Connector Architecture (JCA) as well as the web services support in J2EE is brand new and will require new code, but those are minor overall. Although Microsoft.NET is based on MTS and COM+, we are concerned that the migration to .NET will be taxing compared to J2EE. First off, .NET is based on the "managed code" framework, which steals a lot of ideas from COM+ and MTS, but it's still an entirely new infrastructure based on an entirely new code base – CLR. Taking advantage of (he most valuable aspects of the CLR impost one-time frictions.

For example to accommodate a Common Type System (CTS) which standardizes on data types used between languages, the original Visual basic data types have been dismissed. Consequently, code dependent upon those original Visual Basic data types will break, and there is currently no migration tool.

Another example is the COM+ migration path. In.NET terminology, code that runs within the CLR is referred to as management code, while code running outside the CLR is called unmanaged code. If you're a COM+ developer and want to take advantage of the new CLR, then you have two options for migration:

Rewrite existing code as CLR code. COM+ code needs to be rewritten to accommodate the CLR's automatic garbage collection mechanism and its deprecation of pointers. Dependence also need to be removed to the COM registry. Keep your existing code as unmanaged. To collaborate between managed and unmanaged code, special measures must be taken.

So as you can see, migration is not free. But to Microsoft's credit, we do understand that with the innovation of the CLR, that this is a necessary step for their customers to evolve into their new platform, and with such a radical change nothing less could be expected. However, we feel obligated to warn users that the migration path will not be easy compared to J2EE migration path, as some might have you believe. Consider these statements from a recent Gartner report:

"This is fundamentally a brand new platform," said Gartner Analyst Mark Driver, comparing the migration to NET as more drastic than the switch from MS-DOS to Windows. "This is the tiger changing its stripes… the migration to .NET will be a difficult one for IT departments because it represents such a major shift from the current Microsoft Platforms. For Instance, developers will have to rewrite as much as 60 percent of the code for some existing Windows applications if they want them to take advantage of Microsoft's .NET platform, Gartner analysts predicted. That's frightening prospect for companies who are currently switching to Windows 2000, or for those who still run Windows 98 and NT.

V. SCALABILITY

Scalability is essential when growing a web services deployment over time, because one can never predict how new business goals might impact user traffic.

A platform is scalable if an increase in hardware resources results in a corresponding linear increase in supported user load while maintain the same response time. By this definition, the underlying hardware (win32, UNIX, or Mainframe) is irrelevant when it comes to scalability, because both J2EE and .NET allow one to add additional machines to increase user load while maintaining the same response time. The major implementations based on J2EE architecture, as well as .NET, provide load-balancing technology that enables a cluster of machines to collaborate and service user load that scales over time.

The significant difference between J2EE and .NET scalability is that since .NET supports Win32 only, a greater number of machines are needed that a comparable J2EE deployment due to processor limitations. This multitude of machines may be difficult for organizations to maintain.

VI. SYSTEM COST

A wide variety of implementations based on J2EE architecture are available for purchase, with price points varying dramatically, enabling a corporation to choose the platform that meets its budget and desired service level. Costs are typically in the single-digit thousands of dollars per processor, although there are higher-end implementations and lower-end ones. At the time of this writing, Microsoft had not released pricing information for the .NET platform.

As far as hardware, J2EE supports UNIX and Mainframe systems, while both J2EE and .NET support the Win32 platform, which is generally the less expensive alternative. There is a level playing field for hardware costs, and the hardware cost debate becomes a moot point.

The takeaway point is that you can get low-cost solutions with both Microsoft and J2EE architecture. Microsoft's solution has an aggressive price, whereas J2EE architecture allows you choose your service level. For example, with J2EE you can have a high-end, expensive solution (iPlanet running on Sun Solaris in an E-10000 server), or a low-end, inexpensive solution (jBoss running on Linux on a Cobalt RAQ server). There's also an assortment of free and/or open source tools and services that support Java and XML. It should be noted that you pay for what you get, and most organization will not go for this low-end solution, but rattier will embrace a midrange solution as a happy medium.

If you are trying to make heads or tails out of the price wars, we recommend that you consider this: the price of the platform is always a drop in the bucket compared to the total cost of the project. This is defined as the price of the server platform, the cost to train developers, the cost to build and evolve a solution on that platform, the cost to maintain the solution, and any business opportunity costs from picking the 'wrong' platform.

We hope that firms realize that the total cost of ownership of a project dwarfs any short-term cost differences between underlying platforms. We recommend you do not consider the price of the platform when selecting between J2EE, .NET, or any other platform, but rather consider the more important other factors.

[6] Middleware Company Case Study Team (2003) "J2EE and .NET (RELOADED) Yet Another Performance Case study," Middleware Company Case Study Report.
[7] EGov Standards Perspective (Draft)
Egovstandards.gov.in/egs/brainstormingsession/egovstandard_perspective-draft.pdf/download

VII. CONCLUSION AND FUTURE WORK

e-Governance is a vast field and researches are going on in this field extensively. We have started our research in this field by making a comparative study between the two leading frameworks used in eGovernance application development .NET and J2EE. In this comparative study we have used the following parameters:

1. Platform VS Language independence.
2. Which platform has better Web-Services Support?
3. Migration from Previous Platform.
4. Supports for Existing Systems
5. Scalability
6. System Cost.

The e Governance application developing community can make use of this comparative study and can select any one of these two frameworks for their development work. In future we like to compare the security features of these two frameworks.

REFERENCES

[1]. M. Lehmann (2002) "J2EE and Microsoft .NET". Oracle Whitepaper. T_wp.pdf

[2]. Microsoft.com/net/ecma/Microsoft (2001) "Microsoft .NET Ecommerce Application Server Benchmark", Microsoft Corporation Whitepaper. rk%20Results.doc

[3]. S. Kachru (2003) "On the relative advantages of teaching Web services in .NET Vs. J2EE", Master's Thesis, NCSU.

[4]. Middleware Company (2002) "J2EE and .NET Application Server and Web Services Benchmark", Middleware Company Benchmark Report. benchmark.pdf.