UNDERLYING PLATFORM OF THE E-COMMERCE SYSTEM: J2EE VS. NET

Hamid Jahankhani, Mohammed Youssef
University of East London, School of Computing and Technology, Longbride Road, Dagenham, Essex RM8 2AS

Keywords: Microsoft.NET, Web Services, J2EE, E-business solutions, Syspro, ERP, SME.

Abstract: when considering the implementation of any new Web-based application these days, the main two options available to developers are to either base the application on Sun Microsystems' J2EE (Java 2 Enterprise Edition) platform, or on Microsoft's .NET platform. Although other platforms do exist, the IT industry has identified these two as the main choices. .NET initiative is a broad new vision for embracing the Internet and the Web in the development, engineering and use of software. One key aspect of the .NET s strategy is its independence from a specific language or platform. This paper is about the strategic decision making that any Small and Medium size Enterprises (SME) should make to adopt a technology platform for a new project. This paper refers to an on ongoing development to provide an integrated business information and e-commerce system for a manufacturing company. The company uses Syspro ERP system. Consumers of ERP systems are demanding solutions that can be easily integrated with Web applications in order to provide such services as e-commerce to customers and browser-based access to remote workers. The aim of this paper is to compare the two technologies and discuss the main reasons why it is believed that .NET would be more appropriate than J2EE as a technology platform for the e-commerce solution.

1 INTRODUCTION

The .NET platform offers powerful capabilities for software development and deploying independence from a specific language or platform. Rather than requiring developers to learn a programming language, programmers can contribute to the same software project, but write code using any or several of the .NET languages (such as Visual Basic .NET, Visual C++ .NET, C#, COBOL, Fortran and others) with which they are most experienced or skilled. In addition to providing language independence, .NET extends program portability by enabling .NET applications to reside on, and communicate across, multiple platforms thus facilitating the delivery of Web services over the Internet. .NET enables Web-based applications to be distributed to consumer electronic devices, such as mobile phones, Handheld PC's and persona1 digital organizers, as well as to desktop computers. The capabilities that Microsoft has incorporated into the .NET platform created new software-development model that will increase programmer productivity and decrease development time (Deitel, 2002).

.NET is also a set of standards, and an operating platform, to enable different applications and organisations to communicate over the Internet, using industry-agreed protocols such as Simple Object Access Protocol (SOAP).

.Net is a set of technologies for connecting information, people, systems and devices over the Internet through the use of XML Web services. In order to create an XML platform there are five key areas of deliverables, which are Clients, Servers, Services, Developer tools experiences and solution. Thus Microsoft adopted the use of XML Web services as one of the key components for its next generation Internet platform .NET, which addresses all of these areas in the .NET Framework.

This paper reports on the ongoing development to provide an integrated business information and e-commerce system for a manufacturing company. The company uses Syspro (Formerly Impact Encore) ERP system.

In this paper we will be comparing the features and properties of .NET with the J2EE.

The reason why J2EE is chosen is because of the fact that it is one of the most widely accepted standards which is similar to the .NET with similar properties. In fact it is the JAVA that has given

Jahankhani H. and Youssef M. (2004).
UNDERLYING PLATFORM OF THE E-COMMERCE SYSTEM: J2EE VS. NET.
In *Proceedings of the Sixth International Conference on Enterprise Information Systems*, pages 250-257
DOI: 10.5220/0002619302500257

existence to the .NET framework in other wards Microsoft's aim is to dominate and replace JAVA with .NET. Therefore by evaluating and comparing the two giant platforms we will have a clear picture of advantages and disadvantages of both of the platforms.

2 INTEGRATING ERP WITH EXTERNAL APPLICATIONS

The manufacturing company uses Syspro solution for its ERP system. E.net solutions is Syspro's very latest response to developer's needs of a bridge between external applications and the core ERP system, (Syspro, 2003). The term e.net coined by Syspro does not refer to, and is totally different from, the Microsoft term .NET. However, the two terms are similar enough to cause confusion. Since e.net solutions utilises a COM interface to allow integration of external applications with Syspro, it follows that applications based on COM technology as well as those based on Microsoft's other more recent technology framework, .NET, would naturally be compatible with e.net.

Recently, Syspro have further increased the ability of their ERP system to integrate with external applications by enabling the business objects to be accessed through Web Services. Developers can therefore now choose to integrate their applications with Syspro either through the original COM interface or by utilising the Web Services.

In order to perform the most basic of its envisaged functions, the e-commerce solution being planned will need to be tightly integrated with the company's Syspro ERP system. However, it is almost certain that the final e-commerce system will be more than just a website that exchanges data with Syspro. Rather, the system will be a comprehensive platform consisting of several components, with each component possibly being required to interact with a variety of different applications in addition to Syspro. Some components may play an indirect role in supporting e-commerce by automating certain offline internal business processes; such components may be required to interact with Microsoft Excel, for example. Other components may play a direct role in e-commerce but may be unrelated to the website aspect of the system; such components may rely on applications such as Microsoft Outlook and Syspro's DFM (Document Flow Manager). While the final form of the complete e-commerce solution is unknown at this stage, one certainty is that maximum benefits will only be achieved if this solution is tightly integrated with the company's overall IT infrastructure rather than just Syspro. Any comparison of .NET and J2EE therefore needs to examine the ability of the technology to enable such integration.

Like most small-medium size enterprises (SMEs), this company's IT infrastructure is based on Microsoft technology and most (if not all) of the software used by the company is either Microsoft software or other Windows-based software. Large companies, on the other hand, usually have a highly heterogeneous IT infrastructure that consists of a wide variety of technology platforms (Windows, Unix, IBM mainframe, etc.). This fundamental difference between SMEs and large enterprises heavily influences their choice of technology (J2EE or .NET) when considering the implementation of new applications that require close integration with the overall IT infrastructure.

Being a small company with a Microsoft-centric IT infrastructure, this organisation fits the profile of a typical .NET adopter. Although this kind of profiling may provide a rough guide as to what technology is generally appropriate for which companies, when making a strategic decision to adopt a technology platform for a new project this kind of general analysis should be supplemented with a more specific analysis of the project. It is important to mention that whether the e-commerce system is based on J2EE or .NET, either choice would not greatly affect its ability to integrate with Syspro.

3 .NET FRAMEWORK

The Microsoft.NET framework is the result of the ASP's team and of the other teams in Microsoft who hunted a better way to create Web applications that would improve speed, provide more built-in system services, improve state management, and separate the HTML (interface) from the script (business rules). Precisely, they were aiming to make it easier to build Web-based applications. Also, they wanted to give developers additional ways to have access to extensibility points within this new ASP framework that would enable more developers (that is, all developers who were not C++ developers) to create functionality similar to ISAPI filters and extensions. Moreover ASP.NET applications offer a greater degree of control; flexibility, and performance than the standard ASP applications (Tabor, 2001).

We all know that every single thing in this world is composed of more than one part or component similarly the .NET Framework consists of the Common Language Runtime (CLR), the Services Framework, and the Applications Framework. The

CLR manages a common data type system, a common metadata structure for describing components, and a common security mechanism, and also manages memory, object invocation, and so forth. The Services Framework is a rich set of libraries and tools developers can use to create applications that work within a managed execution context. The Applications Framework includes tools for developers to create traditional Windows Forms applications and ASP.NET.

The Common Language Runtime (CLR) is the heart of .NET, the foundation on which all the dreams of the programmable Web are resting for Microsoft. It is arguably the most important move forward for Microsoft-based development and constitutes the single greatest paradigm shift yet introduced for the PC. In this section CLR is introduced with discussion on how different or similar to the ASP/COM environment. Code that runs within the CLR are known as "managed code." since there is no IL compiler for Visual Basic 6.0 and Visual C++ 6.0.

The CLR offers a number of major benefits. First, the CLR defines a common set of data types across all CLR programming languages, allowing for multi-language integration. This feature makes it possible to create classes in Visual Basic .NET that derive from classes created in C# and vice versa (Tabor, 2003), (Kennedy, 2002). The CLR enables a development environment with many desirable features.

These include cross-language integration, a strongly typed shared type system, self-describing components, simplified deployment and versioning, and integrated security services. Since the CLR is used to load code, create objects, and make method calls, it can perform security checks and enforce policy as managed code is loaded and executed. Code access security allows the developer to specify the rights or permissions that a piece of code needs in order to execute. For example, permission may be needed to read a file or access environment settings. This information is stored at the assembly level, along with information about the identity of the code, (Kennedy, 2002), (Sullivsan, 2003). The CLR provides code-access security, which is, in essence, mini-checkpoints your code must pass through based on the actions it is trying to perform.

This security measure ensures that unauthorized users cannot access resources on a machine and that code cannot perform unauthorized actions. If the programmers code accesses information or perform actions on the machine it is running on (access files on the hard drive or access the network), it must ask the CLR if it has permission to do that. The CLR at runtime determines the source of the code and where it was obtained from, as well as other information stored in the code's assembly, and grants or denies permission to the code to perform the given task.

In addition to code access security, the runtime supports role-based security. This builds on the same permissions model as code access security, except permissions are now based on user identity rather than code identity. Roles represent categories of users and can be defined at development time and assigned at deployment time. Policies are defined for each role. At runtime the identity of the user on whose behalf the code is running is determined. The runtime checks what roles are associated with the user, and then grants permissions based on those roles.

The CLR encourages moving away from scripted languages compiled at run time to the concept of Managed code. Managed code means a clearly defined level of cooperation between an executing component and the CLR itself. Responsibility for many tasks, like creating objects and making method calls, is delegated to the CLR, which provides additional services to the executing component.

Source code is compiled into Microsoft Intermediate Language (MSIL) that is then consumed by the CLR. In addition to MSIL, .NET compilers also produce metadata. This is information the CLR uses to carry out a variety of actions such as locate and Load class types in a file, Lay out object instances in memory, and resolve method invocations and field references (Tabor, 2003), (Kennedy, 2002), (Sullivsan, 2003).

The CLR defines a common metadata so that components can be self-describing; other words, no separate type-library file or header file is necessary because the class definition, version, and so forth reside within the component or executable file. This metadata is generated from the component's actual source code, so it is never out of sync with the actual executable file.

The metadata that the .Net Framework compliers produce includes class definitions and configuration information. It allows an application to totally describe itself.

In other words, an assembly can be one physical DLL file with one or more classes and resources, or an assembly can have multiple DLL files, each with one or more classes and resources. From the user-programmer’s perspective, all the classes appear to be packaged into the same component.

There are a number of advantages to this approach, including how you can formulate a security policy around assemblies, control versioning in your components (that is, which version of the dependent component should be used if multiple versions are present on the system), and

improve control of the configuration and deployment of your applications.

The assembly's metadata is actually known as the manifest, which includes dependency information, the version of the component, the types that the component supports, and other information.

Since assemblies are self-describing, no explicit registration of your components is necessary. You simply copy your assembly into a directory (designated as a \bin directory) and the CLR takes care of managing the registration process. Delete the file, and the CLR takes care of unregistered the assembly.

This method is in sharp contrast to the registration (and unregistered) process for COM/COM+ components that required you to create an application in COM+/MTS or use the regsvr32.exe command-line utility (Tabor, 2001). Through the use of MSIL and assembly metadata, code written in many languages can be debugged together, with no loss of information or control as we step through Multilanguage projects. An exception thrown by a component written in one language can be dealt with using a component written in another language.

Cross-language inheritance is supported. Assemblies can be made private to an application or can be shared by multiple applications. Multiple versions of an assembly can be deployed on a machine at the same time. Application configuration information defines where to look for assemblies, thus the runtime can load different versions of the same assembly for two different applications that are running concurrency.

This means that installing an application now becomes as simple as copying the necessary files to a directory tree, provided security permissions are satisfied (Tabor, 2003), (Sullivsan, 2003). One of the most important benefits of the CLR is that it provides high-level services to developers that are not restricted to a particular hardware platform or Operating System. Code written to the CLR does not need to be re-written to run on different platforms and under different Operating System One immediate benefit is that the same code can run on all of the various flavours of 32-bit Windows and also 64-bit Windows. It extends the familiar ideas of runtime libraries, template libraries, and the Java Virtual Machine. It means that developers can reuse more code and have to develop less in the way of the more basic services themselves.

The CLR and .NET simplify the programming model and make it more consistent. Initially, the CLR supported only four languages namely Visual Basic .NET, Visual C++ .NET, Visual C# .NET, and Visual J# .NET. There were many companies working on other compilers that produce managed code for languages including COBOL. Pascal and Perl. Now with all the effort put on to the .NET Framework it is capable of supporting more than 24 languages including Fortran, Cobol etc.

The key design goal and features of .NET can be briefly classified as following:

- New memory management, featuring the use of garbage collection rather than reference counting and deterministic finalization
- Just in Time (JIT) Compilation. Various designs were implemented, explored, and refined, including Optimising JIT and Fast JIT models
- Thread and Process Management
- Virtual Object System
- Remoting
- Common Type System (CTS)
- Code Access Security (CAS).
- Code Access Security (CAS)
- Support for debugging and profiling.
- Unmanaged code support
- Simplified, flexible deployment model

4 .NET VS JAVA

Before continuing with this comparison it should be noted that this comparison was conducted without any prejudice to any one platform in particular. We shall not be comparing each and every aspect because if we try to cover almost every aspect we shall not be able to end this comparison. Therefore in order to simplify we are only concentrating on the main issues.

Microsoft .NET is product suite that enables organisations to build smart, enterprise-class Web Services and applications. Microsoft .NET is largely a modification of Windows DNA, which was Microsoft's earlier platform for developing enterprise applications. Windows DNA includes many proven technologies that are in production today, including Microsoft Transaction Server (MTS) and COM+, Microsoft Message Queue (MSMO), and the Microsoft SQL Server database.

But the new .NET Framework replaces these technologies and includes a web services layer as well as improved language support (Vawter, 2003). .NET architecture comprises of the following, (Farley, 2003):

- C#, (C Sharp) a "new" language for writing classes and components, that integrates elements of C, C++, and Java, and adds additional features, like metadata tags, related to component development,
- A CLR "common language runtime", which runs byte codes in an Internal Language (IL) format as discussed in previous section in this

paper. Code and objects written in one language can, apparently, be compiled into the IL runtime, once an IL compiler is developed for the language.

- A set of base components, accessible from the common language runtime, that provides various functions (networking, containers, etc.).
- ASP+, a new version of ASP which supports compilation of ASPs into the common language runtime (CLR) and thereby enabling the programmer to write ASP scripts using any language with an IL binding (Farley, 2003)
- Win Forms and Web Forms are new UI component frameworks for building Windows and Web Applications and are accessible from Visual Studio.NET.
- ADO+ a new generation of ADO data access components that use XML and SOAP for data interchange.

The Java 2 Platform, Enterprise Edition (J2EE) is an industry standard and was designed to simplify complex problems with the development, deployment, and management of multi-tier enterprise solutions. J2EE is an industry standard, and is the result of a large industry initiative led by Sun Microsystems. The J2EE architecture is based on the Java programming language, (Driver, 2002). The process is as follows, (Vawter, 2003).

1. Developers write source code in Java.
2. The Java code is compiled into byte code, which is a cross-platform intermediary, halfway between source code and machine language.
3. When the code is ready to run, the Java Runtime Environment (JRE) interprets this byte code and executes it at the Run time.

In large-scale J2EE applications, business logic is built using Enterprise JavaBeans (EJB) components. This layer performs business processing and data logic. It connects to databases using Java Database Connectivity (JDBC) or SQL/J, or existing systems using the Java Connector Architecture (JCA). It can also connect to business partners using web services technologies (SOAP, UDDI, WSDL, ebXML) through the Java APIs for XML (the JAX APIs), (Vawter, 2003).

5 SUPPORT FOR WEB SERVICES

J2EE application is hosted within the container, which provides qualities of service necessary for enterprise applications, such as transactions, security, and persistence services.

The business layer performs business processing and data logic. In large-scale J2EE applications, business logic is built using Enterprise JavaBeans, EJB) components. This layer performs business processing and data logic. It connects to databases using Java Database Connectivity (JDBC) or SQL/J, or existing systems using the Java Connector Architecture (JCA). It can also connect to business partners using web services technologies (SOAP, UDDI, WSDL, ebXML) through the Java API's for XML (the JAX APIs).

Business partners can connect with J2EE applications through web services technologies (SOAP, UDDI, WSDL, ebXML). A servlet, which is a request/response oriented. Java object can accept web service requests from business partners. The servlet uses the JAX APIs to perform web services operations (Sun Microsystems, 2003). Shared context services will be standardized in the future through shared context standards that will be included with J2EE.

Traditional 'thick' clients such as applets or applications connect directly to the EJB layer through the Internet Inter-ORB Protocol (IIOP) rather than web services, since generally the thick clients are written by the same organization that authored J2EE application, and therefore there is no need for XML-based web service collaboration.

Browsers and wireless devices connect to Java server Pages (JSPs) which render user interfaces in HTML, XML, or WML. `In addition to the specifications, Sun also ships a reference implementation of J2EE. Developers write applications to this to ensure portability of their components. This implementation should not be used for production but rather just for testing purposes.

The .NET application is hosted within a container, which provides qualities of service necessary for enterprise applications, such as transactions security, and messaging services.

The business layer: of the .NET application is built using .NET managed components. This layer performs business processing and data logic. It connects to databases using Active Data Objects (ADO.NET) and existing systems using services provided by Microsoft Host Integration Server 2000, such as the COM Transaction Integrator (COM TI). It can also connect to business partners using web services technologies (SOAP, UDDI, WSDL).

Business partners can connect with the .NET: application through web services technologies (SOAP, UDDI, WSDL, Biz talk).

Traditional 'thick' clients, Web browsers: wireless devices connect to Active Server Pages (ASP.NET) which render user interfaces in HTML, XHTML, or WML. Heavyweight user interfaces are built using Windows Forms.

Multiple language support is one of the key differences between .NET and the J2EE and the greatest feature of the .NET platform that dominates over the J2EE .NET offers language-independence and language-interoperability as stated, the most Intriguing and fundamental aspects of the .NET platform. A single .NET component can be written, for example, partially In.NET version of Visual Basic, and C#, Microsoft's new object-oriented programming language and many other .NET supported languages. The feature of .NET proves to be the key differentiators between .Net and J2EE, mainly points towards Microsoft's real play.Microsoft has two notable things with .NET that It is opening up a channel to developers in other programming languages and many other openings to non-.NET components by integrating XML and SOAP into their messaging scheme. .Net supports development in any language that Microsoft's tools support due to the new CLR . With the exception of Java, all major languages will be supported Microsoft's C# language is equivalent to Java. In addition to this Microsoft have included a refined version or the rewrite of JAVA as J#.NET, which is available as a programming language within VisualStudio.NET environment.

The multiple language support that Microsoft has introduced with the CLR is an exciting innovation for business. Cross language compatibility .NET is enfranchising PERL, Eiffel, COBOL and other programmers by allowing them to play in the Microsoft Programming environment. By using XML and SOAP in their component messaging layer, Microsoft is bolstering their diplomatic face and adding an element of openness to its platform which I suppose will definitely be encouraged by the developers and businesses, (Vawter, 2003).

J2EE promotes Java-Centric computing, and as such all components deployed into a J2EE deployment (such as EJB components and servlets) and must be written in the Java Language.

6 RISK, MAINTAINABILITY AND OTHER REASONS REGARDING LANGUAGE SUPPORT

Many existing systems are internally convoluted. Disrupting such existing systems is a risky proposition as once disturbed it is extremely difficult to put everything back to normal even for the knowledgeable engineers.

We think that a combination of languages running in the CLR may lead to a mess of combination code which is very difficult to maintain. If you have an application written in multiple languages, then to fully develop, debug, maintain and understand that application; you will need experts in different languages. Several languages equates to an increase in Development training expenditures but then again there is an argument that if there is a problem with existing code it can be replaced by any language in which the programmer is proficient.

With combination language code, developers are unable to share best practices. While individual productivity may increase, communication breaks down, and team productivity decreases but again other had it will reduce the cost of development.

Developers using different languages may have very quickly coded .NET system using VB.NET and C#. Skills transfers like VB.NET developers to understand and write code in C# or other code base developers and bring them into .NET standards. The resulting lack in productivity equates in reduced time to market and a higher total cost of ownership. In many other cases, it is much better design to standardize on a single language, and to treat legacy systems as legacy systems and integrate which can be achieved either by J2EE or .NET.

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. Microsoft .NET is based on MTS and Com+, we are concerned that the migration to .NET will be taxing compared to J2EE. It is entirely new infrastructure based on an entirely new code base – CLR. COM+. To accommodate a common type system (CTS) it standardizes on data types used between languages, the original Visual Basic data types have been dismissed (Vawter, 2003).

The COM+ migration path In .NET terminology, code that runs within the CLR is referred to as managed code, while coding running outside the CLR is called unmanaged code.

If one is a COM+ developer and want to take an advantage of the new CLR. There are two methods of solving in the first methid COM+ code needs to rewritten to accommodate the CLR's automatic garbage collection mechanism and its depreciation of pointers. In the other method the existing code has to collaborate between managed and unmanaged code and for that special measures must be taken.

This is another key issue with regard to .NET and J2EE. in this case J2EE definitely dominates the .NET as J2EE is platform-agnostic running on a variety of hardware and operating systems, such as Win32, Unix and mainframe systems where as .Net does not support most of these.

This portability is absolute reality because the Java Runtime Environment (JRE) on which J2EE is based, is available on any platform (Vawter, 2003).

As mentioned above J2EE is a standard so it supports a variety of implementations such as BEA, IBM, Borland, (Pallatto, 2001) and Sun. The danger in an open standard such as J2EE is that if vendors are not held strictly to the standard application portability is sacrificed.

Sun has built a J2EE compatibility test suite which ensures that J2EE platforms comply with the standards. This test suite is critical because it ensures portability of applications. J2EE portability will never be completely free. It is ridiculous to think that complex enterprise application can be deployed from one environment to the next without effort, because in practice organizations must occasionally take advantage of specific features to achieve real world systems. .NET only runs on windows its supported hardware and the .NET environment. There is no portability at all. It should be noted that there have been hints that additional implementations of .NET will be available for other platforms (Vawter, 2003). The future of eBusiness collaboration is undoubtedly web services. J2EE supports web services through the Java API for XML parsing (JAXP). This API allows developers to perform any web service operation today manually parsing XML documents. JAXP can be used to perform operations with SOAP, UDDI, WSDL and ebXML.

Transforming XML-to-Java and Java-to-XML. Parsing WSDL documents and performing messaging such as with ebXML this feature is extremely easy to build. J2EE compatible 3rd party tools are available today that enable rapid development of web services. There are at least sixteen SOAP implementations that support Java. Almost all of these implementations are built on J2EE. There are only five UDDI API implementations four of them support Java (IBM UDDI4J, Bowstreet jUDDI, The Mind Electric Glue and Idox WASP), (Vawter, 2003). The tools that ship with Microsoft.NET also offer rapid application development of web services with automatic generation of web services wrappers to existing systems. Concluding above J2EE you can develop and deploy web services today using JAXP and .NET you can develop web services today using .NET which is much easier to build and deploy. The Sun J2EE product portfolio includes a modular and extensible Java based IDE that pre-date both Sun, J2EE and .NET. Microsoft has always been a strong tools vendor.

There are many implementations based on J2EE architecture that are available for purchase. With many price points varying dramatically, this enables a corporation to choose the platform that fits with their budget and desired service level. 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. The takeaway point is that you can get low-cost solutions with both Microsoft and J2EE architecture.

Scalability is essential when growing a web services deployment over time, because one can never predict how new business goals might impact user traffic. J2EE and .NET allow one to add additional machines to increase user load while maintaining the same response time. The significant difference between J2EE and .NET scalability is that since .NET supports win32 only a greater number of machines are needed than a comparable J2EE deployment due to processor limitations. This multitude of machines may be difficult for organizations to maintain.

7 CONCLUSIONS

Every SME goes through the strategic decision making when they adopt a technology platform for a new project. The decision making becomes more difficult when attempting to integrate business information and e-commerce system through an ERP system. Consumers of ERP systems are demanding solutions that can be easily integrated with Web applications in order to provide such services as e-commerce to customers and browser-based access to remote workers. When attempting to rewrite their software to provide this type of functionality, .NET is the natural choice of platform since most small-medium enterprises that use these ERP solutions have IT infrastructures that are based on Microsoft software.

Amongst SME's .NET development is usually the more appropriate choice. One reason why this is so is that these type of organisations usually have less technical expertise and resources available to them, making fast and simple application development (RAD) a priority. While J2EE development can also provide a RAD route to the implementation of applications, it is nowhere near as effective as that provided by .NET; the advantages of J2EE generally lie in its ability to create highly complex enterprise-scale cross-platform applications, rather than applications that are easy and quick to implement. This is a matter that even advocates of J2EE admit to.

In a comparative study of J2EE and .NET, it is clear that .NET is definitely having command over J2EE when it comes to multiple language support and the development environment.

ACKNOWLEDGEMENTS

This on going research and development is undertaken as part of Knowledge Transfer Partnerships Program.

REFERENCES

Deitel N.: Visual basics .net, how to program, 2nd edition, Prentice hall, UK, 2002.

Driver, M., .net Magazine, April 2002, .NET vs. Java: No Easy Answers http://www.fawcette.com/dotnetmag/2002_04/magazine/columns/strategy/default_pf.asp

Farley J.: Microsoft .net Vs J2EE: how do they stack up?, http://java.oreilly.com/news/farley_0800.html cited on March 2003.

Kennedy A., Syme D.:.Design and implementation of generics for the .Net CLR, www.eecg.toronto.edu/~voss/ece1724f/Slides/DotNet.ppt , November 2002, cited May 2003.

Microsoft: defining the basic elements of .net, http://www.microsoft.com/net/basics/asp, cited February 2003.

Pallatto J.: Borland Takes Lead in Industry Standard Benchmarking of J2EE™ Application Servers, http://www.borland.com/about/press/2001/j2eelead.html, cited April 2003.

Sullivan G. A.: .net, the big picture, http://www.gasullivan.com/whitepapers/, cited on April 2003.

Sun Microsystems' Java website; Java 2 Platform, Enterprise Edition Client Access Services (J2EE CAS) COM Bridge 1.0 Early Access http://developer.java.sun.com/developer/earlyAccess/j2eecas/download-com-bridge.html

Syspro Support Zone, Syspro e.net Solutions Online Documentation: Web Services http://support.syspro.com/user/enet_solutions/index.php?section=Web%20Services

Tabor, R: Microsoft .net XML web services, SAMS, USA, 2001.

Vawter C., Roman E.: white paper, J2EE vs. Microsoft .Net, http://searchwebservices.techtarget.com/whitepapersAuthor/0,293844,sid26_gci829769,00.html, cited April 2003.