

.NET Technology

By

Aruna Paramasivam, Associate
Jeff Gellman, Technology Manager

Topics covered in this paper include:

- The .Net Framework
- .Net Architectural Overview
- .Net: A Platform for Enterprise Web Services
- Deployment Considerations

Overview

This LiquidHub white paper is intended to provide a high-level overview, in non-technical terms, of how .NET technology works, and how it can be employed toward business goals. The vision of .NET is explained, and a summary of the components of the unique .NET framework follows. The value that .NET applications can bring to an organization is detailed, as are a few of the existing short comings in .NET technology. The ultimate goal of this white paper is to provide business decision-makers with a high-level outline of .NET and its capabilities, and to assist readers in determining whether to invest in further examining this burgeoning technology.

What is Microsoft .NET?

Microsoft .NET, or simply .NET, is a term that has stealthily but rapidly moved into software development nomenclature in the last few years. .NET is considered by many to be the next step in the Internet revolution. The .NET concept dates back to 1999 when it was a part of a discussion of Next Generation Windows Services. Before the derivation of .NET, the acronym of choice by Microsoft was DNA – Distributed internet Architecture. While .NET is immersed in this notion, it has by far surpassed its initial conception. To fully exploit the potential of the Internet and the web-based applications it enables, a robust and easily integrated architecture is required. Microsoft has filled this void with the .NET vision. With successful promotion by Microsoft and widespread acceptance by companies, .NET has the potential to become the standard foundation for building powerful yet flexible Internet-centric applications.

The .NET initiative is based on an entirely new architecture in comparison to previous versions of Microsoft tools. It is intended to elevate the development environment to a new level of sophistication, capability and ease of use. .NET provides Internet users with a way of “programming the Web,” giving us a platform for the development of interoperable Web applications. In broad terms, then, .NET can be thought of as software that connects information, people, systems and devices.

Definition

According to Microsoft, .NET is a “revolutionary new platform, built on open Internet protocols and standards, with tools and services that meld computing and communications in new ways.” .NET is billed as a means for “connecting a world of information, people, systems and devices together to radically improve communication and business performance” regardless of the platform used. A more practical way of explaining the concepts behind .NET would be:

- It is a new environment for developing and running software applications
- It provides the framework for an Internet-centric industry standard that makes it possible to tie together applications, information and data sources using Web Services
- It enables software to become platform and device-independent.

- It employs many standard run-time services available to components written in a variety of programming languages
- It has inter-language and inter-machine operability

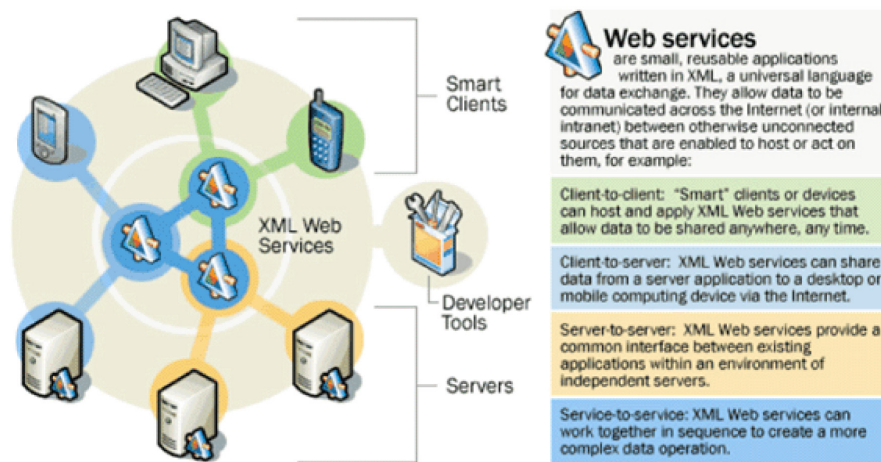
Distributed Computing

Microsoft .NET uses distributed computing, as opposed to the typical two-node client/server systems that are predominant today. Distributed computing is a programming model where processing can occur in numerous points throughout a network. Developers can thus determine the most logical node for processing to take place, whether at a server level, personal computer, handheld device or other smart device. This makes the concept of .NET particularly efficient and creates a true open-architecture environment. Systems that use such distributed computing are known as distributed applications. .NET provides a robust and secure environment for the development and execution of distributed applications.

Web Services

The coordination and execution of processing that occurs in distributed applications is enabled by Web Services. Web Services are units of code that allow programs written in different programming languages and on different platforms to communicate and share data through standard Internet protocols. Such protocols include Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).

In a typical .NET application, SOAP lies above the XML layer. SOAP is an emerging standard for transmitting messages across the Internet. SOAP facilitates application-to-application interoperation between multiple processing points and geographic sites by exploiting the Internet. WSDL provides a means to describe the basic format of these requests, whether the underlying protocol is SOAP, XML or other encoding. WSDL is a key component of what is known as the UDDI initiative to provide directories and descriptions of online services for E-commerce purposes.



Source: Microsoft, Inc.

Figure 1: .NET Web Services

Web Services comprise a platform-independent method that enables applications to communicate and share data over the Internet, as defined by public organizations such as W3C.

Languages Supported

Out of the box, Microsoft currently provides support for the following languages:

- Microsoft Visual Basic .NET
- Microsoft Visual C#
- Microsoft Visual J#
- Microsoft Visual C++ .NET
- Microsoft Jscript

Additionally, third parties provide support for languages such as Cobol, Fortran, Perl, Eiffel, SmallTalk and Python.

For a language to compile and execute within the .NET framework, it must conform to what is known as the Common Language Specification (CLS). Unparalleled language interoperability is available within the .NET framework because these languages adhere to the Common Type System (CTS). These languages all have the ability to support a rich set of data types that are encompassed by CTS.

Platforms Supported

The .NET Runtime is compatible with Windows XP, Windows 2000, NT4 SP6a and Windows ME/98. Windows 95 is not supported. It is important to note that some parts of the framework do not operate on all platforms. For instance, ASP.NET is only supported on Windows XP and Windows 2000. In addition, Windows ME/98 cannot be used for development purposes. Furthermore, Internet Information Services (IIS) is not supported on Windows XP Home Edition, and therefore cannot be used to host ASP.NET. However, the ASP.NET Web Matrix server does indeed run on the Windows XP Home edition. This opens up the environment to ASP.NET users. (See www.microsoft.com/iis for more information on this subject)

.NET Tools

The following tools can be used to develop .NET applications:

.NET Framework SDK

This SDK is free and can be downloaded from the Microsoft site. It contains compilers for C#, C++, VB .NET and several other utilities that are useful for development.

ASP .NET Web Matrix

This is a free ASP .NET development environment provided by Microsoft. This can be particularly advantageous for users of XP Home Edition, since it opens up the ASP .NET environment to those machines, which as mentioned above, cannot run IIS. The ASP .NET Web Matrix is available on the Microsoft site. The download includes a GUI as well as a simple web server that can be used instead of IIS to host ASP .NET applications.

Microsoft Visual Studio .NET

Microsoft Visual Studio .NET includes support for all the Microsoft languages such as C#, C++, and VB .NET, with extensive wizard support. MS Visual Studio .NET constitutes the core of .NET development. It provides a complete development platform to design, develop, debug and deploy .NET applications and Web Services. It comprises tools for building Web and Windows applications as well as Web Services and also has a full set of data access tools. There are also complete error -handling features available for local debugging, remote debugging and tracing. Visual Studio .NET is available in Standard, Professional, Enterprise Developer and Enterprise Architect versions.

In general, Microsoft Visual Studio .NET should be the default means for developing and executing .NET applications. It greatly simplifies the process of creating Web Service client- or server-applications and provides built-in support for the entire .NET development process.

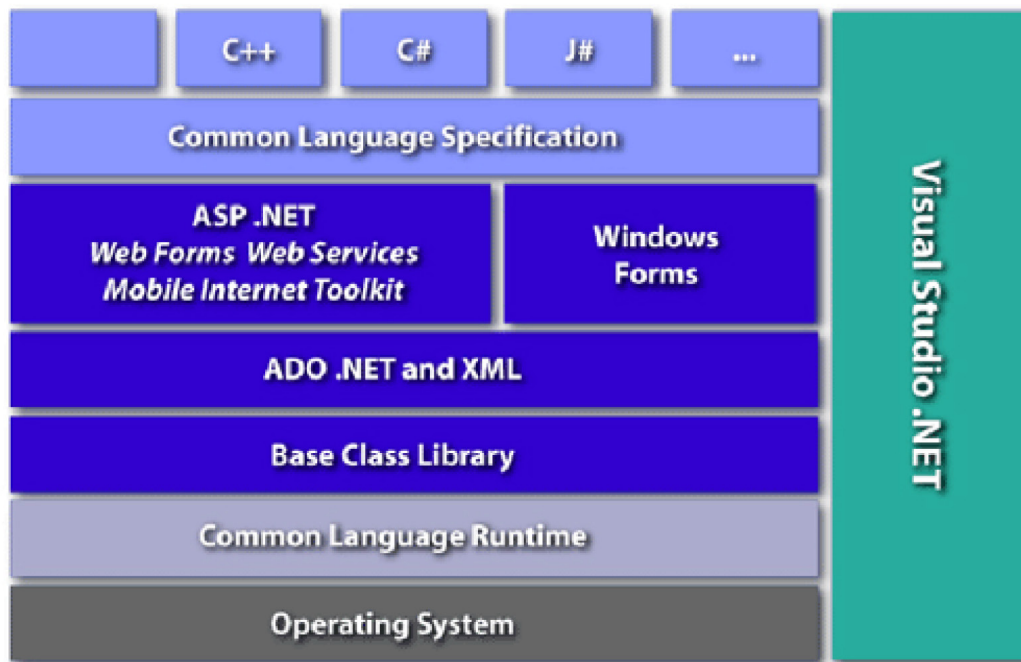


Figure 2: .NET Framework and Visual Studio .NET

The .NET Framework

The .NET framework provides the foundation for developing, deploying and running Web Services and .NET-based applications. It is a standards-based, multi-language application execution environment that provides the necessary compile-time and run-time basis for building and running these Web Services and .NET-based applications. The following paragraphs describe the major components of the .NET Framework.

Common Language Runtime (CLR)

The CLR executes programs compiled from any Microsoft .NET compiled language. It is a controlled

execution environment that not only compiles and runs code, but also displays bugs and handles runtime services such as language integration, security and memory management.

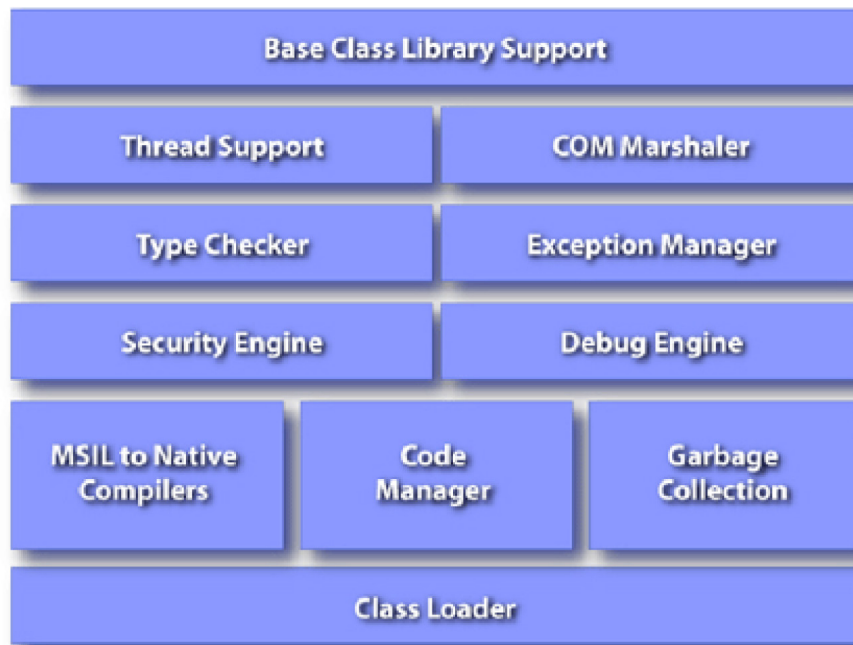


Figure 3: Common Language Runtime

In addition, the CLR also provides the mechanism for sharing code between systems. This minimizes conflicts between applications by enabling incompatible software components to coexist. Multiple versions of program libraries can co-exist on one machine as a result, thereby preventing what is commonly known as “DLL hell.” .NET applications are compiled and built on one platform. The bulk of the functions performed by the CLR are done transparently, thereby simplifying the work of IT administrators. The CLR is sometimes referred to as a managed environment. The code executed and managed by the CLR is called managed code. Managed code supplies the information necessary for the CLR to provide services such as memory management, cross-language integration, code access security, and automatic lifetime control of objects. This managed code allows different programming languages to integrate fully with each other. Such integration is available as all managed code in the CLR is compiled down to an object-oriented intermediate assembly language called Microsoft Intermediate Language (MSIL).

As a modern, object-oriented assembly language, MSIL has instructions to define new classes, create objects, call methods and access properties. Therefore, when individual languages provide syntax to perform any of these tasks, the source code is compiled down into MSIL containing special assembly instructions. These commands, in turn, are compiled by the Just-In-Time (JIT) compilers into calls to specific CLR services. These calls are what handle all common class and object creation, method invocation and garbage collection services among other functions. The end result of this process is that all .NET languages’ classes and objects are ultimately represented in memory in a common manner. This commonality opens the door to allowing a class in one .NET language to inherit a class developed in another .NET language. Inheritance across languages allows for true software reuse. Hence, the hundreds of built-in and fully documented classes within the CLR are immediately reusable and extendable from any .NET-compiled language.



Figure 4: .NET Framework Class Library

The CLR, however, does not interpret this MSIL directly. Instead, the CLR loads this code on demand and natively compiles it into machine code in just-in-time fashion before it is run. This approach, combined with other techniques such as caching of compiled code, is what enables .NET to execute at native speeds at all times. The interoperability fostered by the environment-independent model also aids in enabling the feature of distributed computing, generating the open architecture of the .NET environment.

Class Libraries

The .NET Framework Class Libraries provide reusable code for common tasks such as data access and web service development, as well as Web and Windows forms. The Libraries supply the resources needed to build applications with data access, as well as web server and networking features. Framework classes can be used as a model to illustrate how classes in the class libraries are reusable and extensible. All .NET programming languages use the same framework classes. For example, multiple programs can use the same Console class to display information on the screen. The console class is just one of many classes within the System namespace. The System namespace contains many low level classes for mathematical manipulation, garbage collection and other such functions. There are also a number of sub-namespaces within "System" that are name-spaces in their own right. Here are a few examples:

- System.Collections: includes basic container classes such as ArrayList, SortedList, Stack and Queue
- System.Diagnostics: includes classes for tracing and debugging
- System.IO: holds classes for file IO such as Stream, MemoryStream, FileStream, Path and Directory
- System.NET: includes classes for network programming such as Socket, DNS, IPAddress, Connection and HttpWebRequest
- System.Security: includes classes dealing with permissions and cryptography
- System.Threading: holds classes like Thread, ThreadPool, Mutex and Timeout

- System.Web: includes classes that facilitate Web development
- System.Windows.Forms: holds classes needed for Win32 development such as Menu, Tooltip, Control, Button, Data-Grid and ScrollBar
- System.XML: provides classes for interacting with XML data

Security of the .NET Framework

The .NET Framework facilitates a wide range of flexible security options that can be implemented by developers, administrators and users. The following is a high-level overview of the security features provided by the .NET framework.

Role-Based Security

Role based security grants access and privileges based on the user's assigned roles and functions within an application. Within the .NET environment, this is done via authentication and authorization. Credentials such as user name and password are examined and when the user is authorized, authentication is accomplished. After this is performed, application code then determines the role of the user and level of access and permission is allocated to the user.

The .NET framework supports the following authentication protocols:

- Basic
- Digest
- NTLM
- Kerberos
- SSL/TLS client certificates

Authorization can be done in several ways in the .NET environment. For instance, XML can be used to designate which users can access certain URLs. Custom authorization checks can also be created. Or, to simplify further, developers can simply tap into existing Windows authorization mechanisms. These authentication and authorization methods can be combined in numerous ways to build powerful yet flexible security within the distributed computing model of .NET.

Evidence-Based and Code Access Security

Evidence-based and code access security are exemplified by:

- Code residing in a particular code directory
- Code coming from the Internet or Intranet
- Code bearing a certain hash value
- Code signed with a certain key

The type of code access security denotes what resources users within specific domains can access. This layer of security provides administrators with additional granular control over the machines within their domain. This additional security overlay ensures further protection from potentially dangerous code.

Cryptography

Within the .NET framework, functions for encryption, digital signatures, hashing, and random number generation encompass cryptography. The following algorithms are supported by the .NET framework:

- Symmetric encryption such as DES, TripleDES, or RC2
- Asymmetric encryption such as RSA or DSA
- XML digital signature specification and hashes such as MD5 or SHA1

These core security components, when combined and layered, provide a great degree of flexibility in the various levels of security available to meet the different and varied needs of each .NET application.

Value of .NET Technology

After examining the .NET Framework, it becomes clear that there are numerous potential technical advantages to adopting .NET technology.

Increased Application Reliability

The .NET framework provides improved reliability for development and execution of code thanks to automatic version control of components and also the ability to isolate applications from one another. As a result, there are no conflicts between applications and associated components. Web services and web applications built on the .NET framework are capable of automatically detecting and recovering from errors such as deadlocks and memory leaks to further ensure application availability to all users.

Better Performance

The .NET environment is able to provide better performance due to the advanced compilation and caching techniques supported by the CLR. According to Microsoft, organizations that have moved from ASP to ASP .NET have seen significant increases in speed. In some cases, this has even been in the range of 300-500% increase, which is obviously quite substantial, especially where quick data access and information transfer are key—as they almost always are.

Higher Level of Flexible Security

.NET provides a variety of security options for application directories, local environment variables, databases, network servers, and printers. Evidence about origin and authorship of code and components can be collected via the .NET framework. The runtime environment can then combine that evidence with administrator-set or default security policies to make fine-grained decisions about whether to run that application or enable it to access a particular resource. Using a combination of authentication, evidence-based security and cryptography, policies can mandate that only applications originating from a particular location or bearing a particular digital signature or Authenticode publisher signature can access certain resources and perform certain operations.

Integration and Interoperability

The .NET framework fully supports existing Internet technologies, thereby allowing for complete interoperability between HTML, HTTP, XML, SOAP, XSL, Xpath and other web standards. .NET applications can thus integrate seamlessly with one another. With the use of open standard Web Services, data can

be transferred between a wide variety of client devices, such as various computer operating systems, cellular telephones, personal digital assistants (PDAs) and even game consoles. This level of integration is potentially advantageous from both an internal and external perspective. Internally, applications with external sources of data and information can still be integrated within networks by means of the .NET framework. Externally, .NET technology can be used to integrate the products and services of an enterprise into the business of its key clients or favored suppliers and vendors because of its platform independence and ease of integration.

Decreased Programming Effort

Programming is generally easier within .NET, primarily because of the self-describing property of the CLR. All code written for the .NET environment is delivered in packages. These packages are called assemblies—one or more .EXEs and/or DLLs that represent a logical set of functionality. These files also include all the metadata information to completely describe their contents. The CLR itself, as well as development tools, can read this information for all sorts of purposes, from determining file dependencies, to verifying versions, to automating the generation of client code to access the internal objects. CLR system classes can enable developers to interrogate assemblies at runtime and query the component's capabilities. Objects can also be called dynamically from the scripting environment if needed. Furthermore, the metadata system is extensible, allowing developers to add their own custom information to assemblies through user-defined attributes. The metadata system also allows developers to write tools to later extract that information for their own purposes. All these features significantly decrease the programming effort involved in the .NET environment, providing increased productivity and allowing for simpler, quicker application development.

Increased Developer Productivity

The .NET framework supports the integration of over twenty languages. Thanks to the multi-language support of .NET, experienced developers do not have as steep a learning curve as may be presented by another new technology. Components supported by .NET, even when written in different languages, can continue to integrate with one another easily. This allows developers to choose the language with which they're most comfortable, thereby increasing individual developer productivity and allowing the creation of development teams with varied experience and programming language capabilities. The programming model within .NET is fairly intuitive. There is a large amount of code available in the class libraries for reuse. The .NET framework transparently handles features such as memory management without additional effort required by the developer.

Easier Deployment

To install and deploy a Windows-based application, there are typically several registry settings, files and shortcuts that must be created. However, these steps are completely eliminated in the .NET environment. .NET components are not referenced in the registry. Applications can be deployed to a client or server by simply copying the application directory to the target machine with no registration necessary. To uninstall a .NET application, only the application's directory tree needs to be deleted. In addition, Windows-based smart client applications can now be deployed to and up-dated on specific personal computers by copying the necessary components to a Web server that can be accessed by the target end users.

Easier Administration

.NET applications are highly manageable. Application configurations are contained in XML files, which

can be edited and updated using any text editor. Scriptable command-line tools and snap-ins are also available, and they're very easy to use and execute.

Business Advantages

To step back to the broader perspective of business goals, there are a number of economic and strategic advantages driving the adoption of .NET technology and standardized Web Services. Several of the technical benefits described above, such as higher level of security and increased reliability, also present clear business advantages.

Accelerated Life Cycle Development

Applications can be developed quickly with .NET. Developers can focus on the needs of the applications and eliminate mundane tasks such as memory allocation since this is automatically taken care of by the .NET framework. Code can also be reused easily from the class libraries saving much development time. Furthermore, beyond the advantages of object-oriented design, developers and business also benefit by spending less time debugging when involved with a .NET application. The CLR within the .NET framework supports a consistent exception-based error handling system. All errors in .NET code are handled the same way. This allows for code in one language to throw an exception and have the exception caught in another language.

In a .NET environment, business changes can be implemented more quickly within applications and with much less programming effort and dependencies tracking. An example of how businesses can benefit from .NET development and integration is Newport News Shipbuilding, which used .NET-connected software to build an application to connect with partners. The company reported that it improved its time to market by 19 percent by building and launching its new application with .NET technology.

Lowered IT Cost

Due to accelerated lifecycle development, and developers being able to use their existing skill sets for programming in the .NET environment, grand-scale corporate IT costs can be reduced on many fronts. With .NET speeding up implementation of applications, there is a measurable drop in development costs by day. The ability to use existing Web Services also presents substantial savings, as will the opportunity to use present development efforts for future business-driven changes. There is generally no significant cost in retraining or hiring new developers when a .NET project is undertaken. Programmers can rely on much of their existing skill-sets since .NET supports multiple programming languages.

Integration for Better Service and Lower Operating Costs

.NET technology makes integration between applications, the Internet and sources of data easier because it is platform and device-independent. Using a standard framework reduces the cost, time and expertise required to tie together separate applications. The .NET framework facilitates a Web Services environment, which presents a standard framework within which applications can share data. Consequently, information can be made available to any application or device type. Application-to-application transactions can use an Internet-based communication link, which provides organizations with greater flexibility to conduct their businesses. The ability to connect systems can have a significant impact on a company's ability to execute its core competencies, whether the need exists to connect a handful of internal applications or to integrate with an extensive outside supply chain. For example, with .NET services, a company's order management system might now directly interface with a web-based credit verification system. Thorough

integration between these two systems provides customers with quicker and more reliable transactions resulting in better overall service and customer satisfaction.

In other cases, many smaller suppliers have never taken to Electronic Data Exchange (EDI) methods. Many companies continue to use isolated computer systems and still rely on phone, fax and email methods. However, with .NET connected software, their communication gaps can be bridged easily to lower overall operating costs. Business and supply chain transactions can often be conducted so much more efficiently with .NET integration that the ROI on development is nearly immediate.

Increased Mobility Support

With the .NET standard, data can be easily transferred to field workers via mobile devices using a variety of communication methods. Such devices integrate directly with applications and data, reducing the amount of time and effort required to obtain and transfer information. Increased mobility support can present significant value depending on the type of business operations being supported. For example, sales staff can now have easy access to data that may have been contained previously in isolated back-end systems. With the use of familiar operating systems such as Windows XP, which has .NET capabilities, information can be easily transferred to a wide range of smart devices. This allows field personnel to analyze data they may require while still in the field, allowing employees to focus on their primary roles rather than travel and administrative tasks.

Easier Web Development

Using ASP .NET, an application can be transformed into a Web service with just one simple line of code. This is made possible because ASP scripts are morphed into ASPX files and IIS acquires a new ISAPI DLL to run these files. This in turn allows both ASP and ASPX files to be handled on the same site. With Web Forms, the foundation of ASP .NET, the HTTP code that displays the page and application code can be physically separated. This makes for much simpler maintenance and easier code reuse. .NET thus allows for quick development as well as easier maintenance of web-based applications.

Quick Value Obtained via Out-Of-The-Box .Net Compatible Solutions

Microsoft is providing several .NET connected applications that serve to provide out-of-the-box business functionality. Without requiring that businesses develop systems from scratch, these applications can be integrated into extant systems to provide .NET connectivity. For example, Microsoft Office XP, Microsoft's Office suite of desktop applications, is now .NET compatible. Office XP applications can now be integrated quickly and easily into other .NET-connected applications and services. Microsoft Great Plains Business Solutions, which is a common business management and accounting package, is also now .NET compatible. The benefits provided by .NET can quickly be gained through the use of these of out-of-the-box packages-for quick implementation and seamless integration.

Microsoft MapPoint .NET is another example of a fully .NET connected application. An XML Web service provides the functionality for the .NET connection application to integrate dynamic mapping and location services. One way to integrate this application effectively would be to use it within a sales force automation (SFA) package that requires access to location and mapping information. This would enable sales representatives who may be making in-person calls in various territories to be connected to the information they need for travel and communication using a wide variety of devices.

Another useful .NET connected application is Microsoft .NET Passport2, which is currently available at no cost. .NET Passport is a web-based service that provides authentication and verification service to consumers. A user may register with Passport and provide information such as name, address, email and other data that they would normally submit during any web-based transaction. This user can then automatically share this information with any other .NET passport connected site without having to re-type and re-share this information. Microsoft .NET passport has a strong security policy and enables users to maintain a personal profile online safely. Thanks to the provision of such basic software functions in readily available forms, quick value can be obtained by using Microsoft .NET solutions.

Current Issues With .NET

As with any new technology, (having just been released in 2001) .NET is still evolving. There are still a couple of areas where .NET falls short. It should be noted that Microsoft has shown a strong commitment to improving and providing enhancements to existing .NET technology in many ways thus far. They have released compilers for a range of languages and have submitted the CLR specification to open body standards to broaden the horizons of languages that are compatible with the .NET framework. .NET supports the use of other non-Microsoft databases such as Oracle and DB2. ADO .NET can be used to link a database of choice with .NET applications; drivers are provided out of the box for Oracle and IBM DB2.

.NET has also garnered a substantial level of industry dedication. Independent technology developers and value-added resellers all have vested interests in establishing a Web Services standard. It appears to be well understood in the industry that if the .NET standard emerges and is successful, then enterprises all stand to gain from an ever increasing span of capabilities, services, functionalities and features to add value to their businesses. For the organizations with limited financial resources and technical expertise, the option remains where they may gradually adopt .NET functionality as feasible, while leaving other applications operational as they are.

Lack of Linux/Unix Direct Interoperability

There is no direct interoperability with .NET and Linux or Unix. However, this constraint may soon change since Microsoft has submitted the CLR specification to open standards bodies. Presently, interoperability between .NET programs and Linux/Unix based applications is still possible using Web Services as a link.

Migration of Existing Applications

Within Microsoft languages, there are many discrepancies between past versions and what .NET uses. In the past, Visual Basic has been a popular choice of programming language for business applications. Many systems have also been written in Visual C++. However, for the .NET environment, new versions of both these languages have been created such that these older versions cannot run without changes. For example, the memory protection that is incorporated in the CLR makes it difficult for C++ applications to be moved into .NET without significant changes. As for VB .NET, there are numerous changes in syntax, which need to be made. VB developers also need to adapt to static classes that are not instantiated as opposed to the ordinary classes they are accustomed to.

Gartner Group analyst Michael Ricciuti stated on CNET News.com (Oct 2002) that “a surprisingly large amount of change” is usually required to move existing applications into the .NET framework. The Gartner Group has concluded that only 40% of existing code written using Visual Basic 6 can be migrated to Visual Basic .NET without “substantial redesign and coding.” In some cases, it can be very difficult to use .NET technology within existing systems and where application migration is required. Significant development

efforts, which consequently result in high costs, can be required. In such cases, developers will need to re-write code completely, although no features will be gained. In addition, these programs have to be tested thoroughly again for due diligence before implementation, resulting in significant quality assurance efforts before complete deployment. The return for this cost, time and effort equates to a more efficient, effective, secure and reliable application that takes advantage of the numerous benefits of the unique .NET framework.

Conclusion

.NET has been designed as a technology architecture and development initiative, but it can also be considered a business strategy. Companies providing .NET connected applications have the technical and functional incentive to open up their applications and permit transfer of data via Web Services across the Internet. .NET allows business leaders to focus on solving problems and creating opportunities rather than on managing software programming efforts. The .NET framework takes away much of the “plumbing and maintenance” generally associated with software development and condenses the work of actual programming.

.NET applications simplify the task of developing and integrating complex application environments. Using the concept of distributed computing, .NET ushers in a new open architecture model, with the Internet as its core. With the elimination of platform and device dependence, .NET brings about seamless integration and direct connectivity to computer, applications and handheld devices. Thus, data can easily be accessed, transmitted and acquired as needed throughout enterprises. .NET allows businesses to use the Internet as a means of worldwide communication by presenting a robust and stable open architecture for the development of web applications. It provides a new level of reliability and security that was previously unavailable in Internet-reliant applications. It should also be noted that .NET is still a fairly new technology which was only announced in 2001 by Microsoft as a vision. Hence, not all associated problems and issues that may be associated with it may have been located. Microsoft has made a commitment to .NET and is dedicated in its promotion of this technology. Other partners and resellers in the industry are also eager for the establishment of a standard framework for applications to share data since it is clear there are numerous business advantages associated in adapting .NET applications. .NET provides another avenue for the innovative enterprises, looking to focus their expertise on meeting the new accelerating demands of business, with the financial resources and technical expertise to undertake a new venture.

References:

www.microsoft.com/net
www.gotdotnet.com
www.dotnetjunkies.com
www.dotnetextreme.com
www.devcity.net/net
www.dotnet247.com
www.dotnetdan.com

Notes

- 1 – See www.microsoft.com/mappoint/net/ for additional information
- 2 – See www.passport.net for additional information

About the Strategic Technology & Advancement Research (STAR) Team

The Strategic Technology & Advancement Research Team serves as the proving ground for our ideas.

STAR brings together the best and brightest of LiquidHub to encourage new thinking and develop through leadership around technology. Breakthrough ideas emerge and are exercised by the Team, to strengthen client engagements and educate industry leaders.