Understanding Microsoft's .NET

Announced at Microsoft's Forum 2000, Microsoft's .NET platform introduces a radical new way of developing Internet applications. By basing the .NET vision on open standards, Microsoft has finally admitted that an all-Microsoft shop is unlikely.

By Simon Bisson
Internet Consultant

The overpowering humidity of a Florida summer isn't really the best place to introduce a new set of development tools, but over 6500 developers recently gathered in Orlando for Microsoft's 2000 Professional Developers' Conference. Normally a many-tracked conference covering the whole range of Microsoft's products and operating systems, and the interests of a wide cross-section of developers working with Microsoft technologies, this year's event was different. Instead it was the first public appearance of the technologies that form the building blocks of June's Forum 2000 announcement of .NET.

With senior staff admitting that they're betting the future of the company on the .NET platform, it's clear that it is intended to be the foundation of the entire next generation of Microsoft's software - from operating systems to applications, servers and games, and even to the flagship Office suite. However, despite its wide ramifications, the basis of .NET lies in the Internet. As Paul Maritz, Group Vice President, Platforms Strategy And Developer Group, describes it, .NET is an evolution from the single Web site model to what he calls "Web services".

Web Services

Web services does not mean your typical online application, like Amazon.com or your current ASP of choice; instead, XML data connections link applications and services over the Internet, as well as enabling the use of "smart" clients rather than Web browsers. Developers and enterprises will find that the .NET framework is nothing but a set of technologies designed to turn Windows into an Internet-based distributed computing platform, with integral Enterprise Application Integration (EAI) capabilities through its use of the SOAP (Simple Object Access Protocol) XML communications technology developed by Microsoft, Userland, DevelopMentor and IBM.

Anyone wanting to start work with .NET will have to wait until the next release of Microsoft's development suite, Visual Studio. Previously known as Visual Studio 7, it will instead be given the .NET suffix - like all the services and applications Microsoft will be launching as part of this initiative. Together with a set of server suites, including new electronic commerce tools, an XML-aware version of SQL Server and a set of building-block Web services like the existing Passport distributed Web authentication system, Microsoft intends to use Visual Studio.NET to introduce a wide range of new technologies, including the much trumpeted C#, a new C++ derived language. Visual Studio.NET should enter public beta shortly, with a final release due in the summer of 2001. An early version was released at the PDC, and part of it, in the form of the .NET SDK technology preview, can be downloaded from Microsoft's MSDN Web site.

One of the biggest changes in Visual Studio.NET is a new multi-language strategy. Not only does Microsoft now have a single development environment for all its own languages but, with the introduction of the Common Language Runtime (the CLR), Visual Studio.NET offers a framework for third parties to develop languages that can be used through the familiar Visual Studio IDE. Languages announced at the PDC included new versions of ActiveState's Perl and Python implementations, as well as implementations of COBOL and the functional programming language Eiffel. With Visual Studio.NET and the new ASP+ Web application delivery tool, you're now able to create Web applications in COBOL without having to switch development platforms - if that's what you really want to do.

Common Language Runtime

The CLR is the heart of the whole .NET framework, and is critical to its success. One of its more important features is its single object framework, which allows multiple languages to work together. Using the CLR, a Visual Basic.NET program can call a C# component that itself works with objects written in Eiffel or COBOL - or Perl or Python or Java. The single object framework even makes it possible for objects written in one language to inherit from another, as the CLR provides a strongly typed environment that third-party developers can build languages on ready for use within the Visual Studio IDE, and a common object model that means that all components utilising the CLR inherit from the same base classes. With Web application development today meaning multi-skilled development teams are working on large component-driven application architectures, these technologies will give overstretched project managers the ability to use their developers most effectively.

By making the CLR the key to application development in Visual Studio.NET, Microsoft has also changed the way its development tools compile the resulting code. Where Visual Studio currently uses a conventional compilation to produce processor-dependent binaries (along with the usual megabytage of support libraries and runtimes), things are very different in Visual Studio.NET. If you've worked with Windows CE, you'll be familiar with the Common Executable Format developed to deal with the many processors used in Windows CE devices; it is compiled at runtime by the Windows CE operating system. Visual Studio.NET takes a similar approach by compiling to an Intermediate Language (IL), which is suitable for delivery to any system that supports the CLR. The final compilation is carried out using a JIT compiler at runtime, with two possible compilation modes: a slow compilation that produces optimised code, and a fast "dirty" compiler. The IL code is produced no matter which language you use - unless you choose to produce "unmanaged C++", which still compiles directly to binaries.

C#

Looking around the Net, it's clear that one of the more controversial components of Visual Studio.NET and the whole NET framework is the introduction of a whole new language, C#. The latest member of the C/C++ family, C# has been designed by a team led by Anders Hejlsberg (who created Delphi for Borland), to create a C-style language focused on the creation of software components. As part of Microsoft's COOL project, C# has been seen as Microsoft's response to the ongoing Java legal action with Sun, which has stopped it producing a Java development environment. It's true that C# does contain some similar features to Java - including garbage collection. However, the development of C#, which has been going on longer than Sun's lawsuit, is best seen in the light of the overall distributed computing model at the heart of .NET, as a language designed from the ground up to create Web-accessible services and components.

"With the introduction of the Common Language Runtime (the CLR), Visual Studio.NET offers a framework for third parties to develop languages that can be used through the familiar Visual Studio IDE."

As Hejlsberg explained in his PDC presentations, everything in C# is an object, enabling complex components to be developed quickly. By including attributes in object definitions, C# programs are able to take advantage of specific Visual Studio.NET compiler functions and external libraries built into the CLR, including delivering object calls as SOAP interfaces. To counter fears that C# is a proprietary language, Microsoft has also offered the C# language and the CLR to the Swiss-based ECMA standards body, turning it into an open standard handled by the same neutral organisation that manages the development of JavaScript.

Commitment To XML

Since Microsoft included an XML parser in Internet Explorer 4.0, it's been clear that it has made a large commitment to XML. The development of the BizTalk XML-based EAI tool was another sign of XML support - as was the recent display of a custom version of Visio that produces XML schema for BizTalk. However, the most important part of Microsoft's XML strategy for .NET is its work on the open SOAP standard alongside Macintosh content management specialists Frontier, a process recently joined by IBM. The Simple Object Access Protocol is a simplified form of the more complex RPC standard, redesigned for use over http

connections. This allows remote objects' methods to be called and the results delivered over a common protocol. As http is a connectionless protocol, a SOAP application is dependent on advertised services and on a reliable network - there's no point in including SOAP components in an application designed to run over a mobile phone connection without using assured message delivery tools like IBM's MQ Series or Microsoft's MSMQ.

As a result of this commitment, Visual Studio.NET has been given the ability to automatically create XML descriptions of the available SOAP connections on any Web service, using the SOAP Contract Language (SCL). An SCL document is an XML description of a set of objects, together with the available methods - and the URLs used to call them. A developer importing an SCL description into Visual Studio will find that the IntelliSense auto-completion features are aware of the Web services imported into the IDE, ready for their use in your code.

Error Handling

One key issue when working with SOAP is that there are no automatic error-handling features in the CLR, so you will have to develop error-handling code of your own to cope with network problems or failures in remote Web services. As .NET moves more and more developers into creating distributed applications, good error-handling techniques will become more and more important. Visual Basic.NET will include new constructions to help create error-handling code, using the familiar try-and-catch exception handling used in Java. Web services will add new administrative issues for companies wanting to include them in their applications. Before using SOAP connections to third-party services, including Microsoft's planned .NET building blocks like Passport, organisations will need to treat its relationships with Web service providers as it would with any ASP, including the negotiation of service-level agreements - making sure that the legal framework for trusting key components of business-critical applications to third parties is in place before any development begins.

New Way Of Working

Working with .NET is going to introduce new complexities as quickly as it provides solutions for age-old distributed computing problems. As companies begin to start using Web services to provide both internal and external user interfaces, the ability to develop Web front-ends for applications quickly and effectively will become critical. Previously Microsoft's Web development technologies were firmly based around FrontPage and Visual InterDev, using VBScript in Active Server Pages (ASP) to include business logic in Web applications. This could be a slow process, as designers and developers had to work closely together, making sure that code and design were able to work together.

"The underlying vision of .NET - a smart network made more powerful by smart clients - is a compelling one, especially in these days of electronic business and Web-mediated transactions."

.NET is intended to reduce these tensions and to speed up development of Web interfaces through ASP+. This finally allows developers to separate design from their business logic with a new way of working, closely related to Visual Basic 6.0's Web classes: Web forms. By allowing design and code to reside in separate files, and by basing ASP+ on the .NET CLR, this also means that Web application developers can work in languages other than VBScript or JScript, as compiled languages will replace the awkward ASP server-side scripts. There's no need to worry about having to update existing ASP applications, as these can run in parallel with ASP+, reducing the need for conversions. However, you're likely to want to change them as soon as possible - even if it's just to take advantage of Web forms.

Further Information

www.microsoft.com/net/
msdn.microsoft.com/vstudio/nextgen/default.asp
www.ecma.ch/
msdn.microsoft.com/net/
www.activestate.com

Mobile Solutions

The videos on the Forum 2000 Web site make it clear that Microsoft's .NET vision also entails a world of mobile users, with PDAs and mobile phones as standard access devices. With the US rapidly catching up with Europe and Asia, and with the 3G global standard just around the corner, mobile communications and applications are going to become more and more common. If you're going to be using Visual Studio.NET to develop mobile solutions, Microsoft provides you with a derivative of the Web forms technologies: mobile forms.

Mobile forms are designed to allow device-independent application development, so the same mobile form will work with a Nokia 7110 and a Palm PDA, as it will deliver content based on the type of device accessing an application. There's a possible problem here, as this approach does require Microsoft to gain a deep understanding of the various characteristics of the many different mobile devices that will be rolled out over the next year. If you consider the deep rivalry between Microsoft and companies like Palm and Symbian, it's clear that this could be difficult to achieve. However, all is not lost, as it will be possible for developers to create their own device profiles for use with mobile forms.

Conclusion

We're going to have to wait some time before we can start producing large-scale .NET applications. The next year's release of the Visual Studio.NET development environment will allow you to create new distributed cross-platform applications built around the .NET framework, but full-scale implementation of Microsoft-centric .NET services will have to wait for at least one generation of the BackOffice server suite, and at least two generations of Windows. This is because it won't be until the Backcomb release of Windows 2000 that .NET will be supported fully at an OS level, along with a new user interface, and thus the full .NET environment will not be available until well into 2002 (and maybe 2003). So actual deployment of the full .NET vision is unlikely to occur for another year or so after that - taking us to 2003 or 2004. The sheer volume of new tools, technologies and ways of working in the .NET framework will make uptake a slow process, and the learning curve will be particularly steep if you've not yet begun working with XML and Web technologies.

If you want to start work with .NET then you can either download the SDK from the MSDN site, or wait until Visual Studio is released some time in the second half of 2001. There's a lot in .NET, especially with the XML technologies. These XML services will make it attractive to businesses wanting to use the Web services model enabled by SOAP for enterprise application integration of existing legacy applications and hardware. If you want to get a head start on using SOAP with your existing applications and systems, there's also a toolkit that adds SOAP capabilities to Visual Studio 6 on the MSDN Web site, or various other online SOAP resources that offer tools for other languages - including Perl and Java.

The underlying vision of .NET - a smart network made more powerful by smart clients - is a compelling one, especially in these days of electronic business and Web-mediated transactions. By basing .NET on open standards Microsoft has finally admitted that an all-Microsoft shop is unlikely, and that companies will use the systems they are happiest with, delivering networks that mix Windows, Unix, mid-range systems and mainframes. In this world .NET is just another glue technology, as well as the future of Windows. Microsoft says it is betting the company on .NET, and they must think it's a bet they can win.

"The most important part of Microsoft's XML strategy for .NET is its work on the open SOAP standard alongside Macintosh content management specialists Frontier, a process recently joined by IBM."

PCNA

New Reviews from Tech Support Alert

Anti-Trojan Software Reviews

A detailed review of six of the best anti trojan software programs. Two products were impressive with a clear gap between these and other contenders in their ability to detect and remove dangerous modern trojans.

Inkjet Printer Cartridge Suppliers

Everyone gets inundated by hundreds of ads for inkjet printer cartridges, all claiming to be the cheapest or best. But which vendor do you believe? Our editors decided to put them to the test by anonymously buying printer cartridges and testing them in our office inkjet printers. Many suppliers disappointed but we came up with several web sites that offer good quality cheap inkjet cartridges with impressive customer service.

Windows Backup Software

In this review we looked at 18 different backup software products for home or SOHO use. In the end we could only recommend six though only two were good enough to get our “Editor’s Choice” award

The 46 Best Freeware Programs

There are many free utilities that perform as well or better than expensive commercial products. Our Editor Ian Richards picks out his selection of the very best freeware programs and he comes up with some real gems.