Microsoft Visual Studio .NET: A View From Across The Fence

by Steve Scott

Being given the chance to write about Microsoft Visual Studio in a magazine read by often fanatical Delphi developers could be viewed as either a wonderful opportunity or a poisoned chalice. As I write this sentence I have not yet decided which I think it is. I have always held the view that, as a developer, I will use the best tool for the job. Over the past six years that has almost always been Delphi. It's a great testimony to our beloved development tool that, even when trying to implement new Microsoft technologies such as COM, MTS, ASP and ISAPI, etc, we have still been able to say quite honestly that we were using the best tool. So, with Visual Studio 7, or as it is now named Visual Studio.NET (VS.NET), clearly on the horizon are we going to be able to continue in this vein?

Before we delve into the ins and outs of VS.NET I want to comment on the fact that I can even write about the next incarnation of Microsoft's development environment when it's not due to be released until late this year. VS.NET Beta 1 has been freely available since November 2000. There are public newsgroups for it, frequented by the development team themselves, as well as it being possible for anyone to report a bug they might find. If you compare this to Borland's approach to beta programs, the difference is quite stark. If anybody you know is on the Delphi 6 beta they can't even tell you, let alone discuss the product with you. I'm sure Borland feel they have very good reasons for handling software releases the way they do, but I don't understand them. Although I have no connection with Microsoft at all, and in fact virtually never use the current version of Visual Studio, I really feel like I am able to contribute towards the next release of the product and in some way help shape the final release. Now I'm sure I'm being really naive here, and the chances of any feature I might propose actually making it into the product must be very small indeed, yet that feeling of being involved makes it very likely I will buy the product when it ships.

In total contrast, I feel like I am currently sat here waiting for Borland to tell me what I want in Delphi 6. Unfortunately, perception plays a great part in the success or failure of software releases and, in this case, Microsoft seem to be winning. The other problem this leaves me with is that I can only make comparisons between VS.NET and Delphi 5, even though Delphi 6 will most likely be the shipping version of Delphi when VS.NET finally hits the streets.

OK, so what is different about VS.NET compared to the current version? It's tempting to say *everything*. VS.NET is not a progressive upgrade from Visual Studio 6. As its name suggests, it is designed as

Perspectives: An Introduction From The Editor

Last month, in the Editorial, I said 'we will shortly be introducing a new column called *Perspectives*...' Well, here we are with the first instalment!

The column is different to other parts of the magazine in three ways. Firstly, and most importantly, it aims to give readers a wider perspective than just Delphi and Kylix. Personally, I feel strongly that as developers we should not become totally absorbed in the tool or development approach we happen to be using now: we need to keep an ear open to other technologies. We'll be providing that information in *Perspectives*.

The second thing that's different about *Perspectives* is that the authors will vary: when we cover a particular issue we aim to have input from the person we feel can best give you the information you need. This month, we have Steve Scott discussing Microsoft's forthcoming Visual Studio .NET (VS.NET for short): he writes from the viewpoint of someone who is an active and committed Delphi developer, as well as an active ad experienced user of the VS.NET beta. You will probably have realised by now that the .NET framework and VS.NET will bring significant change in Windows software development (and maybe more widely, perhaps sooner than we had thought, too).

Other topics which could feature in future *Perspectives* instalments include palm devices, the C# language, Java developments, other platforms, competing tools and database technology.

Which brings me to the final thing that's different about *Perspectives*: you won't find it in every issue, only when we feel there's something important to cover. So you needn't worry that you will be losing out on pure Delphi and Kylix material! One of the reasons for increasing the number of pages was to allow us to add more material but maintain the same quantity of pure Delphi and Kylix articles.

Chris Frizelle, Editor (chrisf@itecuk.com) a development environment for Microsoft's new .NET framework. In fact, unless you want to use Visual C++ (which is still not very visual, by the way), I don't think you can develop a standard Windows API-based application. (I do have to exclude Foxpro from all my conclusions, as I have no idea where it fits into this situation.) The fact that VS.NET unashamedly targets the .NET framework raises a whole load of issues; the main one is whether you actually want to develop for the .NET framework in the first place.

As Delphi developers, the closest thing we can compare the .NET framework to is the VCL. Microsoft has spent a long time (over three years) developing a language-neutral object wrapper around the Windows API. This wrapper, or framework, has also been expanded to include memory management and exception handling, as well as a whole host of other goodies. In short, any language designed to use the framework will get immediate garbage collection and exception capabilities, among other things. What's more it can fully utilise and extend any objects or components written in any other .NET-compatible language.

The framework will at first be distributed as an add-on pack for Windows 95, 98, NT and 2000, and will be an integral part of future Microsoft operating systems. It is suspected that at some time in the future Microsoft will stop making the Windows API public and only allow access to the OS through the .NET framework. With this in mind, we have already resolved the above issue: it's not a matter of *if* you want to develop .NET applications, but how long do you intend to hold out before you do. This one is a bit like the Windows 3.1 versus Windows 95 debate we all remember so well from a few years back. The truth is, today almost no one is developing for Windows 3.1, despite what they might have said back then.

As you can see, comparing Visual Studio .NET and Delphi 5 is no longer reasonable, as they target different platforms. When .NET is released, Delphi 5 will be to .NET what Delphi 1 became to Windows 95. As I write, Borland have made no official announcements concerning any plans to support the .NET framework, but a source at Borland has told me that 'Borland are going to be supporting .NET and are working closely with MS to ensure that Borland are able to do this' *[And yes, Mr Lawyer, we have permission to say so! Ed]*. However, until such time as they at least make an official announcement, or start supplying beta product, anyone who sees a big future in .NET is left with no alternative but to examine Visual Studio.

So I've laid my cards on the table: I have said that I think .NET is the future, so what does VS.NET give us to implement this future? The product includes Visual Basic .NET: totally re-engineered and not to be compared with the existing Visual Basic product. It also gives us a new language called C#, which can be described as a C++-based language with most of the nasty bits taken out and quite a lot of the best bits of other languages, like Java and Pascal, added in. Both languages seem fully functional, support single inheritance, events (in the form of delegates) and properties. I have not come across any language limitations yet, though I have not really pushed them to the limit either (I leave that sort of thing to Dave Jewell).

As is the requirement with all .NET languages, both VB and C# produce output in what Microsoft have called Intermediate Language, or IL. The .NET Common Language Runtime (CLR) then takes the IL and produces machine code. The upshot of all this is that there is no discernible runtime performance difference between any .NET language, as they all use the same final just-in-time compiler.

Choosing a language in VS.NET really does come down to your preference of syntax, as all the functionality is supplied by the framework. I've been mainly playing with C#, which is the invention of our old friend Anders Hjelsberg and is also the language which the .NET framework is written in. I'm going to upset quite a few people here by saying that, if Borland leave me in a position where I have to use some language other than Pascal, then C# is not going to be an unpleasant experience.

Visual Studio .NET, at this moment in time, is pretty slow and a little flaky, but it is only a Beta 1 product. The IDE is useable, but I have to say I do prefer the Delphi IDE. It's not my intention to do a full review of the IDE here, just to say again that, if I had to use it, then it is perfectly useable and an awful lot better than Notepad!

Obviously, VS.NET supports a load of new project types to support the new features found in the .NET framework. ASP.NET is a major rehashing of the previously terribly messy ASP technology. I have to say that Microsoft are very good at rubbishing their own technology once they have a replacement for it. ASP, once the flagship of Microsoft web development, is now apparently considered a little cumbersome and difficult to use now that ASP.NET has arrived. Joking aside, ASP.NET looks really good and is a definite on my list for future implementation. The ability to write your own server-side controls is something well worth investigating. If you are a web developer, you must look at this technology. Web Services are also another big thing in .NET, and VS.NET makes developing them pretty easy. Microsoft sees Web Services as the core of future distributed application development. Whether this will be the case only time will tell but, in the meantime, I can think of loads of places where I could implement them.

Up until now I have mainly been concentrating on the new web features available in VS.NET, of which there are many. It would be a mistake, however, to believe that .NET is only about the internet. The .NET framework supplies a whole host of components for traditional single-tier and n-tier development. The IDE, as I said, is still a little clunky (like VB) when building forms, but the good news is that the day of the ActiveX control is finally over.

Just as we can extend the VCL in Delphi by developing our own components, so can we now do exactly the same thing in VS.NET. It is rather impressive, actually. You can inherit from a standard .NET class and extend it using VB, and later inherit from your new VB class in C#. When you debug it, the debugger will step through both the VB and C# code. Language really is no longer a barrier in VS.NET. My only disappointment, so far, is that we don't have the source for the .NET framework itself. I'm not sure if it will be released, but it would be nice if it was. Many traditional Microsoft tool users are obviously raving about features which us Delphi developers have had for some years, but it was obvious that eventually Microsoft would catch up on some of these things.

In summary, despite a whole raft of improvements, VS.NET still looks and feels like Visual Studio (but remember that this is beta software: who knows what might happen between now and the final release). The applications it can produce, however, are of a new breed and way ahead of the traditional Windows applications which we can currently develop using Delphi.

For the past six years, in my opinion, there has been no viable or realistic alternative to Delphi as a general Windows development tool. Now, with the world of .NET rushing headlong at us, I have to ask whether there'll be a viable alternative to VS.NET. I do hope that Borland have properly woken up to this new era and have not been too distracted by this interesting but (in my view) commercially questionable Linux thing.

I will develop .NET applications and, if I have to, I will use VS.NET, but I'd rather see Delphi.NET. The trouble is, with Microsoft releasing this technology to the developer so early, it is going to feel like it's been around for a while even before it is released. This means that Borland really have to get to work *fast*. I'll say it again: *fast*. We have already seen, with Paradox and dBASE for Windows, what happens to development products that arrive too long after their Microsoft competitors.

Steve Scott is CTO of Ascotfirst.com, based in Gloucestershire and Ireland. He has worked with Delphi since its release and not only uses it as his main development tool, but regularly trains others in its delights. In recent years, he has specialised in web development as a developer, trainer and consultant. He also acts as a UK Borland User User Group Technical Leader and resident commentator. You can email Steve at steve.scott@ascotfirst.com