How Visual Basic Morphed

Malcolm Moore
© 05-Aug-2011

Background in Windows XP

As Windows 2000 became the industry standard with its far superior disk management system, the old favourite of Windows 98 SE, which used the Disk Operating System (DOS) behind it was being phased out.

My understanding is that Microsoft recognised the value of Windows 98 SE as a really good visual skin, and that Windows 2000 needed a makeover as Windows 2000 Service Patch (SP) 5 was launched. My interpretation of XP is “cross product”, where the better visual skin was crossed with the far more reliable operating system.

Morphing these two operating systems together meant that the DOS limitations and unreliability along with the lack of read/write control for managing users in Windows 98 SE could be re-skinned as Windows XP, and this would provide a brilliant Windows Operating System from 2002 until at least 2012.

Background in BASIC and C

The programming language BASIC went through several morphed stages from Gee Whiz BASIC (GW Basic), which worked through the console on a pre-Windows DOS, through several iterations into Quick Basic on DOS, through several more iterations into Visual Basic 6 on Windows 2000 and Windows XP.

The programming language C went through several parallel iterations including C# and C++, then several iterations into Visual C++ as it picked up much of the Windows operating system in it. My understanding is that the Windows operating system is written in Visual C++ and as such, Visual C++ is extremely close to native machine code separated by an emulation process over a hardware abstraction layer (HAL) that is chip specific for the processor on the motherboard.

BASIC was originally written as a separate programming code that originally had absolutely no association with the parent computer other than “hooks” so that the BASIC language could communicate with various ports or consoles. As BASIC morphed into Visual Basic (VB) it include more and more “hooks” so that it could talk to more and more operating system-based ports and facilities.

These “hooks” became rather complex and clumsy as they grew from a family of C-based dynamic linked libraries (DLLs), and from a family of C-based Application Programming Interfaces (APIs) so the environment variables can be virtually directly connected and programmed as desired.

While all this was going on, the connections of these DLLs and APIs were beginning to look rather common and it soon became obvious that for most Visual Basic programming that a common framework would be far less error prone – but that common framework was entirely written in a form of C, and that Visual C++ was the parent to that Framework.

Morphing into Visual Basic .Net
The problem was that Visual Basic 6 had fully matured and it has a number of operability problems that simply cannot be resolved unless it were to be totally revised, and this was the problem. How to revise Visual Basic 6 and introduce the C++ based Frameworks for connectivity through the Windows operating system?

I believe that the answer for Microsoft came in a very similar fashion to that for Windows 98 Se and Windows 2000 moving to Windows XP.

I believe that in the case of Visual Basic 6; the product line was abruptly stopped and its skin then had a make-over to give it all the visual features that were lacking and/or were giving serious grief. Concurrently, the Visual C++ product was working perfectly but its' reputation was only for very serious programmers to use, whereas the mainstream programmers were using Visual Basic 6 – and this was the dilemma.

By rewriting the visual interface to look like Visual Basic (VB) but actually code in C++ then Microsoft had a winning formula. Mainstream programmers could then move into Visual Basic .Net and continue coding with almost the same structure as Visual Basic 6, and having the reworked Visual Basic graphics, all the intricate operational problems with Visual Basic 6 are then removed. The third part of the puzzle was that the framework overlays that interface with the Windows operating system can then be used instead of thousands of DLLs and APIs.

In practice, Visual Basic dot Net (VB.Net) looks and feels like Visual Basic 6 but the programming structure is entirely based on C++ and so there are many language changes to learn and become proficient with before becoming proficient with VB.Net.

For example, one of these giveaways is in VB.Net to use the term "Inherits", which is a C programming language construct, showing that VB.Net is really C++ underneath.

Including an Interactive Framework
As it is, VB.Net has gone through a few iterations 2005, 2008 and now 2010, and likewise, thousands of DLLs and .COM files that were common with VB6 have been merged into an interactive framework to directly talk with Windows just above the machine code level, and this makes for extremely fast programmed operations. Many of these DLLs had very similar code with a few differences for specific interfacing and a framework radically reduced the variations, duplications, bad coding practices and a leap into fully object oriented programming (OOP).

Instead of changing several hundred DLLs to fix a common interfacing or security problem far more efficient to change a part of a common framework and address all the problems at once. In another unsurprising move, this framework was called Framework 1 then named Framework 1.1 with developments, and it too has also gone through developments and is now up to Framework 4.

Concluding the Morph
The good news about the morphing of C++ to be under VB.Net is that finally all these languages are converging as they are all using the computers' operating system to a far greater extent than many years ago; and with the interfacing now all done in C++ then this is far more efficient and involves much less interpretation and code conversions in between.