Pick the .NET Language for You

by Juval Löwy

Discuss this opinion in the Talk to the Editors of *Visual Studio Magazine* forum on our Web site. Use this Locator+ code: **VS0309GO_D**

The opinions expressed in this editorial are those of the author and not necessarily the opinions of *VSM*.

The C# and VB.NET languages and development environments are practically identical today. However, I believe this is temporary. In the long run, they will evolve to better serve different markets and needs. Let me explain.

Developing an application as rapidly as possible to be the first to market often comes at the expense of long-term maintenance. These goals conflict, because each requires different sets of skills, quality control, and management mentality.

Traditionally, Visual Basic 6 and its predecessors ruled the Rapid Application Development (RAD) market. Relatively unskilled developers could produce applications quickly and effectively. VB6 did a great job of encapsulating the underlying Windows programming model and interaction with COM. But VB6 had a problem: It would hit the wall at a certain functional point of the application. There were things you simply could not do in VB6 without resorting to tools outside the environment or having extraordinary knowledge of VB6's internal workings (see Figure 1).

On the other hand, languages such as C++ and technologies such as MFC and COM provided limitless capabilities. You could incorporate everything from multithreading to object-oriented modeling to Windows message hooking. Moreover, you could gain such new functionality at an incremental cost in developer skill most of the time. But even the most trivial MFC application required a considerable skill level, and more complex applications were often beyond the reach of most developers.

.NET offers a clean slate. It lowers the entry barrier for developer skills substantially—at least compared to C++. Yet VS.NET doesn't impose the tool limitations VB6 did. .NET's advantages and programming models are analogous to those of C++, COM, and MFC. Proficient VB6 developers appreciate .NET because they can acquire new skills and add advanced functionality incrementally. .NET doesn't require a disproportionate increase in skill level and effort.

At the same time, developers experienced with C++, MFC, and COM love .NET because suddenly everything becomes so much easier. The question is, what happens with relatively unskilled developers whose applications were already easy to build and adequate with VB6? Unfortunately, most VB6 developers fall into that category, and .NET has little to offer them. For RAD purposes, they are better off with VB6.

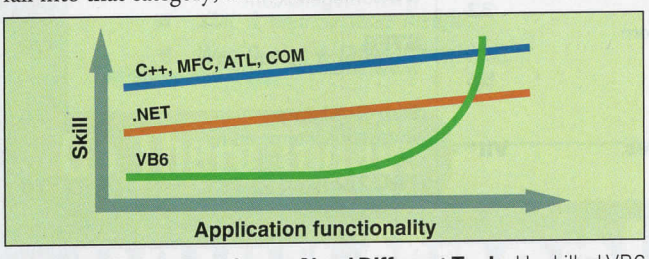

Figure 1 Different Developers Need Different Tools. Unskilled VB6 developers can do a lot with VB6, but find .NET to be too difficult. C++ developers and advanced VB6 developers appreciate the elegance and productivity of .NET, and are not deterred by the higher entrance barrier.

.NET today lacks a tool or language that encapsulates interactions with the .NET class libraries and runtime. Such a tool would provide task-oriented development (Microsoftese for RAD) comparable to what VB6 provided with Windows programming. Microsoft should be well aware of this situation. Future versions of Visual Basic .NET will most likely evolve to provide this missing functionality, to better serve its target audience. Expect to see advanced wizards, classes, and task-automation tools. These will enhance productivity, allowing you to generate applications as fast as possible.

On the other hand, the time saved using a RAD tool is insignificant when amortized over five or seven years of product lifecycle. In such cases, the real question is the cost of long-term maintenance and extensibility. And C# is Microsoft's answer to that question.

Maintenance and extensibility have a lot to do with concepts such as proper design, architecture, abstractions, and component and interface factoring. Developers who wield these concepts tend to spend the bulk of their time focusing on the code itself, rather than on wizards and surrounding tools. C# presently services such developers with a feature set that includes operator overloading, fine-grained control over events and interface definition, built-in documentation, and in the future, generics, iterators, and more.

For code-focused development, features such as edit-and-continue and background compilation are almost irrelevant. Conversely, task-oriented developers don't require unsafe code, generics, and the like. C# and VB.NET's current versions already reflect this divergence, which I expect to widen over time, as the languages evolve to suit their target users more and more.

Task-oriented (RAD) developers should pick VB.NET moving forward, even if they're coming from a C++/MFC background. Developers who value ease of long-term maintenance above all else should pick C#. VSM

Juval Löwy is a software architect and principal of IDesign, a consulting and training company focused on .NET design and migration. He's Microsoft's regional director for the Silicon Valley. His latest book is *Programming .NET Components* (O'Reilly & Associates). Juval speaks frequently at software-development conferences. Contact him at www.idesign.net.