

Getting Real

Yes, Virginia, There Is Life After Visual Basic

By Robert X. Cringely

Complexity is a huge problem in the Microsoft-centric world of desktop computing. Over the last two decades, Redmond has built a large volume of software that kinda-sorta works together, but in the process has also created security gaps and potential performance nightmares. The latter is typified by that cheery term, "DLL Hell," which is even used by Microsoft, itself, in support documents and press releases alike. For software developers, there are several ways out of DLL Hell, but the one I find most interesting is the use of REALBasic from Real Software, which ought to be especially attractive to the more than two million programmers today using Microsoft Visual Basic.

DLL Hell came along because in the mid-1990s, Windows applications were growing in complexity faster than desktop memory and disk space were increasing. The answer to saving disk and memory space without restraining application developers was by sharing code whenever possible through the use of so-called Dynamic Link Libraries. More than one application could use each library, reducing the total amount of code that needed to be loaded or stored. But the problem was that Microsoft supported no plan for DLL versioning, and a new application might load a new version of some DLL that was incompatible with another, long-used, application running on the same machine. So depending on what order applications and DLLs were loaded, programs might or might not work together. And of course, the possible combinations were almost limitless -- so great, in fact, that it was judged impossible to test. It was Hell.

The irony of DLL Hell is that today it is, for the most part, not necessary. Disks are bigger and cheaper and memory is cheaper, too, and it would probably be easier for everyone to just go back to the old system of loading each application as a single .exe file, except of course that's not the way we handle it. Instead we have installers that try to snoop out the biggest DLL conflicts, and we have kludge-arounds like Microsoft's .NET framework that simply require wrapping 21 megabytes of additional code around an application, making it less secure at the same time. So our cure for the disease -- caused by trying to save memory -- is adding more memory, but not the type we really need.

And that's why I like REALBasic, which feels in many ways like a return to a simpler time, except that it also does some things that we could never have done in the good old days, and that we cannot do today with Microsoft products -- like making applications totally cross-platform.

Twenty-five years ago I worked for John Kemeny, co-author of the original (and uncopyrighted, to the chagrin of Kemeny's heirs) BASIC language, which he used to teach programming skills to undergraduates at Dartmouth College. We've come a long way since then, and while there are a lot of programming languages around that still include the term "basic," most of them have strayed far from those interpreted Kemeny and Kurtz roots. REALBasic is no different in that regard -- it is a fully object-oriented language on the order of C++ that includes a compiler. REALBasic includes an Integrated Development Environment (IDE) that feels to developers a lot like Visual Basic while the language, itself, feels more like Java. In short, there IS a learning curve, though it is steep.

Pet peeve: While many people think a steep learning curve means something is difficult to learn, what the term actually means is that the rate of learning is quick, which ought to mean that it is easier, not harder, to learn. Steep is good. Tell a friend.

So if you are an experienced programmer, REALBasic ought to be easy to learn. And the reasons you might want to learn it are simple. You can write software on a Windows machine, for one, and deploy it on Windows, Macintosh, and Linux. Or if you prefer to write software on a Mac, you can deploy on Windows, Mac, or Linux. And in the first quarter of next year, I'm told, you'll even be able to write your REALBasic apps on a Linux workstation and deploy

them on Linux, Windows, or the Mac. The applications are compiled to machine language and don't even require an installer. They tend to be smaller and faster than most alternatives, AND for Windows deployments they are backward-compatible to Windows 98, which current Microsoft solutions definitely are not.

REALBasic began as a Mac shareware product many years ago and has evolved over time into the current product, which is quite mature even if not enough people know about it. The company publishes a magazine and has user conferences just like the bigger boys do. Those who've tried REALBasic love it.

One target audience for REALBasic is the huge population of Visual Basic programmers, especially those who are dissatisfied with Visual Basic 7, which to the eyes of many programmers is not the grand solution that Microsoft had been promising. And of course, Visual Basic 6 is no longer for sale. But converting a Visual Basic application to REALBasic isn't a simple matter of cross-compiling. Real's VB Project Converter application crunches original Visual Basic code, and does most of the heavy lifting required to convert an application from one language to the other, but it doesn't do everything. For one thing, some VB operations don't have precise counterparts in REALBasic. So this isn't something you'd trust to your nephew who likes to play with computers, still the conversion process is vastly quicker than rewriting an app from scratch. One major application vendor allotted 30 days of programmer time to migrate an app and ended-up using only 12 days. Novell on their web site gives a simpler porting example that required only six hours.

Big supporters of REALBasic are Novell, IBM, Apple, and...Microsoft. A demo version of REALBasic ships with Mac Office, and Microsoft's SQL Query app was written in RealBASIC.

Real Software is based in Austin, Texas, and is another of those little companies I want to encourage. And, as always, I have no financial interest in the company. What they are doing is, I believe, unique. Here is a cross-platform compiled language that has full XML support and can function as a very capable parallel to .NET for those who want .NET capability without buying into the Microsoft security model. Its backward compatibility exceeds that of Microsoft products, and the ability to deploy on Linux and Mac simply increase its attractiveness. And if you absolutely must have .NET deployment, then Real Software CEO Geoff Perlman says he'll add it. "We could support .NET as another platform if customer ask for it," he said.

If you don't have access to source code and want to deploy a Windows app on Linux, then take a look at WINE or Codeweaver (deploy a single copy of Microsoft Office as a server to 100 clients -- sheesh!), but understand that those emulation environments incur some additional overhead. If you do have source code, then consider REALBasic, which seems to me the best non-Microsoft solution. Remember that Microsoft doesn't even try to make money on its developer tools, the whole point of which is to force users to upgrade to the newest version of Windows or Office. If you don't want to buy into that economic model, maybe REALBasic is for you.
