# Avoiding Basic Trouble

Adrian Kosmaczewski

2006-10-06

I remember that, late 2004, I was asked by my employer to evaluate the migration of a huge (huge, did I say huge?) Visual Basic 6 “classic” client-server application to an SOA-based Visual Basic .NET one. The application was a business-critical one for several customers, kind of a government ERP system, built initially in VB 3 or 4, and slowly migrated through the years to new versions of VB. Until VB.NET came out.

Now for those that might not know it, even if those two versions of Visual Basic come from the same company, well, they are not AT ALL compatible. That’s how trouble came in.

The problem is that VB6 and VB.NET are architecturally different. They have huge differences in terms of underlying mechanisms, data types and internal protocols, and you just cannot migrate a VB6 app “the easy way”, doing “File/Import/VB6 Project”. It is just not possible at all. Worst of all, these technical words, about architecture and so on, you cannot tell them to your clients; they just say “but it’s Visual Basic, right? So why don’t you just port it and shut up?”. They are blinded by Microsoft’s all mighty presence. The consultant has to suffer. That’s how they see things.

The migration was even more difficult because the application was worth 500 KSLOC, had not the slightest separation between tiers (yes, it was good ol’ plain vanilla VB6, with UI code mixed with business and persistence code), it has not been documented at all, and the only way to know the internals of the thing was to ask the lead developer… who was all fed up with the management of our company, and was about to leave. So you can imagine the scenario.

I should have proposed to use RealBASIC. These are smart people; they have figured out that there is a huge quantity of people still operating with “legacy” (ahem) VB6 code, and that they would happily use .NET if they only could. So RealBASIC came up with a product that:

- Lets you import existing VB6 code
- Provides a Basic dialect with full OOP features (polymorphism, inheritance, etc)

- Lets you build from the same source code, binaries for Linux, Windows and Mac OS (9 and X, PowerPC or Intel).

So that's what I call smart. Why hasn't Microsoft came up with something like this? Instead, in all of its arrogance, Microsoft has dropped millions of "classic" VB users into darkness and oblivion. I know quite a few of them; they like the fast time-to-market that you get with VB. But they are fed up with the company behind.

Now, let's imagine for a minute that VB was open-source technology:

- There would have been a strong opposition to the VB developers to drop support for earlier version; this would have resulted in a "fork" of the open-source project.
- My former employer would have surely had access to the forked Visual Basic compiler, adding true OOP features to the language without dropping support (maybe with a syntax similar to that of RealBASIC)
- The VB.NET version would have appeared, nevertheless, and new VB development would be done in that language instead.

Not only that, but there would be VB plugins for Eclipse, there would have been cross-platform compatibility for a long time, more innovative extensions here and there, and the language would not be that crappy after all. Yes, I have worked with it; some big (big, did I say big?) industrial groups have built their entire IT infrastructure with it. And they are paying the price of using a non-standard, closed, proprietary programming language, with increased costs and migration problems (now and tomorrow).

Now, if you ask me, my strategy, if I were to create a cross-platform application, would be the following:

- Use standard ISO C++ for the backend;
- Use SQLite for storing info in files (instead of pure binary files);
- Use wxWidgets for the frontend (Win and Linux);
- Use Cocoa and Objective-C++ for the Mac frontend.

Then, have everything stored in a Subversion repository, use Eclipse with CDT as IDE, document it with Doxygen, and use CppUnit for unit tests. That's how I would do it. Even if the RealBASIC approach seems interesting, I'd rather use open source technologies instead.