

Re: Complex Windows Applications in C# -- how high is the bar?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-01/msg03201.html>

- *From:* Jeroen Mostert <jmostert@xxxxxxxxx>
 - *Date:* Wed, 23 Jan 2008 19:55:34 +0100
-

Norm Powroz wrote:

Obviously, this is kind of a newbie question, but bear with me...

I am looking for information/opinions on the level of complexity that can be handled in C#, given the current state of the GUI support, using WinForms or WPF. Here's why:

It's Turing-complete, innit? Should be unlimited.

OK, all flippancy aside...

I have an application that is several years old, constructed in C++ using MFC and the BCG Library. It creates (potentially) hundreds of child windows containing various Windows controls, handles docking windows, uses multiple toolbars, ODBC, direct database connections with raw SQL, and so on and so forth. The best way to visualize it is to think about Visual Studio itself -- it looks a lot like that.

Now the issue -- I need to carry out some major work on this application in order to stretch it to handle an entirely new database structure. Because of some philosophical changes in the database structure, it will be easier to scrap the existing application and build a new one from scratch.

I've heard, seen and **done** this so many times that I feel confident in saying that you're almost certainly wrong. Almost.

It always feels easier to programmers to scrap the whole thing and start anew. It sure is more **fun**. And once every so often, when the system has truly reached critical mass and maintenance has become extremely costly, this is actually true. But think very hard about whether you're doing it for this reason, or because you've got shiny new tools and the existing application looks ugly to you. Have you **really** thought about how much time and effort it would cost to adapt what's there, irrespective of how much you'd enjoy it?

Re: Complex Windows Applications in C# -- how high is the bar?

See also <http://www.joelonsoftware.com/articles/fog0000000069.html>

This leaves me with a decision -- do I reimplement in C++/MFC/ODBC, or should I take advantage of new toys,

Apt wording.

and either target Managed C++, or shoot the works and do it in C#.

If you're going for a full rewrite with sparkles on top, and you're planning to do everything anew, then I'd pick C# over Managed C++. Managed C++ is great for what it is, but it's everything and the kitchen sink. It's great if you need to leverage existing C++ code while still profiting from the managed world, but at the same time this mixing of ideas can cause overhead, clutter and lots of annoying glue. It's not my language of choice for new projects.

I am not a C# virgin, but neither do I have the same level of confidence/competence with it as I do with C++.

If you've got prior Java experience, this helps a lot, since C# and Java are practically twins (the philosophies behind them differ in quite a few ways, though).

If you're a good C++ programmer, C# should not give you much trouble. If you're a *great* C++ programmer, C# might give you more trouble, as it doesn't invite you to frolic around with esoteric features as much. Most people consider this a win, though.

I know that the basic program logic will be completely portable, and the database bits can all be handled with either ADO.NET or LINQ. My main concern is the GUI -- this is a very complex application in terms of its user interaction, since it reduces the construction and maintenance of a large database to drag-and-drop simplicity by way of the user interface.

Now here's where you might run into trouble. Although WinForms is solidly based on Win32 windowing, it's obviously much higher level. If you're looking at MFC with a lot of low-level glue (as most apps seem to end up), reusing that logic is pretty much pointless, so you're going to have to redo the GUI quite thoroughly. But hey, you wanted a rewrite. :-)

You could consider using Managed C++ anyway and rewriting only those things that can really profit from managed code. There's nothing wrong with letting the complex GUI stuff remain in the hands of Win32 code and use managed code for the database access, for example.

I have dabbled with both WinForms and WPF, but I am still uneasy about how "down-and-dirty" I'll be able to get.

Re: Complex Windows Applications in C# -- how high is the bar?

This isn't that much of a problem, since the good old WindowProc overriding is still present in .NET. You *can* get down to the individual msg if you need to. But of course, the whole point of WinForms and WPF is that you don't have to do that -- unlike with MFC, where it was very hard to avoid. The problem will lie more in getting thoroughly acquainted with what WinForms and WPF are comfortable doing, and what'll need some careful programming.

--

J.

.