The Business Case for Upgrading Apps to Visual FoxPro in 2013

Whil Hentzen

So 2013 is getting ready to stare us in the face, and it seems every week we're getting calls from customers who have FoxPro 1.x, 2.x or Visual FoxPro applications in production, wondering what to do with them. These customers have thousands, perhaps tens of thousands of hours invested in their applications, systems that in some cases run their entire company, but they're looking at potential problems as Windows goes through its 5th or 6th new release since their application was initially written.

Current FoxPro users have several potential solutions to their quandary.

Door #1 is to do nothing. A lot of companies have been following this path for the last decade, and are starting to worry that while 'no choice is still a choice', it's no longer a good choice.

Door #2 is upgrading their app to the latest version of VFP. No database changes, use of a language they're familiar with (to some degree), may even be able to reuse some of their code and definitely some of their business logic. The flip side is that VFP's EOL is looming.

Door #3 is to rewrite their app from scratch in another language and platform. Promising when compared to #2, considering that VFP is supposed to be on its last legs, but the investment in a complete rewrite in a new language looks staggering, and rare is the company who put money aside for this work.

So, all in all, none of these look very appealing, so they're casting furtive glances at the box that Carol Merrill is bringing down the aisle, hoping for a last minute miracle.

This article argues that since 2011, the business case for upgrading these Fox apps to VFP - instead of doing a complete rewrite in a different language - has actually become stronger. It will come as no surprise to you, gentle reader, that this viewpoint is anathema in many circles. Naysaysers will bring up many specious arguments about why continuing to use VFP is a bad idea. In this article, I'll first debunk those myths. Then I'll provide some solid reasons that make many types of upgrading a compelling proposition. Finally, I'll deliver the knockout punch that will have you reaching for the "Uninstall .NET tutorial" button in your Control Panel app.

Myth #1: The emotional argument

First, there's a very distinct emotional component in the anti-VFP crowd's mindset - VFP is not 'cool' anymore. (Some might argue that it never was, but I'll address that crowd later.) Everyone wants to play with the new shiny things. VFP is very definitely not new or shiny.

Think about the last time a project you worked on failed because of technology. Kind of hard to come up with one, isn't it? Not impossible, sometimes the widgets just don't fit together. More much more often, though, it's the Peopleware (thank you, Tom DeMarco) that fails. Mismatched expectations, unrealistic goals set by unknowledgable management, ego-driven wars between programmers, these are the things that ruin projects.

Indeed, that last one is as big a reason as any - we've all been thought of as the smart ones as we've grown up, so we're used to being right, having our opinion being listened to because it's the most knowledgeable one in the room. And just as epic battles have been fought over Chevy vs Ford, Coke vs Pepsi, and the Red Sox vs the Yankees, developers wage religious wars over the choice of development tools, and Fox has been in the midst of these battles since it took over the desktop from dBASE in the late '80s, and then was bought by and sidelined by Microsoft in the early 90s.

Many supposed "technical arguments" are merely thinly disguised straw horses for emotional disdain. The fragility of the .DBF structure, the lack of a true 'data dictionary' in a

supposed 'database', the overwhelming abundance of commands and functions, the lack of multiple inheritance.... you'll hear developers perjore Fox for these and other reasons. Yet every language or development tool has its weak points; the important thing to remember is whether a weak point matters in the environment it's being used in.

So, separate the emotional arguments from the valid technical reasons, and realize that many of the most heated diatribes come from folks whose growth in social maturity hasn't kept pace with advancements in their technical wizardry. Many people will argue against a technology due to heartfelt desire, rather than rational business justifications, and not be very nice about it.

Myth #2: The technical weakness argument

Second, VFP is, technically, long in the tooth - it never worked hand-in-glove with the Windows subsystems, and today, even more so. SQL access can be problematic. Getting VFP to talk to the Web required the use of a third-party toolset, and getting VFP to build mobile apps is.... better left not even attempted. There **are** better tools.

VFP has its roots in a programming tool developed for the purpose of tracking sports statistics so that the author could better compete in the office football pool. A combination of relative ease of use (compared to other available programming languages at the time), a built-in data engine, and an English-like syntax fueled growth and widespread adoption throughout the world.

It's simple and humble roots, though, means that VFP has never been integrated fully into the Windows ecosystem. This was never a problem when building standalone or LAN applications for the desktop, but means that sophisticated desktop, client-server, Web-based and mobile systems are difficult to impossible to develop. But this discussion doesn't apply to those kinds of systems - we're talking about existing systems that were written and fully deployed with functionality available in 1.x/2.x/early VFP versions. So the fact that VFP doesn't do mobile, for example, is irrelevant. We're not upgrading a 20 year old mobile application.

VFP is technically pretty old. So let's not use it to push state-of-the-art. There are plenty of simple desktop bread-and-butter apps that need to be upgraded and don't need those capabilities. VFP will work just fine.

Myth #3: The termination of Microsoft support

Most importantly in many people's eyes, Microsoft's termination of support in 2015 is the death knell for the product. Who in their right mind would use a programming language that has been EOL'd (End Of Life'd) by it's owner?

Frankly, the termination of "support" is irrelevant. FoxPro 2.x apps were EOL'd in the late 90s, but somebody evidently forgot to tell the apps, as there are plenty still running fine nearly 15 years later. Who needs "support" for VFP from Microsoft? When was the last time you called them for help with your VFP app? If you have a technical question, there are plenty of forums and huge knowledge bases full of questions and answers.

Myth #4: "VFP developers are hard to find"

Nonsense. At the risk of sounding self-serving, go to hentzenwerke.com, write down the emails of all of the authors and editors, and email a handful. A dozen emails will find you a highly skilled and available developer.

Yet you'll still hear people with this lament. What they often mean is "VFP developers with 10 yrs of experience who will work for $50/hr are hard to find." Yes, yes they are. A few years ago, there was a job opening in the south that was being tossed from one headhunter to the next like a hot potato - and they couldn't understand why someone wouldn't relocate for a 6 month, $45/hr job.

So, yes, CHEAP VFP developers *are* hard to find; if that's what you need, good luck with that. Maybe there's a reason that $50/hr VB, Java, and .NET developers are easy to find.

Benefit #1: VFP developers are careerists

At the risk of painting with too broad a brush, developers with other languages often get bored even though their toolset regularly gets rev'd. For example, there are ongoing discussions between .NET developers at conferences, user group meetings, and online where where they are getting bored with .NET. .NET is maturing after more than 10 years of development.

Even the .NET framework only rev'd from 4.0 to 4.5 the last time around. Additionally, Microsoft is not adding things to .NET that appear to be high on developer's ER lists. However, they had the time to revamp the UI to monochrome, which is drawing a lot of flak.

Many of the things that are "new" in .NET are just making other parts of .NET you spent six months learning and perfecting obsolete, a bad practices, or at least uncool.

In contrast, many Fox developers have been in place for two decades or more – how long have YOU been using FoxPro? We're happy and content using it. The best practices we learned 10 years ago are still best practices today. We've gotten GOOD at FoxPro; we are efficient and versatile. And we're not bored. Indeed, even today, don't you get a small thrill every time you fire up the IDE and start typing into the Command Window? I'm reminded of the saying, "I'll give up FoxPro when you pry it from my cold, dead fingers."

Various .NET MVPs have been regularly looking for 'the next best thing', checking out Ruby, Python, Haskell, Scheme, Objective C, among others. To be sure, VFP MVPs regularly migrated from Fox to other languages over the last decade, but not because of boredom – rather, due to the fear that the language wasn't going to last.

Benefit #2: No license fees

We seems to forget that in the Windows world, VFP's model of NO LICENSE FEES is a bastard child. Once you spent your $795 for VFP, nobody in your development ecosphere needs to spend another nickel - not for server licenses, client access licenses, runtimes, nothing. That's a big win for our customers.

Benefit #3: VFP is stable

The product has barely been touched in the last ten years. Some may argue that this is a bad thing, but I say, au contraire, papa bear. Let's face it - we know the language, we know the bugs, we know the workarounds. It's solid and dependable. How many times have you written an interface to an Office product, only to have it break when Office gets upgraded, because the object model was changed (or existing functionality somehow was broken.) An app written in VFP today isn't going to break when "VFP 10" comes out, it won't have to be modified to handle the new features of "VFP 2016", it won't generate panic support calls in five years when an obscure part of the Windows API changes.

Furthermore, as mentioned earlier, since the language doesn't change every few years, the knowledge base on the Web has the answers. Anymore, there are very rarely situations where someone runs into an implementation problem or a programming bug that is 'brand new', unlike languages that have been upgraded or largely rewritten in the new version.

Benefit #4: Fire and forget

Consultants dissing VFP, promoting instead a language that is getting revved regularly, are acting in their own self-interest. They move a customer to a regularly upgraded toolset and lo-and-behold, have created years of ongoing consulting revenue streams for themselves. How many VB or SQL Server or Java or .NET apps written ten years ago are still running 'as is'? They aren't.

As the infrastructure gets updated (I don't use .NET for anything, yet I get updates passed along as part of Windows regularly), the toolset regularly requires mods to be made to their app. VFP is very much 'fire and forget' - good for customers but not so good for developers who ship the app and don't see ongoing support revenue.

And second, the argument that VFP doesn't play well with Windows is actually an advantage. Again, VFP apps are 'fire and forget'. We build an EXE, throw it in a folder with a pointer to the data, and we're done. Windows updates don't break our apps the way they do with languages that are tied tightly to the operating system.

I've built perhaps a dozen large (1,000 hours or more) VFP apps in the last five years and each one of them was delivered, the source turned over to the company for future tweaks, and the game is done. My customers don't have to annually budget for regular upgrades just to handle the new version of a product, but that doesn't add any new functionality that they need. In that scenario, that's a big source of comfort for many customers.

Benefit #5: VFP development costs less

Ask any developer who has built a half dozen desktop apps in both .NET and VFP how the costs compare. The handful of long-time .NET and VFP developers I've talked to say that the same desktop app in .NET will cost at least twice as much, perhaps as much as ten times as much. (These numbers don't take into account the cost of the tools, add-ons, licenses, and so.)

Let's suppose a conservative multiple of three. That means the $250,000 upgrade an existing desktop app to VFP will run $750,000 in .NET. The VFP app will be 'fire and forget' – once installed, it's basically done, except for functional enhancements, for the next ten years (see the next section for an explanation of why this is so.)

Meanwhile, the .NET app will continually have to be managed to handle updates to the current .NET framework installed, and there is the very real danger of it rendered obsolete and

unusable as .NET is rev'd every few years. The $750,000 spent to upgrade doesn't include the maintenance costs for the next decade.

Conclusion: The future game plan includes VFP for many apps

And here's the big finish. An app written in VFP and deployed on Windows today can be expected to live through early 2020s. Why? Because businesses are moving to Windows 7, not Windows 8, and it's been demonstrated that VFP apps will run just fine on Win7.

The word on the street around my neck of the woods is that - by and large - businesses are moving to and staying on Windows 7. Businesses who are on XP (or 2000) are moving to Windows 7 and will stay there for a decade, just like they jumped in with both feet on XP. Mainstream support for Windows 7 ends mid-decade, like VFP, while extended support ends in 2020. So VFP and Windows 7 can co-exist hand-in-hand for, at the least, nearly another decade.

Ask yourself – or ask your customers – are they chomping at the bit with Windows 8? Have they been deploying beta versions on production workstations because the user demand is so high? Are they impatient to roll out Windows 8 to their entire userbase?

Or are they still using XP, and reluctantly realizing they have to bite the bullet and make a full scale switch to Windows 7, realizing that it'll be a lot of work, but at the same time, the investment to do so will carry them through to the early 2020s, just as XP carried them through the early 2010s?

So, my argument is that - in general - they're not moving to Windows 8 and Metro. As a result, your VFP app that runs on Windows 7 has a decade or more of life in it. By 2021, who knows what the computing landscape will look like? Who knows what the business landscape will look like? The U.S. will have been through three more congresses, two presidential elections, and untold changes to the tax code, potentially affecting R&D and capital depreciation. The Web will have changed, hardware will have changed, and there's always the possibility of the appearance of a game-changing technology that turns our world around.

For many in that large group of existing desktop apps already written in Fox, the upgrade to VFP is a pretty safe bet for the next decade.

Author Profile

Whil Hentzen is a independent software developer based in Milwaukee, Wisconsin (as opposed to Milwaukee, Minnesota, as many people think.) His writing has killed many trees over the years, but none since 2007. He has realized he really sort of misses it. You can reach him at whil@whilhentzen.com.