## Version icon: 2.x, 3.0, 5.0
## Platform icon: DOS, Windows, Mac, Unix

## Figures: NONE
## File for Companion Disk:

I'm Afraid We'll Have to Rewrite . . .

Whil Hentzen

We've sold six VFP applications since DevCon—on top of four in the few weeks before DevCon. This means two things. First, we're in desperate need of several developers and technical support personnel (just like every other FoxPro development shop in the country), and, yes, our phone number is …<grin>. Second, not all of these apps are new; several are rewrites or upgrades from existing DOS systems. They could either be pure conversions or rewrites with significant new functionality. The key is that they're not brand new—we don't have a clean sheet of paper with which to start.

It's tempting to strut into each customer who's looking at a conversion and pronounce, "We can't do a conversion—it simply doesn't make sense. We're going to have to rewrite the whole thing from scratch." One can present a great number of arguments that make logical, technical, and strategic sense. However, this argument simply isn't going to fly when the customer explains, "We have 32 users on a system with 88 screens, 142 reports, and 290 megabytes of live data. We are not going to spend nine months rewriting it from scratch, and then, over a weekend, shut the old system down and switch over to the new system." The customer is going to require a systematic, step-by-step, progressive conversion.

So where to go from here?

The first piece in our strategy is to simply run the existing application in VFP 5.0. What do you need to do in order to make this happen? Three things. First, you'll have to convert the various components to 5.0 format. Many things will convert reasonably easily: menus, reports, programs, and the like. The conversion of screens isn't as clean a task. Opening a 2.x or 3.0 screen in VFP 5.0 brings the Transporter forward, and two choices are available. A functional conversion converts the SCX and moves the code into poor choices of snippets. A visual conversion converts the screen and puts the code into a PRG.

Unfortunately, I don't really like how either conversion is done. Why? Probably the same reason why most people prefer to rewrite existing functions instead of reusing them. In the first case, the screens work but the code is in the wrong place. Nonetheless, the stuff works. In the second case, the conversion changes into a form where you can tweak things as you need. In either case, you're faced with a fair amount of work, but you'll be able to get the existing application up and running. And that's the point right now - get the app running in the new version. We'll pretty things up later.

The second part of getting things running is dealing with any third-party librariespsed by the system Unless you're a C programmer, you're probably going to have to get new versions, unless the functionality can be replaced by VFP 5.0's native functionality or another way. If you haven't already, this is a good place to encapsulate that functionality and call a common function from multiple places.

The third step is to handle the foundation read and the event model. Yes, the old model would work, but in order to take advantage of VFP 5.0's new features, you're eventually going to need a new foundation. Here's where it needs to be taken care. Putting it off won't buy you anything and will inhibit you from most new work. Regardless of how the application was coded in the first place, this won't be a trivial undertaking. But it can be accomplished with far less work than the aforementioned wholesale conversion. Eventually, you'll have a READ EVENTS and an application object supporting the new system.

The second piece in our strategy is where the fun begins. The application is running and it's time to incrementally handle procedure files and data structures. Examine the procedure libraries used in the current application and move what you can into the application class. This is the time to create additional class libraries as needed, and move more procedures into those classes as well. At the same time, consider taking advantage of the new capabilities—data types, database container, and so on.

Now you can add functionality to the application in true VFP fashion, and it’s a whole new ball game.