December, 1998

Editor's View

What is Component Development?

And Why Are We Hearing So Much About it?

The science (or is that art?) of computer programming has gone through many changes in the not so many decades it's existed. A lot of the changes have been aimed at making it easier to write solid, maintainable code.

The introduction of subprograms (procedures and functions) way back when meant that you could write a small piece of code and test it every which way. Once it was working, you could call on it from lots of different places and count on it to do its job every time.

Object oriented programming took this idea to the next step. You could create an abstract object with its own memory (properties) and a number of capabilities (each in the form of a subprogram called a method) and test it to death. Again, once you made it work, you could count on it to do its job.

Component development (Microsoft's version is called COM) is the next step in this evolution. With components, you get access to functionality external to your application and maybe even external to the PC it's running on. As with the objects you develop, components provide you with tested parts you can plug into your applications. In fact, the term "component" is a pretty good one as it brings to mind a stereo system, assembled by plugging various components into a receiver. Each component does only one task and knows how to communicate with the receiver.

Why do we need access to things outside our applications? For several reasons.

First, we've finally learned the lesson that a single application or development tool doesn't have to be all things to all people. For example, Word is a tremendously powerful word processor. Why would we want to try to recreate its functionality in our applications (or even ask Microsoft to include that functionality in Visual FoxPro)? Instead, just let us use Word directly. Same thing for Excel and other good applications.

But it goes beyond that. Some of the tasks we're assigning to PCs today are more complex than what we used them for last year or the year before. We're talking about running enterprise-level applications using networks of PCs. In order to do these things, we need reliable tools to ensure that mission-critical data is safely maintained. Components let us work at a much higher level than developing all the code for all the pieces ourselves. They make it possible to create very complex applications.

But what if you're not building that kind of application? What if your users will never *need* multi-tier applications using COM components and distributed transactions. Components still have something to offer.

Consider using ActiveX controls in situations where they can provide a better input mechanism. Think of automation to other applications to simplify tasks for which VFP wasn't designed. (I'll tell you next month how I used automation to Word to handle some tough problems.) You don't have to use *every* new technology in every application to get the benefits of COM.

VFP 5 and 6 also let you build your own components. Consider encapsulating key functionality for use by your applications and others.

Do keep in mind that many of the COM technologies (like ADO and MTS) are still fairly new and the road may be a little bumpy at first, as the integration of all these tools evolves. As the new tools mature, they'll get easier to use and more reliable.

In the meantime, use the pieces you need. If your applications can benefit from the whole raft of options, use them all cautiously.

No matter whether you're developing enterprise-level applications for multi-national corporations or bookkeeping systems for the small business down the road, look forward to more and better components to handle routine tasks so that you can focus on the things that are unique about your applications.

In memory of Geary Rachel

I've written many times about the FoxPro community, talking both about the willingness to share technically and about the relationships formed through that sharing. I've formed some friendships in this community that I know will last well beyond our common interest in FoxPro.

The hard part of belonging to a community is dealing with the death of a member. The FoxPro community recently lost Geary (pronounced "Gary") Rachel. Although he wasn't one of the superstars of FoxPro, he was a giving, caring contributor to the community. (His contributions included a FoxPro Advisor article several years ago on speed differences when using the "m." prefix for variables.)

Geary had lifelong physical challenges and on-going health problems. Despite them, he always presented a pleasant face to the world and a positive attitude about his own future. At conferences, there was always a camera around his neck, and many of us have photos he kindly sent afterward. Contributing Editor Mac Rubel tells me that, despite a withered right arm, Geary was always ready to go fly-fishing and, even at the end, was making plans for the future. Ultimately, disease won. But the FoxPro community is richer for having known Geary and is poorer today for his loss.