

Re: Implementing a Macro Language

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2004-06/1199.html>

From: Joseph M. Newcomer (*newcomer_at_flounder.com*)

Date: 06/11/04

Date: Fri, 11 Jun 2004 11:16:13 -0400

Well, the way most of these macro systems work is in terms of automation, which is why they look so easy. Examples include the macro systems in Word and other Office products.

What do you mean "store the strings chosen"? What if they are in list boxes, what about check boxes, owner-drawn combo boxes, etc.? There aren't any "strings" in an owner-drawn listbox or combo box, so what do you store? The pointer to the ItemData?

The selection mechanism is usually a fatal problem, because so few applications make it available, except possibly via COM interfaces.

Feel free to try, but I've seen so many of these efforts crash and burn that I consider it futile. Especially because they spend many months on the project, and finally we get a question in the newsgroup that explains what they can't do, and most of us who reply say "Yes, you're right, that cannot be done".

I've done a few of these myself, and even in restricted cases they are extremely hard to get right. I would never attempt to build one for the general case.

Or to put it another way: suppose someone came to me and said "We have \$100,000. It is yours if you can deliver us a working macro system that <for example, insert your description here>". Unless I could get them to write an incredibly restricted spec, much more restricted than what you proposed, I would have to turn down such a contract because I know that I could not fulfill it. Which means either I'd be wasting my time for a year, with no hope of being paid, or I'd be stealing their money. And I don't believe a spec restricted enough that I could accept the job would actually be a useful spec.

While there are many objections to COM as a technology, it was created to solve a very complex problem, the problem of automating an app. While we may disagree with the details of how COM works, nothing changes the fact that the abstract model, as complex as it is, is what is needed. .NET gets rid of most of the baggage of COM while retaining the functionality, which is an abstract automation interface. So what you are attempting to do is to do is reinvent COM or .NET automation interfaces. I'm not sure I'd be willing to tackle that with a crack team of serious Window experts, let alone on my own.

joe

On Fri, 11 Jun 2004 08:53:17 -0600, "Jonathan Wood" <jwood@softcircuits.com> wrote: