Subject: Re: IDL objected oriented question
Posted by David Fanning on Wed, 09 Apr 2003 19:52:45 GMT
View Forum Message <> Reply to Message

Pavel Romashkin (pavel_romashkin@hotmail.com) writes:

> There is no question that IDL object implementation is crippled. Just as
> there is no question that even as it is, it is very useful.

I would say "limited", rather than crippled. But I agree
it is more useful than not having objects at all.

RSI has a problem, though. Objects are real programming
tools. But because they are somewhat limiting (e.g., no
public data, no operator overloading, etc.) they don't
appeal, particularly, to the C++ programmer. And they appear
too complicated to the average scientist who is not necessarily
a computer programmer. Yet IDL is seeming sold to
people (it appears to me) with less and less programming
background. How in the world can we get people with
limited programming expertise to use the tools that will
benefit them the most in their programs?

I think RSI's answer to this is to build yet more
sophisticated object tools that can be used from the
IDL command line. All well and good if the tool does
what you want it to do. Which I'm sure it does, some
of the time. :-)

But what about when it doesn't? What if you don't care
to be pushed into really sophisticated object graphics?
What if you just wanted to build your own little tool,
using the direct graphics commands you already know and
understand, in a simple graphical interface, taking
advantage of the numerous benefits of objects? Then,
I'm afraid, RSI has not made life particularly easy for you.

Dave and I have struggled with these questions for over
a year now. We are still struggling a little bit with the
proper amount of "abstraction" to build into our library.
We wish to make the library easy to use, but not opaque
to the average user. We want scientists to be able to use
it easily. We realize it is a quixotic mission (and Dave
tells me that with my programming skills I'm a natural
for the role of Sancho), but we think it can be done.
We have already proven that these simple tools can be
scaled into large applications.

And because we have based our object framework around
the concept of a widget framework, we have shown that
even programmers with just basic widget programming
experience and no prior knowledge of objects can
easily learn to program in this system. In fact,
our library probably has a more consistent interface
for the user than even widgets themselves.

> With regard to Catalyst, I suppose, you just right click on
> dfanning.com/catalyst.zip and choose "download to disk", and enjoy :-)

Yes, something like that. You will have to throw a few
coins in the jar, too. :-)

Dave and I have spent about 8 man-months building this
system in our "spare" time. We plan to use it in our own
consulting work because it will give us an edge in building
applications faster than the competition. We haven't decided
exactly how we are going to sell it. (I've always belonged to
the oh-hell-give-it-away-and-be-famous school, but Dave has
a young family to think about.) But, of course, I can't
resist writing about it. We have solved a LOT of problems
associated with marrying objects and widgets. *Someone* needs
to know about that!

> Jokes aside, I thought of doing a similar thing as Catalyst, but for a
> different reason - using small widget systems that don't even have to be
> visible to help objects do their business.

Yes, exactly. A widget can be one of several physical
manifestations of an object. Another could be a VB control, etc.
Objects can have different physical manifestations at different
times, etc. Objects are infinitely flexible, it seems.
Most of our objects are built with the ability to call
a "Control Panel" that is a physical manifestation that allows
the user to change the properties of the object interactively.

> So to say, fill the gaps in
> object implementation with widget events, just like in, for instance,
> VBA, where objects can simply listen to each other (in addition to true
> encapsulation and automatic cleanup :-)

Yes, in addition to being able to pass widget events around
the object hierarchy, our objects also have a "messaging" system
built into them. If your object is interested in registering with
my object you can do so, specifying exactly what "message" you
want to receive. When my object sends a particular type of message
(they are published), then any object that has registered with the

object will receive the message. So, for example, a slider object
could broadcast its value to three or four different draw widget
objects, so all could update their image contents, etc.

> Then of course, it all comes back to a global event sink, object
> self-awareness and transmogrification. Which ends up to be a Common
> variable or elusive orphaned pointer :-)

Uh, right. But we clean it up. (At least I *think* we do. Let me
check with Dave.)

> The reason I haven't done it is, I never really needed it. Shouldn't be
> very difficult and a fun, challenging project. Any takers?

Actually, it was more difficult than I thought it would be when
we started it. But it really, really helps to have someone to
help you with it. Solving all these problems on your own can
be a real drag. :-)

Cheers,

David

P.S. This library is far from finished, but it is in good enough
shape for early adaptors to play with a little bit. We are working
on a User's Guide sort of thing now that explains the philosophy
of what we are trying to do and how the framework works. That *will*
be available on my web page when its finished. In the meantime, we
are looking for someone with a good application (and maybe a little
cash) who wants to see what this library can do for them. We will be
more than happy to hold your hand through the entire process. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155