Subject: Re: Object-oriented IDL?
Posted by Ken Knighton on Tue, 27 Feb 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Ray Osborn <ROsborn@anl.gov> wrote:

> ...I have been
> put off by what I believe is a fundamental limitation in its abilities
> to handle composite data structures.

IDL can handle any type and any structure of data. Ever since pointer support was added, IDL has been able to handle any dynamic or static data structure. IDL can also perform any algorithm that does not require directly accessing the operating system or the machine itself. Perhaps one more knowlegeable than I can provide a detailed description of a data structure or a non-system programming type algorithm that can not be handled by IDL. Of course, IDL can interface to external routines that can perform system programming tasks.

I believe you are really referring to elegance. Object oriented languages, used correctly, can be used to handle some situations in a more elegant and more conceptually straightforward manner than their non-object oriented brethren.

> I would be interested to hear from
> more experienced IDL users, and perhaps the company itself, if they
> think that this is a genuine limitation and whether it's something that
> can/will be addressed in the future.

It is not a limitation, but there is always room for features that will improve IDL and its ease of use.

>
> IDL is an array-based language,

No it isn't, but IDL has good support for arrays.

> but when I perform data analyses, I don't deal with arrays...

Nor does anyone. People analyze representations of the real world. If they are lucky, these representations map to simple data structures.

> If IDL were an object-based language, ...

Any processing that can be done with an object oriented language can be done with a functional/procedural language such as IDL.

> rather than an array-based one, I

> could overload the "+" operator to perform all these operations in a
> consistent fashion. As I understand it (and I'm not an OOP expert),
> operator overloading just involves defining a standard set of procedures
> that are invoked when performing mathematical operations on these
> spectrum "objects". Once written, I can perform these operations much
> more transparently with far fewer lines of IDL code knowing that, for
> example, the errors are properly handled with each operation.

In IDL, you can define functions/procedures to do all of this. The amount of work that is involved is very similar whether you are using an OOL or not. The result may look a bit simpler with an OOL, but I am not a big fan of operator overloading since the casual reader of code has no way of knowing that the operator has been overloaded without going through reams of code to figure it all out. Functions jump up and slap you in the face.

> Incidentally, data plotting could also be simplified since the
> independent axes and their labels could be carried around with the
> spectrum, and would not have to be specified in the plot command.

This can already be done, but IDL certainly needs an improved graphics package to make it simpler.

Well, enough of my opinions.

Ken Knighton knighton@gav.gat.com knighton@cts.com
General Atomics
San Diego, CA
