Subject: Object Blues
Posted by J.D. Smith on Fri, 16 Jan 1998 08:00:00 GMT
View Forum Message <> Reply to Message

I am currently working on an event-driven object communication superclass, to allow for arbitrarily complex "messaging" (as opposed to the limited up-the-tree event-processing paradigm already provided by xmanager). During the course of building this framework, I have come across several irritations with IDL's implementation of its object-oriented interface.

I have already posted my thoughts concerning the first (and perhaps the more crippling) of these to this group months ago -- the fact that inherited keyword parameters are *not* passed by reference makes proper subclassing of, for instance, the various GetProperty methods of the object graphics classes quite cumbersome. (Basically, you cannot "chain" inherited keyword return parameters up the class tree).

The second complaint is that IDL classes are not permitted to define public data members. That is, you can't access "someObject.somedata" anywhere outside that objec'ts own methods. Perhaps RSI saw the exclusion of this feature as a simplifying decision, and one that guarantees encapsulation. Indeed, limiting object access to its method procedures and functions may, at first thought, seem an obvious way to keep object code well-contained. However, it has other unfortunate consequences, usually related to the need to acquire a small group of parameters from an object for extensive and repeated use. The only solution within the current context is a GetProperty method (which, as you may recall, is hampered by the non-reference passing of inherited keywords). This is the solution implemented by IDL's object graphics. I maintain that an object method (such as most of the GetProperty's) which simply returns some subset of the member data without any pre-processing is wasteful and inelegant, and that the true power of encapsulation is in the ability to select which parts of an object to encapsulate.

I therefore urge RSI to consider implementing a public data member capacity into its object interface. This may break the exact parity between structure and class definitions, but I for one have never reused structures as classes or visa-versa, nor do I see the advantage of this interoperability (other than notational convenience).

If you have encountered similar frustrations, please let RSI know about it.

Thanks,

JD