Subject: Re: OO IDL
Posted by Michael Wallace on Thu, 16 Sep 2004 03:40:03 GMT
View Forum Message <> Reply to Message

There are definite benefits to OO, but OO isn't something to use in all places. From the problem you outlined, I don't see any distinct advantage of the OO approach. Personally, I'd be more apt to use OO since I am much more of an OO programmer than a procedural one. However, if you feel more comfortable with the procedural approach, go with it.

-Mike

Robert Barnett wrote:
>
> I'm curious about common ways to call differing versions of code. I have
> implemented OO (Object Oriented) IDL to achieve this common task and
> wanted to know what peoples thoughts might be.
>
> I have several routines, each which have many different versions. In
> many cases, no version is any more recent than any other. It's more that
> each version is applicable for different problems.
>
> The programs are in their own .pro files, with the filename and function
> name being the same so that autoloading works. They are also in
> lowercase so that autoloading works correctly. The version is just
> appended onto the end like so:
>
> cost_function_mem.pro
> cost_function_lb.pro
> cost_function_sr.pro
> ...
>
> simplex_fast.pro
> simplex_slow.pro
> ...
>
> ... and on it goes
>
> This means that I have to do lots of calls to CALL_FUNCTION becuase I
> only know what version I am to use at runtime.
>
> I'm having a play around with OO IDL and seeing if there is a way to do
> this without using CALL_FUNCTION, and seeing if there are any advantages
> in doing so.
>
> The only way I can see to avoid the use of CALL_FUNCTION is to create a

> class for each function.
>
> mem::cost_function
> lb::cost_function
> sr::cost_function
> ...
>
> fast::simplex
> slow::simplex
> ...
>
> It is now possible to call a cost function like so:
> cf -> cost_function()
> Where cf could be
> cf = obj_new('mem')
> cf = obj_new('lb')
> cf = obj_new('sr')
>
> Unfortunatley, this causes a maintainence issue with structures. I now
> also need to define
> mem__define
> lb__define
> sr__define
> fast__define
> slow__define
> However, is it easy to write a trivial shell or perl script for
> generating these.
>
> It seems that both OO and CALL_FUNCTION require the same number of lines
> of code aside from the maintainence of the OO structures.
>
> Some advantages of OO may be
> * The ability for objects to inherit each other, thus being able to use
> each others methods.
> * Each class has its own namespace, ensuring that all methods which are
> not in conflict with other versions
> * Each class could have instance data, thus saving effort in passing
> information down the call stack and back again.
>
> Disadvantages
> * It may not be entirely obvious where instance data comes from
> * It may not be entirely obvious which objects inherit each other
> * A change in class struct definitions requires IDL to restart.
>
> The advantages of OO, although desirable don't seem to have a huge
> impact. Makes me wonder if anyone has an IDL OO success story.
>