Subject: Re: Object based/oriented IDL ? Ever likely ?
Posted by ROsborn on Tue, 19 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

In article <julien.827283540@marsh>, julien@cs.curtin.edu.au (Julien Flack) wrote:

> I am very impressed with a number of features of IDL. However, I think
> that its lacking support for structure (primarily data structures) due to
> its historical affiliation with Fortran (no flames please). This weakness
> becomes noticable when you reach a 3,000+ line application (IMHO).
>
> I think that a version of IDL using object based/oriented technology would
> be immensely powerful and would reach a far wider audience. Is there a
> desire for OO technology in the scientific community, or is Fortran still
> predominant ? Have RSI made any moves in this direction ?
>

I'm amused that you've posted this question, because I posted an identical question only a couple of weeks ago. I got a couple of responses, mostly stating that IDL was flexible enough to cope with any programming needs, and expressing some scepticism of the need for OO concepts. Otherwise, the thread sank like a trace.

Just in case it does spark some interest this time, I'll briefly repeat my reason for raising the issue. The concern I have with IDL is that I deal with composite data objects, or spectra, comprising several arrays (usually x, y and error arrays with axis labels, various attributes etc.). When I combine different spectra e.g. in subtracting a background run, I have a considerable amount of book-keeping to do, such as checking that the x-arrays are compatible, passing the labels across, propagating the errors etc. If IDL incorporated some OO features, it would be possible to define a spectrum class so that I could overload the arithmetic operators and hide this book-keeping from my interactive session, making my online analyses much more productive IMHO.

You have raised another equally valid issue, that of simplifying large IDL programs. One of the responses to my posting drew my attention to a suite of IDL procedures developed at the Institut Laue Langevin, Grenoble, called LAMP, which attempts to treat multi-dimensional spectra in the way I wanted. Strangely enough, it was this suite which made me concerned about the limitations of IDL in the first place. It is a very impressive achievement, and contains some very nice features, but it has required well over 10000 lines of code, many of which I am sure would be unnecessary if IDL contained some OO concepts. I think that it would also be a lot easier to maintain.

I also would be interested if RSI has any intention of moving in this direction.

--
Ray Osborn   Tel: +1 (708) 252-9011
Materials Science Division   Fax: +1 (708) 252-7777
Argonne National Laboratory   E-mail: ROsborn@anl.gov
Argonne, IL 60439-4845

Subject: Object based/oriented IDL ? Ever likely ?
Posted by julien on Wed, 20 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

I am very impressed with a number of features of IDL. However, I think
that its lacking support for structure (primarily data structures) due to
its historical affiliation with Fortran (no flames please). This weakness
becomes noticable when you reach a 3,000+ line application (IMHO).

I think that a version of IDL using object based/oriented technology would
be immensely powerful and would reach a far wider audience. Is there a
desire for OO technology in the scientific community, or is Fortran still
predominant ? Have RSI made any moves in this direction ?

Any news, views and gossip welcome ...

--
Julien.

Subject: Re: Object based/oriented IDL ? Ever likely ?
Posted by kspencer on Fri, 22 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Ken Knighton <knighton@gav.gat.com> writes:

> IDL is not an OOL, but it does have a number of features that
> make it possible to do things traditionally supported by
> OOLs although perhaps not with the same elegance:

This was worth reading, thanks.

> It is unfortunate that some of IDL's truly powerful features tend to
> be hidden or unknown to the majority of users. It is also unfortunate
> that RSI doesn't use them in most of the code they supply with IDL.
> It would be nice to have a bunch of tools supplied with IDL that were
> written in IDL using excellent software engineering practices and
> the powerful techniques that are already available in the language.
> That way, users would have examples to go by when creating their own
> cool software.

Would you give some examples of the "powerful features" you're talking
about? I'm curious, and want to find out if there's anything I'm
missing.

Kevin Spencer
Cognitive Psychophysiology Laboratory and Beckman Institute
University of Illinois at Urbana-Champaign
kspencer@p300.cpl.uiuc.edu / kspencer@psych.uiuc.edu

Subject: Re: Object based/oriented IDL ? Ever likely ?
Posted by Ken Knighton on Tue, 26 Mar 1996 08:00:00 GMT
View Forum Message <> Reply to Message

kspencer@s.psych.uiuc.edu (Kevin Spencer) wrote:
> Ken Knighton <knighton@gav.gat.com> writes:
> Would you give some examples of the "powerful features" you're talking
> about? I'm curious, and want to find out if there's anything I'm
> missing.
>
The answer to this will have to be on the installment plan. :-)

1) Polymorphism

   a. Functions/procedures can be called with a variable number of
      formal parameters.

   b. Since identifiers are dynamically typed, a single func/pro
      can be devised that performs an operation on a variety of
      input argument types.

The following tiny function shows how, by virtue of the fact that
IDL is dynamically typed, functions can be designed with varying
types and numbers of parameters. Note that type checking could
be added to this function to produce errors if incompatible data
types were used. Or, one could use the CATCH statement to react
to any errors that may occur (such as failure to convert a string
to a number if mixed strings and numbers were being used).

```
;Trivial, contrived, and useless example of "polymorphism" in IDL.
FUNCTION Add, p1, p2, p3, p4, p5, p6, p7, p9, p10

  lParams = N_PARAMS()

  CASE lParams OF
```

```
    2L:   xSum = p1+p2
    3L:   xSum = p1+p2+p3
    4L:   xSum = p1+p2+p3+p4
    5L:   xSum = p1+p2+p3+p4+p5
    6L:   xSum = p1+p2+p3+p4+p5+p6
    7L:   xSum = p1+p2+p3+p4+p5+p6+p7
    8L:   xSum = p1+p2+p3+p4+p5+p6+p7+p8
    9L:   xSum = p1+p2+p3+p4+p5+p6+p7+p8+p9
    10L:  xSum = p1+p2+p3+p4+p5+p6+p7+p8+p9+p10

    ELSE: MESSAGE, 'Must use 2 through 10 parameters.'
  ENDCASE

  RETURN, xSum
END
```

There are also ways of doing the above without using a CASE statement.
One of these is to use the EXECUTE command and a FOR loop:

```
xSum = p1+p2
FOR i=3, lParams DO BEGIN
  aExec = 'xSum = xSum + p'+STRTRIM(i,2)
  lErr = EXECUTE(aExec)
ENDFOR
```

Of course, the case statement runs much more quickly and is more
obvious in its logic. However, the EXECUTE statement has its place
and provides on-the-fly compilation and execution of statements.

If you call the above function using a variety of input types, you will
soon notice that the actual parameters can be of any numeric or string
type and can be either scalars or arrays. If strings and numerics are
mixed, then the strings must be able to convert to numeric type. One
can not use structures in the above example, but one could modify this
code to check for structures using the SIZE function and then take
action accordingly.

As you can see, it is fairly easy to write one function that takes
care of a wide variety of possibilities for input arguments.

I'll try to continue this discussion later. Any feedback is welcome.
If someone has a better example, please post.

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