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
