Evolution, Not Revolution!

by
Ralph Moore
President, Micro Digital Inc.

Engineers often contact us who have inherited hopelessly convoluted software from the past. Typically this happens because their predecessors used an inadequate, in-house kernel -- or worse, didn't use a kernel at all. The problem has been exacerbated over the years by adding feature upon feature. The situation is a nightmare for the engineers charged with yet another modification. To them, the solution seems obvious: start over with a commercial kernel and fix other problems as well.

The difficulty with this solution is that upper management seldom approves of it. Managers reason that "if it ain't broke, don't fix it." Also they are probably still smarting from the severe schedule slips and major budget overruns which characterized the development of the current product. For them, that was a nightmare!

The net result of this standoff is that the current software continues to degrade and frustrations continue to mount. This is not good for the company, nor for the individuals involved.

Fortunately, there is another approach: **the evolutionary approach**. This consists of **introducing *smx* into the current product with minimal change**. There are two ways to accomplish this:

1. Treat the current background code as a single or, at most, a few tasks. Make minor modifications to foreground code (encapsulate irs's with *smx* macro's and move some code into lsr's), or

2. Emulate the current kernel with *smx* -- i.e. replace the current kernel with translation routines which use *smx* calls.

The second approach may be appropriate if a well-defined, in-house kernel exists. However, the first approach will normally be the route taken.

The main objective is to demonstrate, with minimal effort, that *smx* can be introduced without upsetting the apple cart. If this cannot be achieved, then *smx* may be returned and only a small cost has been incurred. If the demonstration does succeed, everyone gains confidence to go on to the next step.

The next step is to leave *smx* in the product and to implement required new features via *smx* tasks. Meanwhile, the old software is slowly reworked into more and more *smx* tasks interacting via *smx* facilities. In the process, old problems start to disappear, the code becomes simpler and easier to work with, performance improves, and everyone's frustration levels diminish.

To facilitate the evolutionary approach, *smx* is available on a 30-day trial basis.

###

For further information, contact **Ralph Moore** at Micro Digital, Inc. 1-800-366-2491.

\\server\c\smxd\articles\evolution.doc 6/22//95