RATIONALIZING VICAR WITHIN A TAE FRAMEWORK—SOME PROBLEMS AND SOME SOLUTIONS

Colin P. Harris
Atmospheric Physics Group
Imperial College of Science and Technology
London SW7 2BZ

Abstract.

TAE implementation may impose a strain on centres with modest resources. This may be eased in a number of ways. The balance of a small number of expert users and a large number of computing novices at IPIPS imposes special constraints. Some solutions to these and other particular problems are described.

1. Introduction.

In many environments where complex applications software is used, implementation of a new system may inconvenience users and systems staff alike. Where there is no shortage of staff or time, a TAE-based system may be fully constructed and tested before being released to users. During this period, however, any contruction which reduces the strain of transition for both staff and users is clearly desirable. During the implementation of TAE on the IPIPS image processing system at Imperial College a number of such constructions were learnt. In addition, the selected TAE configuration had to encompass the particular needs of two user groups, and to complement the existing hardware and software.

2. The IPIPS system.

a. The target community.

As described by Grove (1985), the IPIPS system has two complementary user communities. Around one third of the users are researchers with varying amounts of experience who need access to the image processing subroutines in order to produce their own specialised software. The majority of the rest are masters degree students in remote sensing, around 25 in all, with little or no computing experience. Finally, there are visitors and external users, using the IPIPS system as a facility for their own work. People in this group need an intermediate level of user-friendliness. They may have quite complex needs, and may be very experienced on other systems, but will need guidance to find

the appropriate programs from the large libraries available.

The interests of the IPIPS users are wide-ranging. The original system was developed for terrestrial meteorology and Voyager image analysis, but subsequent involvement with the Centre for Remote Sensing at Imperial College has taken it into many more areas. These include Earth resources, ocean colour, pattern recognition and texture analysis, among others.

b. The hardware configuration.

The core of the IPIPS system (Hunt et al. 1985) is a DEC VAX 11/780 general purpose computer, with storage consisting of two 1600 bpi tape drives and four user accessable disks in addition to the system disk. Output devices include printer, plotter and video and photographic facilities. In addition there are two I2S dedicated image processing systems. These consist of user interface (trackball or tablet), colour monitor, and a processing unit coupled to a set of refresh memories, each of which may contain a 512x512x8 bit image. The Model 70E has 6 such channels whilst the newer Model 75 I2S has 15. Each I2S is capable of rapid and complex manipulation of single frames or sequences of images, in black and white or colour. The processing unit permits operations such as image convolution to be carried out in hardware, independently of the VAX. The I2S system is particularly suited to applications in Earth resources and meteorology. For a fuller description of the I2S capabilities see Adams and Driscoll (1979).

c. The software configuration.

The software available on the IPIPS system before the implementation of TAE was based around the VICAR system for image processing and parametering. This was originally transported from the JPL IBM in the late 1970s (see Castleman 1979 for a description of the VICAR system). A transportable VAX version was subsequently developed by IPIPS (Lawden and Pearce 1980). This was complemented by a database system (G-EXEC), by CORE graphics, and by the online image manipulation facilities provided by the I2S. The greatest

single problem was the partial incompatibility of these systems, particularly in their different parameter retrieval routines.

3. Implementing TAE.

The target system to replace the existing software consisted of a division of the VICAR labour between TAE (handling parametering) and BISHOP (a superset of VICAR consisting of a selection of the image-processing subroutines and utilities) (Grove 1985). In addition the parametering routines associated with the other parts of the system were to be hidden beneath a TAE interface, enabling a common mode of user access to all parts of the system. In developing this system a major problem became apparent in that IPIPS simply did not have the manpower to fully convert large parts of the system to TAE in a short space of time. In the interim the less experienced users were having to cope with software wherein some programs were available only under the old arrangement and others only under the new. This is a problem that may potentially affect any system with limited resources.

To escape this problem a two stage implementation philosophy was used. The source of the Vicar programs was initially left untouched, but TAE was used on top of this to enable users to take advantage of the associated menus, tutor modes and second level help. This was done by generating a DCL VICAR command line within a TAE Procedure from concatenated parameter names and values. This command line was then used to run the program. Listing 1 provides an illustration of this.

This strategy proved extremely successful in smoothing the transition to TAE. The pressure on the systems staff to make the change as rapidly as possible was lifted as large numbers of programs could be made available to users in a relatively short period of time. The staff could then replace the temporary Procedures with Process PDFs and upgrade the Fortran source code from VICAR to BISHOP in their own time, and in a way that was invisible to the users. This process is still continuing.

4. a. Further enhancements.

In addition to this interim application, this method led to the implementation of many DCL commands (such as Link, Run etc.) within TAE. Again this was done through Procedures. This was found to have two advantages. Firstly, the user did not have to go through the sequence of commands DCL - <command> - TAE every time he wished to use a sequence of VMS commands. More importantly, only selected parameter qualifiers were made available in TAE. In the Link command, for instance, only /MAP, /DEBUG and /CROSS were used. This guided novice users who may have been lost within the extensive VMS help towards the parameters most likely to help them. See listing 2 for an example.

In addition to the general upheaval, the transition to TAE created some particular problems, and opportunities to remedy some old difficulties. The most ambitious development was a model-independent I2S driver. Applications programs link to a dummy shareable image at link time and to the appropriate (Model 70 or Model 75) real shareable image at run time. A TAE procedure selects the correct image by setting an appropriate global symbol that acts as a pointer. The shareable images associated with each I2S contain subroutines with the same name and parameter list, so that a program calling just these model-independent routines will work equally well on either I2S. The I2S primitive subroutines are different for the two models, and previously a user would have had to write a different program for each I2S. These differences are now hidden from the user. In principle a complete graphics system may be built in this way, with a set of shareable images corresponding to the output devices. This has a number of advantages. Firstly, there is a considerable saving in space since the executable images no longer contain the code to drive all the devices. Secondly, upgrades to individual devices' code may be made without rebuilding the whole system. Finally, new output devices may be added simply by adding new shareable images to the set, without disturbing the existing code. The CORE system might particularly lend itself to this architecture on the VAX, and IPIPS might consider this development at a later stage.

The implementation of TAE enabled simple menus to be constructed as another major development. Even the most experienced users were unfamiliar with all of the 150 plus applications programs available at IPIPS. Helping new and external users to find the programs they needed had long been a problem, and TAE menu trees arranged by application has made a great difference in this area.

In another area, TAE tutor mode has had an unexpectedly beneficial effect. VICAR programs may frequently have 10 or more optional parameters, and occasionally as many as 100. The well-known dislike of ploughing through many pages of hardcopy had always meant that users rarely used the more obscure parameters, although these were often very useful. Tutor mode has greatly increased user awareness of the system's capabilities, and this has in turn made for a more productive usage.

b. An outstanding problem.

The dual upgrade to Version 4.1 VMS and Version 1.3 TAE has induced one problem. Version 4.1 has been observed on many machines to run more slowly than earlier versions. This problem is made worse by the extra processes created by TAE. The observed additional overhead in response time for a given program varies between 5 % and as much as 25 %. The larger values are associated with programs involving user IO, whilst programs that involve mainly calculation are virtually unaffected.

5. Conclusions.

In summary, the TAE system is now essentially fully implemented as part of the IPIPS image processing system. This has been accomplished with only limited resources but the minimum of user disruption. The future development of such systems will surely hinge around the concept of integration. It is essential that the image handling, graphics and database parts of such complex systems are assembled and presented to the users in a coherent fashion. This is a path that systems such as IPIPS, MIPL at JPL, and the LAS all seem to be following. Future systems such as the Galileo HIIPS

system will also need this coherence, and it is apparent that the TAE Executive is currently the best way to provide it.

Refs.

Adams J.R. and Driscoll E.C., "A low cost transportable image processing system," 1st ASSP WKSHP ON 2-D DIG. SIG. PROC., 1979.

Castleman, K.R., "Digital image processing,"
Prentice-Hall, 1979

Grove, L, "TAE and BISHOP in a Teaching Environment," Fifth TAE Users Conference, Maryland (1985).

Hunt G.E., Barrey R.F.T., Clark D.R., Easterbrook M., Gorley R., Marriage N., Muller J-P.A.L., Roff C.E. and Rumball D., "The Interactive Planetary Image-Processing System," IEEE Transactions on Geoscience and Remote Sensing, Vol. GE-23, No 4, July 1985.

Lawden M.J. and Pearce D., "Making the VICAR system portable," J.B.I.S. 33, 369-376, 1980

LISTING 1 - AN IMAGE COPYING ROUTINE

```
PROCEDURE HELP=*
PARM (IN,OUT) TYPE=(STRING,80)
PARM SIZE TYPE=INTEGER COUNT=(0,4) DEFAULT=--
LOCAL COMMAND TYPE=(STRING,130) INITIAL="DCL ACOPY "
LOCAL SIZIN TYPE=(STRING,130)
LOCAL NEWSIZ TYPE=INTEGER
LOCAL I TYPE=INTEGER INITIAL=1
BODY
IF ( $COUNT(SIZE) <> 0 )
     LET SIZIN=" SIZE="
     LOOP
          IF ( I > $COUNT(SIZE) ) BREAK
          LET NEWSIZ=SIZE(I)
          LET SIZIN="&SIZIN" // "&NEWSIZ"
          IF ( I <> 4 )
              LET SIZIN="&SIZIN" // ","
          END-IF
          LET I=I+1
     END-LOOP
     LET COMMAND="&COMMAND" // "&SIZIN"
END-IF
DCL ACOPY:==$P$:[VICLIB.PROGRAMS]ACOPY.EXE
&COMMAND IN="&IN" OUT="&OUT"
END-PROC
.TITLE
A PROC  to copy one image to another.
.help
                         ACOPY

   This PROC was produced as a programming example of memory
```

mapping images into a process's own virtual address space.

However perplexing the above may sound, it is a simple program to use, with only the three parameters.

```
ACOPY IN="FRED" OUT="JIM" SIZE=(1,1,256,256)
```

This will copy the first 256 samples of the first 256 lines from image FRED to image JIM.

```
  ....
.END
```

LISTING 2 - THE LINK PROCEDURE

```
PROCEDURE LINK HELP=*
PARM FILENAME TYPE=(STRING,100)
PARM (DEBUG, MAP, CROSS)    +
  TYPE=(STRING,3) VALID=("YES","NO") DEFAULT="NO"
LOCAL MAPFILE TYPE=FILE INITIAL=-- COUNT=0:1
LOCAL DCLCOM TYPE=(STRING,132) INITIAL="LINK"
BODY
  IF (MAP = "NO")    LET DCLCOM = "&DCLCOM" // "/NOMAP"
  IF (MAP = "YES")   LET DCLCOM = "&DCLCOM" // "/MAP"
  IF (DEBUG = "YES") LET DCLCOM = "&DCLCOM" // "/DEBUG"
  IF (CROSS = "YES") LET DCLCOM = "&DCLCOM" // "/CROSS"
  WRITE "COMMAND IS &DCLCOM"
  DCL &DCLCOM &FILENAME,TAE$BISHOP:USER.OPT/OPTIONS
  IF ($SFI < 0)
    WRITE "Error in linking &FILENAME"
  ELSE
    WRITE "&FILENAME correctly linked"
  END-IF
END-PROC
.title
                    LINK programs with TAE.
.HELP

  This PROC can be used to LINK program modules (object modules
output by any VAX compiler) together with the TAE kernel to
produce executable programs which will run under the TAE command
language.

          ....

.END
```