VLB ARRAY MEMO No. 457

To: VLBA Computer Coordination Group
From: Craig Walker
Subject: Review of 21 May 1984 meeting.

Those present were:
Charlottesville: Cotton, Benson, Wells, Burns.
Caltech: Pearson, Ewing, Fort.
VLA: Walker, Clark

No formal agenda was distributed prior to the meeting.

Last meeting we decided to try to follow the time/angle format chosen by the group led by Rick Fisher that is trying to coordinate the development of on-line systems in the observatory. Clark contacted Fisher and determined that the format uses h,m,s and d,',," to specify times and angles. Numbers without units are assumed to be radians. Examples would be 12h15m8.5s, 25.6m, 56d15'13.5" etc. No format for day was specified and Clark recommends following the VLA DEC10 convention which has the form yymmmdd (eg. 85MAY21). The month can use any number of letters as long as it is unique.

During the meeting, we decided to adopt the above scheme. However since then Pearson has pointed out a possible serious defect. A number of systems use quote marks to delimit character strings. The form specified for angles will give trouble in these systems. Does this affect us?

Most of the meeting was spent discussing software management and in particular the DEC products CMS and MMS. The documentation for these products only recently arrived in Pasadena so there has not been time for a careful study. However Pearson wrote his initial impressions just prior to the meeting and that document is attached. There seemed to be general agreement that MMS would be very nice to have and we will probably use it. CMS seems attractive but expensive. Also it is less clear that we need it as opposed to the VLA cyclic update scheme or the AIPS checkout scheme. CMS was considered for AIPS a few years ago and was rejected for reasons that Don Wells could not remember (perhaps price). The group wrote their own system with similar, although much more limited, functionality. It is not clear if the same decision would be made today or if AIPS would switch to CMS now if it were available.

There was a short discussion of possible alternatives. The major possibilities, other than the NRAO systems mentioned above, are utilities provided with UNIX. We probably could use these utilities with one of the UNIX-under-VMS systems. The main argument for such systems is price but there was general argeement that the extra problems that are likely to be encountered relative to the DEC supported product are not worth it.

We will try to decide on the use of CMS and MMS at the next meeting. Clark and any other interested parties will acquire the documentation before then, either from DEC or from Pearson.

Our current plan is to keep the fundamental copies of all modules on one computer, probably the Monitor and Control VAX in Socorro (probably an AIPS machine for the first year or more). This requires that remote sites transfer a copy of any module that they wish to modify over DECnet. Concern was expressed that this would take too long. There was also concern that items such as include files might not be up-to-date at the remote site. If each site must keep the fundamental copies of the programs that are under development at that site, modules cannot be shared easily between systems. AIPS is currently trying to address this problem by keeping the fundamental copies of all code on CVAX but updating the copies of the execute modules, include files etc. on the VLA VAX's every night. That system is only now being installed but experience over the next few months should determine if it works.

Part of the concern about the speed of the DECnet link is the result of heavy use of the general purpose VAX's now on the net. Ewing made the point that cpu's are getting rather inexpensive and it might be reasonable to use extra cpu's in some way to speed the transfers.

Pearson's draft proposal for an in-code documentation scheme was discussed. Clark is concerned that it is too language specific and too powerful. A simple routine that merely extracts lines between some delimiters is all that might be needed. Also the proposed scheme would document every subroutine where Clark would prefer to have only one entry per module where a module (which is a separate file on the VAX) might contain more than one subroutine (any subroutine that might be useful in more than one place should be in a separate module). Some of the functions of the proposed scheme might be duplicated by CMS and/or MMS if those systems are used. Pearson agreed to rewrite the proposal in view of the comments received and the possible use of CMS and MMS.

We should discuss data bases next time.

From: PHOBOS::TJP "Tim Pearson" 21-MAY-1985 12:56
To: @DIST,TJP
Subj: Program development tools

Written in haste: the Computer Meeting crept up on me unawares. I hope you have a chance to read this before the meeting.

Tim

Report on DEC MMS and DEC CMS
T. J. Pearson
21 May 1985

I. MMS

References:

1. AA-P119B-TE VAX DEC/MMS User's Guide.

2. B. W. Kernighan and R. Pike: "The Unix programming environment", Prentice-Hall, Chapter 9.

3. The UNIX Programmer's Manual (under 'make').

Description:

"DEC/MMS (Module Management System) is a tool that automates and simplifies the building of software systems. MMS is useful for building both simple programs, which may have only one or two source files, and complex programs, which may consist of several source files, message files, and documentation. It can rebuild all the components of a system, or only those that have changed since the system was last built. And, above all, MMS is also easy to use - with one command, you can build either a small or a large system."

MMS is patterned after the Unix utility 'make'. The only differences are those required by the different syntax of VMS file names, the fact that it executes DCL commands instead of 'shell' commands to build the system, and the ability to handle modules in VMS libraries. The documentation is very much better than the Unix documentation, though.

MMS works from a description file which tells MMS how the 'target file' (e.g. TEST.EXE) is to be built from the source files (e.g. TEST.FOR, REPORT.FOR). Thus the command

```
$ MMS TEST.EXE
```

would compile TEST.FOR and REPORT.FOR (if necessary) and then link the object modules to form TEST.EXE. This is a very simple example; the full power of MMS is only clear in large tasks. One could for example set up a description file for AIPS which could be used to rebuild all of AIPS, including extracting documentation from the source code and typesetting it. MMS has certain built-in rules (which can be overridden) that allow it to build simple systems without a description file; e.g. the command

```
$ MMS PLOT.EXE
```
with no description file, would compile and link PLOT.FOR (if there is a PLOT.FOR in the current directory; if not, it would look for PLOT.C, PLOT.BAS, etc.). The rules are actually macros: you can provide your own rules for specail processing.

The actions invoked in the description file (compiling, linking, etc) are carried out in a subprocess created by MMS. Alternatively, you can use MMS to create a DCL procedure for later execution.

Comments:

At first sight, MMS might appear to be a trivial program, that could be written in DCL in an afternoon. In a sense, it is trivial (but not that trivial); but it is also extraordinarily powerful. It is much easier to write an MMS description file than a DCL procedure to do the same thing. MMS provides a clear, "structured" description of the task to be done. By just reading the manual, I feel that I have only glimpsed the power of MMS, and it probably has many applications that I have not thought of. "MMS is not restricted to program development - it is valuable for packaging any set of operations that have time dependencies."

I have received comments from a number of programmers who have used 'make' under Unix. Without exception (although this may be a selection effect) they have been enthusiastic about it, and would dearly like to have something similar available under VMS.

Recommendations:

I have no hesitation in recommending that we adopt MMS as a tool for the development of VLBA software under VMS. (In fact, I want it NOW! I can think of many projects which I would use it for; in fact, I would probably use it every time I have to compile a program.) The cost is $2100 (less discount), equivalent to less than two work weeks.

II. CMS

References:

1. AA-L371B-TE User's Introduction to VAX DEC/CMS.

2. AA_L372B-TE VAX DEC/CMS Reference Manual.

3. AA-Z340A-TE VAX DEC/CMS Callable Interface Manual.

Description:

"DEC/CMS (Code Management System) is a program library system for software development and maintenance. DEC/CMS stores source files in a library [actually a VMS subdirectory], keeps track of changes made

to these files, and records user access to the files in the libraries. In addition, CMS supplies two ways to organize files within a library; these formal mechanisms for library organization provide a focus for system design." [The two organizations are 'class' - a collection of related files (e.g. all the source files for one program, or all the documentation files, etc.; the choice is yours), and 'group' - a collection of all the files of the same 'release' (e.g. one might have CMS groups corresponding to AIPS releases 15APR85, 15JUL85, etc.).]

Comments:

The closest analog with which you may be familiar is the AIPS checkout scheme, which allows multiple users to work on the AIPS project without conflict, and records a 'history' file of changes. CMS is basically a generalization of this scheme. It can operate on any ASCII text files.

The commands CMS CREATE ELEMENT, CMS FETCH, CMS RESERVE, and CMS REPLACE basically provide the functionality of the AIPS check out (RESERVE) and check back (REPLACE).

In addition, CMS provides:

(1) Extensive commands to examine the history file for an individual element [file], a group or class [see above], or the whole project.

(2) Storage of the whole project history: any generation [version] of a file or group can be retrieved. CMS saves disk space by storing the text of the first generation and the differences between successive generations.

(3) Annotated file listings, showing who created each line in the file and when.

(4) Incorporation of history as comments in the source code (in the syntax of the source language) at either the beginning or the end of the file.

(5) The ability to maintain two separate lines of descent (e.g. if two programmers really have to work on the same file at the same time) with the ability to merge and semi-automatically reconcile any conflicting changes made in the two lines.

(6) Extensive protection against system failure. Any transaction which fails to complete (e.g. if you run out of disk space or if the system crashes) is 'rolled back' to a consistent state. CMS detects (but does not prohibit) changes made to its files outside of CMS. DEC's propaganda magazine, Insight (April 1985) reports that Lear Siegler used CMS to create a 'flight management system' for the Boeing 737; it contained 2,000 subroutines and 800 common blocks, and was developed in 20 months without loss of a single line of code. 13,000 source changes were recorded by CMS.

(7) Integration with MMS. (MMS apparently uses the 'callable interface'

to CMS; I can't imagine why any other user would need to call CMS as a subroutine.)

Reservations:

In reading the manual, I have found one annoyance: CMS doesn't work over DECNET. The library must reside on the node you are logged in to when you run CMS.

Recommendations:

CMS looks like a nice product. If it didn't exist, we would have to invent it: this is clear from the fact that the AIPS group has already invented it. Is it worth $8,675? I am not sure. We should examine competitive products, and try to judge whether it is better to continue with the existing AIPS checkout procedure, or to change over to a commercial, supported system like CMS.

III. Miscellaneous testimonials

(received over the network grape vine)

From: Dave Meier (DEIMOS::DLM)

Hope the tutorial didn't bore you too much yesterday. The interesting part is really in the code itself. ... Note the MAKE file. Let me know if you are ever able to run MAKE on a VMS machine. It is one of the most useful programs I have seen.

From: John M Sellens <jmsellenswaterloo.csnet@csnet-relay.ARPA>

Faced with the task of moving a large (i.e. many source files) project from PC's to VMS, I am desperately hoping that there is a 'make' on VMS. I wrote one for PC's, but since the first time I even looked at VMS was late last week, I don't think I'm up to the task of writing one for VMS. Can anyone give me any useful info? I looked in the manuals, but was unsuccessful in my search.

From: Mark Johnson <MHJohnson@HI-MULTICS.ARPA>

Well, DEC does have a product called MMS which does a lot of the same work that make does under Unix. The format of the files are the same (you can even name it makefile...) but you will have to use VMS style file names instead of Unix style names. We have it along with CMS (the SCCS alike) and both do a pretty good job. Both cost $$s.

From: dual!qantel!cadsys!mike@Berkeley (Michael Bruck)

DEC have a product called MMS (Module Management System) that apparently is based on Make. I have never used it so I can't pass judgement. They also sell a source code control system

called CMS (Code Management System) which I have used and it's pretty good.

From: K. Hornblach @ Lear-Siegler (DEC Insight, April 1985)

All of a sudden we had complete control of our software. We did 30 releases to Boeing. We made over 13,000 source changes. We know exactly what went on with each module. We have the entire history of the development of that software in the libraries. CMS worked really well. It solved the problem. We use it for everything now, including control of documentation done on the VAX.