From: Dave Shone
Subject: ++Green Bank - Part 2; suggestions for what we should do next
Date: Fri, 20 Mar 92 11:48:03 EST

Here are a few suggestions for consideration by the meeting this afternoon, although I think we'll want more time to think about these and others too (I hope). Although I've been thinking about this for a few days, I've set these thoughts down in a hurry in order that others can be made aware of them before the meeting. In addition, the way forward may in part be determined by the deliberations of the steering committee, next week.

Scope of the work to be done

It still seems likely that we need to define major interfaces in the system (principal data classes, persistence methods, parameter handling etc.) before the group dissipates at the end of June. These will certainly be refined afterwards, but if a significant amount of design remains to be done in these areas, we may have great difficulties.

In order to have confidence in such a design, we must prototype a number of areas. However, a number of areas could usefully be prototyped independently, save for their dependence on a fair amount of infrastructure such as a user interface, realistic persistent data format etc. This sort of "bootstrapping" problem is inevitable in a new system such as aips++, but it represents a major obstacle to achieving our goal for June 30.

I believe we require some sort of infrastructure for prototyping parts of the system such as the "logical" data system (YegSet handling, calibration etc.). Other parts (such as persistent data formats and associated method, user and graphical interfaces etc.) could be designed and prototyped in parallel rather than having to be done beforehand.

YegSets etc., and logical and physical persistence data structures

I shall briefly digress to describe the basis of what I propose.

The Green Bank scheme (with appropriate modifications) seems to be the logical (in more ways than one) starting point for what we do next, but it says little about physical data structures (although a number of ideas were discussed, nothing concrete was devised in this area). This limitation confers a few benefits though:

* The data classes can be implemented in a number of ways, and thus prototyped using an initial implementation which does not support all requirements (such as integration times which vary from one baseline to another).

* The eventual implementation might be able to support a number of physical data formats.

These ideas are not new; Don Wells has already advanced the idea that his DynaFITS concept could be made to cope with a number of different data formats.

I'm still inclined to regard the objects we wish to manipulate as "logical views" of the actual data - we often wish to look at the same data in different ways, with different methods. Whenever possible, we should try to do things this way, if only for efficiency, e.g., if we wish to form new ways of viewing our data, we don't always want to make a new copy of the bits that are common to the different views.

What should we actually do?

Please bear in mind that these are no more than suggestions; I realise that they may provoke a few knee-jerk reactions, and I, too, have reservations. Nevertheless, I think these ideas may be worth exploring, given that we have a great deal to do in a short time.

I suggest that, for the purpose of prototyping the logical data system of the Green Bank scheme, we adopt the physical data formats and parameter system of an existing system, and in the first instance, I propose that should be AIPS. All I/O should be performed using prototype code. A simple user parameter system would read the AIPS Task Data files, and any other inter task communication would be handled by the prototype. Thus, apart from task initiation and user parameter setting, the prototype would not depend on AIPS code. Certainly, there should be no need to use any AIPS routines in the prototype code, although simplifying assumptions might have to be made about the data structures.

The AIPS catalog system might be used to maintain associations between objects, and we might implement a number of the object classes in the Green Bank scheme as new kinds of binary extension tables. In addition, we must attempt to handle single dish data, most probably as a new physical data type.

In parallel to the prototyping of the logical data system, work would continue on the design and prototyping of the "real thing", as well as any other parts of the system for which an interface must be produced by June 30th. The DynaFITS concept is something to consider, possibly in conjunction with the Cotton-Tody variable-length binary tables proposal for conventional FITS. In addition, for ultimate efficiency, we may require a real object persistence mechanism, rather than something which does a logical-to-physical mapping. Only prototyping will tell us the answers to these questions.

Alternatives

My suggestion is based on the fact that AIPS is a reasonable fit to what we wish to do in a prototype, and many of us are fairly familiar with it. However, I believe there are a few alternatives:

* Use FITS disc files - but this will require more work on the parameter system. We might use parts of Tim Cornwell's SDE system for this.

* Use another system such as Khoros. This might be more acceptable to many, since it is closer to what we are aiming for. However, the learning curve and the need to implement new data structures might require more work. I know very little about Khoros; could we implement a FITS-like physical data system and build the Green Bank prototype on top of this?

* Use something else - suggestions?

Dave