NOTE 202 – UPDATED AIPS++ DEVELOPMENT PLAN

Tim Cornwell

March 25 1997

Contents

1 Purpose

The purpose of this document is to lay out the development of AIPS++ from the recently-released V0.8 to version 3.0, to be released in late 1999. It is intended to be broad rather than detailed, and to set out the major events in the development of AIPS++ that we can expect over the next 2 1/2 - 3 years. We should expect that this plan will change in some details as we gain experience.

2 Current capabilities

The recently released version of AIPS++ V0.8 has the following capabilities, as detailed in the release notes.

- AIPS++ has a powerful command line interpreter called Glish. Glish has both a tutorial and a reference manual.
- AIPS++ has a number of general purpose tools that are accessible from Glish.
  - Access to all data in AIPS++ via the `table` module.
  - Catalogs of directories, interpreting the contents to show different types of files, and catalogs of potentially everything in Glish (*i.e.* functions and variables).
  - Plotting of Glish variables.
  - Assorted mathematical capabilities such as statistics, random numbers, polynomial fitting, interpolation, Fast Fourier Transforms and convolutions.
  - Manipulation of measured quantities with units and reference frames either from the Glish command line or via a gui.
  - Display of AIPS++ or FITS images or Glish arrays using the aipsview program.
  - Conversion of images to and from FITS image format, reading and writing AIPS++ images to and from Glish. Statistics, histograms and moments of images may be calculated. Subimaging and padding are both allowed.
  - Logging of messages to a gui window and a table, also printing to a postscript printer or to ghostview.
  - Miscellaneous useful utilities including: obtaining AIPS++ configuration information, the `help` utility, a printer tool (including a gui), execution of shell commands, reading and writing to external files using C-like commands, *etc.*
- The synthesis capabilities of AIPS++ are as follows:

- Filling from and writing to a UVFITS file,
- Filling from WSRT format,
- Full imaging, deconvolution, and self-calibration,
- Joint deconvolution of Stokes IQUV,
- Robust, uniform and natural weighting,
- Flexible windowing in the deconvolution,
- Non-Negative Least Squares Deconvolution,
- Flexible construction of models for self-calibration,
- A sophisticated multi-component model for gain effects,
- Interactive editing,
- Flagging of gain solutions by antenna, spectral window, and time interval.
- Writing of final images to FITS files.

- All user capabilities of AIPS++ are documented via the AIPS++ User Reference Manual, and on-line help is available from the command line.

3 Needed capabilities

This section gives an overview of the needed capabilities in various areas. The aim is to be comprehensive and brief rather than detailed. We cover only areas in the core AIPS++ system. In addition, there are local site-based requirements that are not covered here.

3.1 Synthesis

The development of the synthesis system is laid out in the Updated Synthesis Development Plan: Note 201. This calls for a significant amount of work to be done either improving current capabilities or adding new ones. If the plan proceeds on schedule, the core synthesis system will be frozen by November 1997.

3.2 Single dish

The main driver for single dish support in AIPS++ continues to be the NRAO Green Bank Telescope, which has chosen AIPS++ as the platform for data reduction. Although this is really a site-specific need, it forms the focus of our single dish efforts now that we have completed support for the Parkes Multibeam project.

Development of SDCalc is now the highest priority. Once this is complete (a few months), we need to revisit the overall development of single dish capabilities.

- SDCalc: a GUI-based single dish spectral data browser and analysis package.
- SDImager: package for various imaging operations such as On-The-Fly imaging.
- More sophisticated tools for multi-dimensional data analysis.

3.3 System

The changes needed in system support cover a lot of different topics, ranging from supporting operating systems to optimization of the use of various resources.

- Port to other OSes, most particularly Windows 95/NT.
- Java support. We have yet to determine exactly what this means. The STAG advised waiting further.
- Improved bug tracking: the current level of bug tracking is adequate for our current needs but may be too limited for long-term use.
- Optimization of memory use and speed.
- Optimization of the use of templates.
- Parallelization both at coarse and fine levels. Doug Roberts has written a plan to cover the development of parallelized code inside AIPS++: see AIPS++ Note 203.

3.4 User Interface

It is clear from beta testing that the user interface is very important in determining adoption of the system by end-users. This was also emphasized by the STAG report. This is an area that will need the greatest amount of user-testing.

- GUIs for specific applications, most importantly the synthesis applications.
- Auto-GUI for DOs. This would generate a simple GUI automatically for any DO.
- Stream-lined mini-interpreter (*i.e.* simplified interface compared to Glish?)
- Interface to existing AIPS tasks to enable use from AIPS++ from the command-line or from a GUI. This could also be done for other legacy systems such as Miriad.

3.5 User-level tools

One of the key goals of AIPS++ is to provide a rich set of tools for the end-user for data processing needs that are not met by core applications. The following are needed:

- Improved table browser: faster, cleaner, use-controlled formatting.
- MS browser, with Measures support
- Plotter and table DOs
- MS persistence via FITS, perhaps based on Diamond-Wells-*etc.* format.
- Make system, allowing the data analysis to be specified as a pipeline of operations, which can be repeated if one inputs is changed.
- Miscellaneous DO-based tools, *e.g.* deconvolution, optimization, other numerical methods, image calculator. *etc.*
- Observation planning tools: while scheduling programs are quite specific to a particular telescope, planning tools are of general utility.

3.6 DO system

The binding between C++ and Glish (and thus the end-user) is via the Distributed Object system (see AIPS++ Note 197). Version 2 is now available but a number of additions are needed.

- Application services *e.g.* plotting, display, notification, *etc.*

- Coarse parallelization: running on different hosts.
- Layer on top of CORBA or equivalent.

3.7 Measures

The Measures system is largely complete. The following are needed:

- Integration in rest of system so that applications use the Measures system for conversions.
- Data tables needed for Measures: acquisition, distribution, and use.
- High precision support for VLBI and pulsar work.

3.8 Library

The library requires various miscellaneous additions:

- Rework Image library to add functionality, improve performance, possibly aid programmers for large problems.
- Fortran, Java bindings to common components. This probably means interfaces to the DO system.
- STL conversion: the Standard Template Library is now sufficiently mature that we can adopt it in place of some of our library, thus easing our maintenance load.
- Image display library. The image display library is now under development at ATNF and NCSA. It will be delivered in stages, starting with basic libraries in mid-1997.
- Rework Arrays/Lattices
- Threading of the Table system.

3.9 Glish

We need to freeze the capabilities of Glish as soon as we can. However, some changes will be necessary.

- Replace the transport layer SDS by ACE
- Complete the Glish/Tk widgets interface

- Object-orientation
- Multi-threading

3.10 Documentation

This is an area that desperately needs help from astronomers. Most of the documentation that exists now has been written by the programmer involved. We need help writing cookbooks and tutorials.

- Context-sensitive help
- Establishment of a documentation group
- Cookbooks
- Tutorials for *e.g.* synthesis processing.

3.11 User support

Rather than attempt to design this from scratch, I'd like to get the help of someone with experience in supporting an existing package. The goal for delivering an outline of how this is done should be mid June.

- User support via virtual helpdesk and similar mechanisms.
- AIPS++ User Group (to meeting yearly, starting summer 98)
- Workshops

3.12 Programming support

- Programmer support. By the time of V1.5, we need to provide support for those wishing to program in AIPS++. This includes documentation, templates, tutorials, and actual workshops.
- Code acceptance/redistribution mechanisms: we will start recieving contributed code in late 1998. We need coding standards, policies for accepting code into the core system, mechanisms for distributing code as-is, *etc.*

- Procedures for testing and releases: we need timetables for testing, and releases. We should try to establish a dedicated testing group. In commercial models, testing is performed by an experienced and somewhat separate group. In the past, we've said that we don't have the staffing necessary for this type of activity. However, I think given the scale of AIPS++, this would eventually be a net gain. It would be an ideal specialized task for a site to take responsibility for. Note that the staffing requirements are different: end-users rather than C++ programmers are called for.

3.13 Management

The management structure for the last two years has aided rapid decision making by being centralized. However, in the operations phase that is approaching, the need will be for considered balancing of work in different areas. This requires a different management structure. I think this new structure should include an operations manager with experience in support of a large package, and perhaps an operations advisory board. We would do well to look closely at the operations of other large collaborative software packages such as IRAF.

We also need to provide structure at the group level. If we are to have a testing group, as suggested above, it could well be centered at one site. Conversely a user support group would do well to be spread around the world with one or two people in each 8 hour chunk of timezones. We must determine what role will be played by the current project center.

Commitments to provide programming and support personnel have to become stronger once operations commence since user support cannot be deferred or rescheduled as we now have to do in development.

Finally, I'd like to see connections to the consortium astronomers formalized and tightened. One approach, is that at every site, an astronomer should be designated as being the key scientific contact. I'd like to see a commitment from the consortium members that each scientist can be called upon for a certain fraction of time for duties like testing software, advising on specific scientific needs, writing documentation, *etc.* Our experience is that informal contacts are inadequate for this.

These are all changes that must be discussed with the AIPS++ Executive Committee.

4 Timeline

Providing a detailed timeline covering detailed delivery dates for all the above is clearly not feasible. However we can provide an overall timetable showing the broad outlines of when we expect to achieve the various capabilities outlined in the previous sections. I intend that we will use this broad outline in determining target dates in our current approach to short and medium term planning.

Some continuing infusion of personnel has been assumed, mostly in the area of operational support provided by people currently supporting existing packages. We also assume that more astronomers will become involved with AIPS++ to provide help on documentation.

AIPS++ V0.8 ***Feb 97*** : First beta release

- Continuum synthesis imaging and self-calibration,
- General Glish-based tools.

AIPS++ V0.9 ***May 97*** : Second beta release

- Rectify remaining problems in V0.8
- Spectral line additions to synthesis processing,

AIPS++ V0.95 ***August 97*** : Third beta release

- Rectify remaining problems in V0.9
- Improvements to the user interface, specifically via a number of gui-based applications.

AIPS++ V1.0 ***Q3 97*** : Limited Public Release

- Synthesis with GUI,
- various GUI tools,
- SDCalc

AIPS++ V1.5 ***Q1 1998*** : Code development release

- Code development system (documentation, examples, templates).
- Freeze synthesis interfaces.

AIPS++ Developer's workshop ***Q2 98*** :

AIPS++ V2.0 ***Q2 98*** : Full release

- Completed connected element synthesis package
- Initial VLBI capabilities
- Mosaicing package
- Contributed code

AIPS++ User Group ***Q3 98*** :

AIPS++ Developer's workshop ***Q2 99*** :

AIPS++ User Group ***Q3 99*** :

AIPS++ V3.0 ***Q4 99*** :

- Complete replacement for consortium packages. *Since this is a site-based choice, I suggest it as a target rather than a core milestone.*