NOTE 222 – AIPS++ RELEASE PLAN

Tim Cornwell

30 November 1998

Contents

1 Introduction **3**

2 Synthesis **3**

3 Single Dish (Bob Garwood) **3**
- 3.1 a) Current state of the library 3
- 3.2 (b) The current state of applications 5
- 3.3 (c) Scientific Priorities 5
- 3.4 (d) Deliverables by first release - 15 March 1999 5
- 3.5 (e) deliverables by second release 6
- 3.6 (f) assumptions about progress in other areas 7

4 Image Analysis **7**
- 4.1 (a) Current State of the Library 7
- 4.2 (b) Current State of Applications 8
- 4.3 (c) Scientific Priorities 8
- 4.4 (d) Deliverables by First Release 8
- 4.5 (e) Deliverables by Second Release 9
- 4.6 (f) Assumptions about progress in other areas 9

5 Visualization (David Barnes) **9**
- 5.1 (a) Current status 9
  - 5.1.1 Library 9
- 5.2 (b) The current state of applications 10
- 5.3 (c) Scientific priorities 10
- 5.4 (d) First release: 1999/03/15 10
  - 5.4.1 Objectives/deliverables 10
  - 5.4.2 Display library requirements 12

5.4.3 Dependencies 12
5.5 (e) Second release: 1999/12/01 13
5.5.1 Objectives/deliverables 13
5.5.2 Display library requirements 13
5.5.3 Dependencies 13

6 Glish (Darrell Schiebel) 14

7 Interfaces (Darrell Schiebel) 14
7.1 (d) deliverables by first release 14

8 Measures (Wim Brouw) 14
8.1 (a) the current state of the library 14
8.2 (b) the current state of applications 15
8.3 (c) scientific priorities 15
8.4 (d) deliverables by first release 15
8.5 (e) deliverables by second release 15
8.6 (f) assumptions about progress in other areas 15

9 Documentation (Kate Weatherall, Wes Young) 15
9.1 (a) the current state of the library 15
9.2 (b) the current state of applications 16
9.3 (c) scientific priorities 16
9.4 (d) deliverables by first release 16
9.5 (e) deliverables by second release 16
9.6 (f) assumptions about progress in other areas 17

10 Quality Assurance Group 17
10.1 (a) the current state of the library 17
10.2 (b) the current state of applications 17
10.3 (c) scientific priorities 17
10.4 (d) deliverables by first release 17
10.5 (e) deliverables by second release 18
10.6 (f) assumptions about progress in other areas 19

11 System 19

1 Introduction

The purpose of this document is to outline the deliverables in the forthcoming release.

The priorities for the AIPS++ project are:

- Have a scientifically usable, robust, well-documented system for the common UNIX platforms.
- Functionality visible to the end user is vital.
- Stabilize the development system. Configuring, building and releasing the system should be fully and easily supported.
- Make a plan for future releases. It has to be investigated and decided which changes need to be made and which capabilities need to be added. Changes and additions have to be weighed against the scientific priorities in the project.

In this version of the document, we have concentrated on the deliverables by the first release. The deliverables by the second release will be reconsidered and updated after the first release.

2 Synthesis

See the system development plan by Athol Kemball.

3 Single Dish (Bob Garwood)

3.1 a) Current state of the library

The current state is generally good. There are two deficiencies which need to be addressed before the first release.

FITS This isn't really a show stopper, but it has dragged out for FAR too long. As I see it, we have three options for the underlying FITS classes and one suggestion for isolating this layer. - Copy the current, old, FITS classes to aips and update their documentation. We loose the random access code in the latest FITS classes. - Start with the latest. The amount of work to document this is probably not terribly more than with the old classes. - Wait for Allan Ferris to deliver something with documentation.

My suggestion is that independent of which of these we go with, that we develop (really formalize the parts which are already there possibly with a few additional parts) a layer of our own which provides our classes with the FITS access they need without ever having to know about Allan's classes. This would also allow us to use some other set of FITS classes (e.g. cfitsio if it ever supported tape devices) without disrupting our classes. We could also, I think, then most easily justify not putting any time into documenting the underlying FITS classes ourselves. In the long run, I think this is the way to go - but it is a fair amount of work and we need to have a body to actually do the work.

Fitting Our primary need here is for a multi-component fitting class. Notice I avoided mentioning the word "Gaussian" although that is at least initially what we need. The trick here that isn't available isn't multiple-components, its the ability to hold various parameters constant at will AND the ability to hold relationships between parameters constant (which if the problem is parameterized correctly isn't a problem once you can hold some parameters constant). The need here is to be able to fit molecular hyperfine lines so that, for example, the relative spacing between the components is held fixed and/or the relative height is held fixed. I also suspect that users will want to fit other functional forms. In the long run, it would be nice if users could specify some general functional form either via a glish function or through some specialized syntax so that they could set up a more complex fit that just a simple polynomial, or a sinusoid, or a Gaussian, etc.

Calibration There are clearly needs for the calibration work. This will be merged into the synthesis development plan.

MS columns We are currently archiving Parkes multi beam data from MS version 1 in SDFITS files. For the large number of MS header words which are not mentioned in the SDFITS convention, we store them in the file as MSSUBTABLE_COLUMNNAME, which effectively preserves information about MS version 1 forever. At a minimum sdfits2ms will need to do the right thing here. It would be nice if there were some concise document describing how MS 1 columns map to MS 2 (this might also aid in helping those of us who will need to rewrite our fillers in doing so).

3.2 (b) The current state of applications

One need which hasn't been well or widely expressed is the need to use pieces of aips++ such as the Measures DO without getting significant portions of the rest of aips++ (right now it always comes along with the logger which drags so many other pieces along with it). The GBT folks need to be able to use the Measures DO for their Doppler tracking needs. They want it to be a pure functional interface where they call something which goes out to the Measures system, does its job, and returns a value. I think the DO and a glish to glish connection is perfectly adequate for the job. But right now, that drags along so much other stuff that have opted for now to fall back on using starlink. This might be simply solved by giving complete freedom as to what logger is being used and then to develop a logger which just ignores everything.

I would also like to be able to return a glish table object which corresponds to the table which is being iterated by an SDIterator. Thats easy enough for the table being iterated is the one on disk, but its not easily done when a selection has been applied. One possibility would be for the guts of gtable to be a true DO and not just an app so that I could have that DO in the sditerator app and assuming that the gtable DO had a constructor which worked from a Table, I think I could create one on the DO side. This may be more appropriately a long term target of being able to share DO objects between clients somehow.

3.3 (c) Scientific Priorities

- Support of GBT commissioning and observing
- Replace UniPOPS and comparable single dish analysis tools
- Modern Single Dish Imaging tools
- Combining single dish and synthesis data painlessly

This last is the real prize which justifies the pain of using the MeasurementSet.

3.4 (d) Deliverables by first release - 15 March 1999

My aim is to have all of the GBT commissioning tools available by early summer. Much of that is done or well under way. The only things which are not are imaging tools and holography tools.

The work here will be done by Bob unless stated otherwise.

High Upgrade the various GBT fillers to improve speed (3w)

High dish should be "done" in the sense that it will be a competent replacement for UniPOPS. The possible exception here is calibration where its very difficult to estimate how much time this will require to be able to do basic things which UniPOPS does the old fashioned way. In addition to a number of user interface changes requested or contemplated in dish, this includes:

- plotter (Joe 4w)
- switch to MS version 2 (2w)
- SDIterator understands version 2 (2w)
- multi-component fitting (2w once Fitter is available)
- calculator (Joe 4w)

High Start of calibration effort. At least a few basic calibration schemes need to be in place (4w)

Medium Planning the holography support. I know that this involves coordination with others within the project who also have holography needs. This doesn't necessarily need to be in the first release, but work needs to have clearly begun by then.

Low Initial imaging tool. The real pressing need is for GBT commissioning so that beam maps can be made. I suspect that this is effectively an on-the-fly imaging tool.

3.5 (e) deliverables by second release

We defer further analysis of these targets until after the first release.

- standard calibration package
- simple pulsar reduction
- ready for GBT observing
- dish can handle the GBT spectrometer: large bandwidths, large number of channels.
- OTF imaging done
- holography done

3.6 (f) assumptions about progress in other areas

I think I've outlined above what I think I need from other areas for what reasons. I think someone else besides Joe and myself clearly needs to put some effort into the Holography unification effort if there is any hope of having that done in time. A fall back will be to simply do it ourselves using what we know to do here in GB. That would be a real pity. I'm also worried about imaging.

Hopefully we will have a replacement for Joe in GB by the time of first release or shortly thereafter. But I can't see that person being up to speed and contributing to the cause until into next Summer. The plan then will be to shift some of Joe's commissioning and observer support back to Green Bank and to spread the remaining work around.

4 Image Analysis

4.1 (a) Current State of the Library

Support is present for:

- creation of images
- access to images and subimages (i.e. regions)
- support for (non-linear) absolute coordinates
- optimal iteration through images with user supplied class to operate on pixels
- computation of expressions with images
- conversion to and from FITS
- computation of statistics from an image
- computation of histograms from an image
- computation of moments from an image
- command line and GUI to create and manipulate regions of interest
- some modest capability for the direct convolution of lattices
- display of images (AipsView)

4.2 (b) Current State of Applications

All of the things in the Library are accessable via the image DO except direct general image convolution plus:

- concatenation of images along an axis (should really be in library somewhere)
- simple hanning smoothing (added as stop gap for multi beam)
- placement of an image in another

4.3 (c) Scientific Priorities

- The user interface has very high scientific priority. Although it is a general issue, I have spent a lot of time on it for the regionmanager (and I'm not done yet). I feel that the scientific returns will be greatly enhanced by getting this right. I think the image module is in general poorly packaged at present and needs effort to improve it.
- Some more basic applications ! There isn't actually very much functionality in the image module.
- Polarimetric analysis applications

4.4 (d) Deliverables by First Release

Neil does all of this unless specified. Probably only High priority can be done.

High Image plane fitter: Gaussian to specified region, interactive. (6w)

High GUI for moments calculation (2w)

Medium Unification of coordinates, regions and measures. Currently these three things are loosely coupled only. I want to tighten this up and at the same time improve functionality (e.g. automatic conversion of a region from J2000 to B1950, a well defined coordinates object for use throughout the system) (10w)

Medium Support for regions directly in Lattice Expression Language and from the user interface in the image module (Ger, 2w)

Low Relative coordinates in the Coordinates classes, plus mixed (world/pixel) conversions (3w)

Low Masks are currently only partially in place in images (regions create masks which are available on read). Have mask handling fully implemented and under user control. (4w)

Low General image convolution functionality (2w)

4.5 (e) Deliverables by Second Release

In order of priority:

- polarimetry analysis suite
- model fitting
- regridding (a la HGEOM)

Third item questionable.

4.6 (f) Assumptions about progress in other areas

- Decent UI !
- Some of Ger's time to work on LEL and masks

5 Visualization (David Barnes)

5.1 (a) Current status

5.1.1 Library

Previous to six months ago, the Display Library (DL) supported the display of 2d and 3d AIPS++ Images as pseudo-color raster images in a window built with the Xt toolkit. Built-in event handling providing zooming and animation based on keyboard and mouse input on the Xwindow. Infrastructure to provide built-in colormaps, and color cubes, was present. There was no interface to glish.

Progress on the DL (found in `trial/implement/Display`) during the past six months has been good, and has proceeded along three main lines:

1. Development of a DisplayData class hierarchy, and implementation of two concrete classes: LatticeAsRaster and LatticeAsContour.

2. Partial design and implementation of Glish/Tk "wrappers" for the main DL objects: PixelCanvas, WorldCanvas & WorldCanvasHolder, DisplayData and Colormap.

3. Continuing use and debugging of more and more components of the DL.

Accordingly, the on-screen display of lattice-based data (Images and Arrays) as raster images or contour maps is now possible, and can be controlled from glish. PixelCanvas, WorldCanvas, DisplayData and Colormap objects can all be built and manipulated at the glish command-line.

Very little of the code developed over the last six months has been checked into the system. This is primarily related to a few unresolved issues regarding compiling and linking AIPS++ Library code against native Glish include files and libraries.

5.2 (b) The current state of applications

No applications have been developed with the DL, except development versions and test-beds for the Glish command-line interface to the DL.

5.3 (c) Scientific priorities

There are two scientific driving forces behind the DL:

1. The provision of AIPS++-native display applications that can be tightly coupled to all stages of data processing: editing, imaging and analysis. This includes display "components" that can be "plugged-in" to objects needing display services, but without the "added baggage" of a complete AIPS++ application.

2. The provision of programmability of the DL at the Glish command-line level. The drive here is to enable the typical Glish programmer—or keen astronomer with a new idea and minimal Glish knowledge—to put together innovative visualisation applications that extend or complement the functionality of the applications provided by AIPS++ in 1. above.

5.4 (d) First release: 1999/03/15

5.4.1 Objectives/deliverables

A reasonable objective is to deliver a replacement for aipsview by the First Release. Without going into a detailed specification here, the basic capabilities would include:

High display of lattice-based datasets (AIPS++ Images and Glish arrays) as pseudo-color raster images and contour maps, with overlay capability (1w)

Low "fiddling" of built-in colormaps (1w)

High zooming (1w)

High Glish-controlled animation through planes of >2d datasets (2w)

High basic axis labelling, titling, and color wedge display (4w)

High PostScript output (**see below**) (? 6w)

High Independent and inter-changeable GUI and command-line interfaces, yielding full "plugability" for other AIPS++ applications: eg. simpleimage (2w)

This application would be written in Glish, making use of Glish/Tk DL agents, together with internal DL classes. A significant side-effect of this application would therefore be the provision of example Glish code showing how the DL functionality can be tapped from Glish.

I am also hopeful that prior the the First Release, a number of new DisplayDatas might be developed, based heavily on the existing LatticeAsRaster and LatticeAsContour classes, which provide hue-intensity-saturation component raster images, red-green-blue component raster images, vector maps, and the ability to read the `FLOAT_DATA` and `DATA` columns of MeasurementSets. However, I do not wish to *guarantee* the presence of any of these in the First Release. I do not propose to provide any form of line plotting in the DL before the First Release: PgPlot is providing excellent tools for line plotting already.

Finally, it would be extremely pleasing if we were able to produce a prototype Fourier analysis tool for inclusion in the First Release. This tool would rely on the "plugability" of various components of the aipsview replacement application, combined with bindings on the PixelCanvases, to present two Fourier transform pair images to the user, and enable some rudimentary inspection and modification of the images. I will actively seek assistance in this area if there is sufficient interest.

5.4.2 Display library requirements

The most important parts of the DL which need further development by me to achieve the above target include:

1. Continuing work to provide DL functionality at the Glish level.
2. More flexibility in negotiation of which data will draw: this will enable Images having non-conformant Lattices to display concurrently, if their CoordinateSystems agree.
3. Development and implementation of a useful axis labelling scheme.
4. Continuing development of the application GUI.
5. Documentation of the DL and the application.

5.4.3 Dependencies

- **Technical issues:**
  1. The native Glish includes and libraries will need to be made available within the standard AIPS++ include hierarchy: the `gtk` part of the DL depends on this.
  2. We need to solve the dynamic linking issues relating to statics.
  3. We need to resolve performance issues related to embedding the DL in the same execution thread as the standard Tk widgets.
- **Axis Labelling:** I need to learn about Mark Calabretta's non-linear PgPlot axis labelling routines; there may be a dependency on some FORTRAN/C/C++/PgPlot issues regarding passing C/C++ functions through to PgPlot FORTRAN routines being solved.
- **PostScript output:** Wim has a target to write a PostScript PixelCanvas for the DL. Obviously for any of the output "promises" made above, this class is needed.
- **Mixed coordinate conversions:** Whilst not necessary, these routines are highly desirable for use in position reporting etc., and are on Neil's "list."

5.5 (e) Second release: 1999/12/01

5.5.1 Objectives/deliverables

With the aipsview replacement well in hand, it will be time to concentrate on firstly the continuing enhancement of the programmability of the DL from Glish, and secondly on developing a number of AIPS++ visualization applications. I envisage that some subset of the following applications and plug-ins might be available in the Second Release:

- A position-velocity slice tool
- A Fourier analysis tool, using multi-channel rasters etc.
- A region plug-in which provides region overlay and interactive region editing facilities
- A colormap editor plug-in for DL applications
- A flagging tool making use of the region plug-in and the fourier analysis tool

Leading up to, but more likely beyond, the Second Release, we will develop volume rendering applications.

5.5.2 Display library requirements

- Support for multi-channel rasters
- Many more DisplayData classes
- Plug-in "technology" for the DL
- Extensive programmability from glish
- 24-bit support

5.5.3 Dependencies

- It would be nice if ImageRegions and the Table query language became more closely aligned over time: they are basically providing the same functionality (on different datasets) using different syntax. If they were to coalesce, then writing an ImageRegion selector application would be identical to writing a uv flagging tool, for example.
- Others: data flow paradigm for AIPS++, ...

6 Glish (Darrell Schiebel)

The only changes foreseen here for the first release are:

High Ongoing cleanup of memory leaks (2w)

High Local/global eval (2w)

High Destructors (2w)

7 Interfaces (Darrell Schiebel)

7.1 (d) deliverables by first release

High Investigate use of TiX widgets (1w)

High Decouple Auto-GUI from objectcatalog (Tim, 1w)

High Parameter saving, restoring, transferring (3w)

High Define format for Auto-GUI widgets, implement a couple (3w)

High Improve help presnetation in AutoGUI (2w)

High Standardized parsing (3w)

8 Measures (Wim Brouw)

8.1 (a) the current state of the library

1 Time, 'infinite' directions, 'stable' Earth platform, magnetic field: some minor improvements (more separation of calculations (90% done) and improved iteration between forward/backward (B1950/J2000 really)) necessary.

2 Unstable Earth (Earth tides etc): no support

3 Moving directions: support for major solar system objects almost no support for minor solar system objects no support for proper motions of 'far' objects no support for atmospheric stuff (ionosphere)

8.2 (b) the current state of applications

Applications are supportive only (conversion guis; specialised conversion machinery). Status:

- conversion guis: not available for differential positions on Earth (baseline, uvw). Available for others
- conversion engines: available where requested

8.3 (c) scientific priorities

- Earth tides etc (for VLBI – EVN especially)
- Proper motion
- Ionosphere (for polarisation work)
- minor solar system bodies

8.4 (d) deliverables by first release

High Earth tides (2w)

8.5 (e) deliverables by second release

- Proper motions
- minor solar system bodies
- ionosphere

8.6 (f) assumptions about progress in other areas

- external data support
- improved vectors/arrays

9 Documentation (Kate Weatherall, Wes Young)

9.1 (a) the current state of the library

- Lots of .helpfiles
- utilities for converting .help files into help atoms/table

- recipes LaTeX macros
- LaTeX to HTML translation

9.2 (b) the current state of applications

- Web framework
- Getting started
- Installation guide
- Glish documentation
- User's Reference manual
- Some recipes
- Documentation Search
- Bug reporting/tracking

9.3 (c) scientific priorities

- More recipes
- Consortium instrument tutorials
- Technical editing of existing documentation, for consistent terminology
- Improved presentation of Reference manual material.

9.4 (d) deliverables by first release

- A framework for presenting recipes (1w)
- Clean up of bug tracking system (3w)
- Clean up Reference manual in both terminology and presentation (3w)
- Remove old documents, duplicates (3w)

9.5 (e) deliverables by second release

- Comprehensive tutorial for all telescopes
- PDF availablity of all documents

9.6 (f) assumptions about progress in other areas

Documentation (recipes) must come from our users.

10 Quality Assurance Group

10.1 (a) the current state of the library

We have regular testing of the C++ test progams under the following archectures: sun4sol_gnu, linux_egcs, hpux, alpha_gnu. I have not seen a sgi testsuite report but Wes tells me it is "very close".

We also have a demonstrated ability to produce "stable" souce releases.

10.2 (b) the current state of applications

Not relevant to QAG

10.3 (c) scientific priorities

Not relevant to QAG

10.4 (d) deliverables by first release

- Improve out C++ testing on all supported architectures. (RGM) Currently only the sun4sol_gnu architecture is at 100% we need to get all architectures so that 100% of the test programs pass.

- Regular stable releases (JEN + MH) I expect stable releases to occur on an approximately monthly timescale. These will be both binary and source releases. The binary releases will be for beta testers and the source releases for AIPS++ affiliates (eg GBT, Multibeam project, JCMT etc).

- Depopulating trial. (RGM) Important code in the following module will be reviewed. Important code means the following modules:

  - TableMeasures
  - Lattices
  - Tasking
  - Coordinates
  - MeasurementEquations

  - MeasurementComponents
  - MeasurementSets
  - ComponentModels
  - Images

  We will be flexible about exactly which classes in these modules need to be reviewed. the guideline is that we should not, by the first public release, be distributing code that has not been reviewed.

- Implementation of the Data proposal (WNB)
- Archive of "representative" data. (WNB) We need to maintain an archive containing, at a bare minimum, a data sets (preferably in local format if we are to test the fillers) from each of the consortium members instruments ie WSRT, ATNF-CA, ATNF-Parkes, VLA, VLBA, GBT, BIMA.

10.5 (e) deliverables by second release

The following items will be in the "alpha" stage by the first release. They will be routine by the second release.

- Standardised automatic testing of all glish code. (JEN) We need automatic ways of testing as much of the glish code as possible.
- Mechanism for user testing. (JEN) We need have have people and proceedures for user testing of guis. Ideally we need someone at each consortium site who will test the guis with data from there own instrument(s). This testing will be complementry to the automatic testing described above and only done prior to a major release. It will not be done prior to release of "stable" versions of AIPS++.
- Mechanism for installation testing (JEN) We need to have a group of people who can download and install AIPS++ releases and tell us the problems they have in doing so. This testing will only be done prior to a major release and will ensure that the binaries and configuration scripts are kept are working . It will not be done prior to release of "stable" versions of AIPS++.

10.6 (f) assumptions about progress in other areas

I have assume that we can get time from non-QAG members like Michael Haller and Darrell Schiebel prior to the major releases.

I have also assumed that we can get timely cooperation from sites that support the alpha and sgi architectures and that we will NOT be testing/distributing an NT version of AIPS++ (or any other new architecture).

11 System

See the system plan by Ger van Diepen.