NOTE 226 – AIPS++ RELEASE PLAN V1.3

Tim Cornwell
National Radio Astronomy Observatory
P.O. Box O, Socorro, NM, 87801, USA

October 1 1999

Contents

1 Introduction

The purpose of this document is to outline the deliverables in the forthcoming release of version 1.3. This updates the previous development plan..

The immediate priorities for the AIPS++ project are:

- Maintain and support the released package,
- Establish a high level of responsiveness to user feedback,

- Establish 6-month plan-develop-test-release cycle.
- Implement and executes a systematic approach to the internal testing of the released package,
- Complete another phase of integration of the applications
- Establish completeness in various key applications areas

Release 1.2 is now going out the door. It has a stable Glish, a reasonable GUI, lots of synthesis imaging, a thin-path for calibration, a version of the single dish analysis program dish, extensive image analysis capability, two versatile display programs, and lots of miscellaneous functionality. The documentation framework is now all in place but the documentation itself must be filled out.

We intend to settle down on a 6 month plan-develop-test-release (PDTR) cycle. The time scale is as follows, counting from the date of the previous major release.

+0 weeks Revise planning documents

+2 weeks Finalize planning, assign targets

+2 weeks Start development, testing

+2 weeks Initiate completion of testing of existing capabilities

+4 months Freeze new checkins, start dedicated testing

+5 months Hard freeze, uprev to next versions of release and develop masters.

+5.5 months Cut final CDs

+6 months Ship CDs

The times required for each step may vary but we intend to stick to this regular major release schedule. Note that patches to the previous release will be issued as needed, probably on a 4-6 week interval.

In the medium term (end of 2000), we expect that the third release (probably 1.4) will have most of the synthesis calibration and imaging tools present for all consortium telescopes. Following that we will next target a developer's release.

We must act with discipline in setting and acheiving targets. In particular, at this point in the cycle, we must not schedule too much to be completed before the freeze. Accordingly, I have changed some of the submitted plans to defer some work until the next release.

2 Applications integration (Tim Cornwell)

A major push to integrate applications was started in November 1998. For the most part this initiative was successful and led to considerable improvements in the stability and useability of the system.

2.1 (a) the current state of the library

The basic Tasking capabilities exist can provide a number of services for C++ programmers. These include getting and returning parameters, showing progress meters, asking the user questions, and displaying plots. Passing arbitrary types through the Parameter system is possible but clumsy. The Tasking code lacks some test programs and has not yet been reviewed.

The Glish side of the user interfaces is in reasonable shape but there is still a lack of uniformity and a dearth of good high-level widgets,

2.2 (b) the current state of applications

Not applicable.

2.3 (c) scientific priorities

Some types of interaction with applications are still lacking.

- A systematic approach to state saving and restoration is missing.
- Application state cannot be easily attached to a product such as a MeasurementSet or Image.
- One cannot pass new values of parameters to a running application.
- One cannot interrupt an application other than by killing it.
- Switching between applications is clumsy.
- Context sensitive inputs and help are missing

- Unset is not propagated uniformly *e.g.* image does have unsets that are trapped in the Glish side of the interface whereas other applications use special values for unset.

- History information is not built up.

The main GUI interface is basically satisfactory. The scheme of an autogui controlled by meta-information seems to work for many different contexts. The Copy-Paste idiom for passing information around seems to be viable for simple entities. One problem is that the GUI widgets are incomplete. We need a wide range of more useful widgets:

- General Measures widget or connection to measures gui

- General Quanta widget or connection to quanta gui

- Selection widgets to fill in queries for certain types of table *e.g.* MeasurementSets, Calibration tables.

- Time specification widgets

Passing information around the user interface is still quite clumsy. For example, to get a region from an image on disk requires far too many operations. Part of this is due to the confusion between *e.g.* disk image file or Glish image tool. One obvious approach is that since we know the type of a table (*e.g.* Image, MeasurementSet, Calibration), we should automatically open the corresponding tool when an image is needed; Thus the catalog should be extended to show the connection between a Table and the corresponding tool. In general, converting from a data file to a tool should be a one-click operation both in the catalog and in the spanner.

We are still missing a high-level concept to bind the interface together. We've called this a workspace in previous discussions.

2.4 (d) deliverables by first release

We delivered the following:

High Investigate use of TiX widgets. It seems that binding the widgets will consume a considerable amount of Darrell's time.

High Decouple Auto-GUI from objectcatalog. This and a general restructuring was performed.

High Parameter saving, restoring, transferring. A simple scheme is now in place.

High Define format for Auto-GUI widgets, implement a couple. This occurred.

High Improve help presentation in AutoGUI. We've moved to displaying most of the help via browsers.

High Standardized parsing. The entryparser has improved matters somewhat.

High Improved stability and robustness of the user interface. The GUI and Glish can now run for many hours between failures. The remaining problems are quite obscure and are hard to repeat.

High Memory use: the memory use is still too high. The megaserver brought some improvements but is not a long-term solution.

2.5 (d) deliverables by second release

All of these should be completed early in the development cycle to allow propagation to the rest of the package.

High Protocols for "connecting elements of the interfaces". This is a key issue for easing the use of the interface. It should become less difficult to do common operations. This includes but is not limited to shortcuts in the toolmanager. (Tim Cornwell, 3w)

High Widgets for Measures, Quanta, Selections. The first two are probably best accomodated by special widgets that call the measures and quanta tools directly, rather than by using the measures and quanta guis. The Selection widget should compose a TaQL query for a set of possible columns specific to a given type of table. (Tim Cornwell, 3w) This requires the next:

High Name registry for types of Table (Tim Cornwell, 1w)

High Propagate history information from Glish to ApplicationInfo (Tim Cornwell, 1w)

Medium New C++ services: display of an image, interrupt a running process, get new parameters. (Tim Cornwell, 1w)

Medium Prototype of workspace (Tim Cornwell, 3w)

Low Propagate unset values to C++. This is needed to avoid the horror of special values being used for logically unset values. (Tim Cornwell, 2 w)

2.6 Deferred but important

Change servers.g to use subsequences This is vital to resolve a complete mismatch between the models used. Glish is event-driven, servers is functional, and the GUIs are event-driven. servers should be changed to be event driven as well. I think this is a lot of work to propagate everywhere.

3 Synthesis (Athol Kemball)

3.1 Purpose

The purpose of this section is to update the existing synthesis development plans (Cornwell 1997, Kemball 1998) as part of the current planning cycle for the second release. This section summarizes the current state of the synthesis library and applications, defines the scientific priorities, and identifies deliverables for the second release. Infrastructure required from other parts of the project is explicitly identified. Items which were listed in earlier development plans but which are not yet complete are in the most part carried over to this development plan, unless they have been replaced by higher priority items.

The identification of synthesis goals and priorities is a continuous process, influenced by outside advice and consultation with users and external committees alike. This section identifies the priorities as currently assessed for this time scale. It is not a list of all possible synthesis development priorities for the project.

3.2 Current state of the synthesis library and applications

Data formats The existing MS v1.0 data format was revised to v2.0 to accommodate unified synthesis and single dish processing, and to allow for additional synthesis capabilities. The new format has been adopted, and all changes to infrastructure classes have been made, with the exception of changes to the fillers themselves, although these changes will only be checked in to the system after the first release.

The capability to read and write data in UVFITS format has been expanded in the last development cycle to support a broader set of data fields or sub-tables. The full MS archive utility using FITS binary tables, already existing in the package, has been reviewed and kept current in the last development cycle.

Data fillers Data fillers exist for WSRT, ATCA, BIMA and the VLA. The VLA filler was added in the last development cycle; the existing fillers have been updated in the same period. The library infrastructure developed for the VLA filler was designed to allow re-use for future fillers, including the one required for the VLBA.

Uv-data visualization and editing A development effort with ATNF in the area of uv-data visualization has led to a requirements document for these capabilities, a design within the current Display Library (DL), and some preliminary implementation code. This uncompleted item is a major priority in this plan.

A preliminary data lister has been developed in C++, but requires integration and expansion. Simple editing capabilities have been added to `visplot` and a new editing tool `flagger`.

Calibration Basic cross-calibration suitable for connected-element arrays has been added to the system. This has included the use of an expanded calibration table format covering both image-plane and uv-plane calibration components, as well as other capabilities. Basic flux bootstrapping utilities have been added to the calibration system, for cases where that is appropriate. Calibration data visualization and editing is rudimentary, and awaits progress in the general area of uv-visualization described above.

Image-plane calibration has been significantly expanded with the addition of more versatile primary beam models and corrections as part of the mosaicing development, which can be specified using the `voltage pattern manager`.

Imaging The mosaicing capabilities in the package has been expanded to include enhanced primary beam corrections and weighting, as well as new or improved deconvolution methods. A `Lattice`-based Clark CLEAN is now available in `imager`, as well as multi-field Hogbom deconvolution and multi-resolution CLEAN algorithms. Discussions with Pixon LLC regarding use of their code in `aips++` have continued. An image-plane deconvolution tool, `deconvolver`, has been added to the system, and includes MEM deconvolution. In addition, general infrastructure has been improved or expanded in support of mosaicing development.

Wide-field imaging has been added using the Sault-Brouw common projection algorithm, and has been extensively debugged with simulated and real data. It is an integral part of the general imaging code. Preliminary on-the-fly (OTF) code has been developed in prototype form.

An automated wide-field imaging script `dragon` has been developed and added to the system. The imaging wizard `simpleimage` has been upgraded to use more recent infrastructure in the package.

Documentation A user-level cookbook has been written in a first draft, along which will be combined with several contributed recipes.

Simulation The structure and infrastructure used by the existing `simulator` has been upgraded, and it is being revised to accommodate more physically meaningful calibration errors, particularly those required by ALMA simulations.

General Synthesis applications are fully integrated into the new user interface provided by the `toolmanager` development.

3.3 Development priorities

The preceding plan identified broader scientific use, the finalizing of interfaces and development for time-critical local needs as overall development priorities. It is appropriate to revise these in the current plan, although we do not shift focus from the key identified objectives.

At the current stage of development of the project, and for the planning interval covered by this document, the following key priorities are identified:

- **Broader scientific use:** Significant synthesis capabilities exist within the package at present, but they need to be more widely used in the scientific community. Improvements in the user interface have helped significantly in this effort, as has the provision of higher-level documentation and public outreach. However, additional work is still required in these areas, especially regarding high-level packaging or automated capabilities, and in the area of cookbook documentation.

The most pressing obstacle to scientific use at this point is probably completeness; the addition of new capabilities beyond thin-path connected-element reduction would add more scientific users.

- **Complete connected-element capabilities:** The first release is targeted at providing a thin-path capability for mainstream connected-element reduction, as outlined in the preceding planning document. It is important that this be expanded in the second release to provide more complete coverage of required capabilities. The primary areas in which additional effort is required to meet this goal are in editing and uv-data visualization.

- **Thin-path VLBI capabilities:** The second release needs to add a thin-path VLBI capability. This is regarded, as for connected-element data, as the capability to reduce mainstream VLBI projects end-to-end. The primary components which need to be added to meet this capability include a data filler for the VLBA interchange format, and an initial fringe-fitter. Specialized capabilities, such as those required for geodesy or space-VLBI are not planned in this development cycle.

- **Time-critical local priorities:** It is important that the project meet time-critical targets required for successful AIPS++ use at consortia sites, particularly where the use of AIPS++ is closely integrated into the critical path required for instrument use or operation.

- **Basic automated and real-time imaging:** Packaging of the existing synthesis capabilities in higher-level tools (such as `dragon`) is essential in expanding the scientific base of the package. It is also closely-related to automated imaging pipelines or real-time imaging capabilities. The current availability of a significant number of components within the synthesis package which are required for constructing these tools is an opportunity which needs to be exploited. This was highlighted in the preceding plan, but more effort needs to be expended on this item in the current development cycle, including co-ordination of prototype capabilities being explored within the consortium, to ensure re-use, and not parallel development, of infrastructure components. In particular, we intend to coordinate the planned development of imaging pipelines at the various consortium sites. All consortium members have an active interest in this area.

- **Testing and defect correction:** A significant effort is expected in testing and defect correction in this development cycle. This includes some work on unit testing and code review, although integration testing will have priority.

3.4 Deliverables for the second release

The deliverables for the second release are listed below:

- Data formats
  - Implement MS v2.0: Complete MS v2.0 implementation by modifying all fillers and FITS writers to use the new MS access classes already implemented. Draft documents have been produced to assist developers in the transition. The conscious effort to maximize compatibility with v1.0 should assist this process significantly. **Est. time: 1 month; assigned: MW/others**.
  - MS v1.0 to v2.0 converter: Convert MS files from MS v1.0 to MS v2.0 **Est. time: 1 month; assigned: RM**.
  - Full MS archive format: Complete the upgrade of the existing MS writer which uses FITS binary tables. Define the format in writing. **Est. time: 1 month; GvD**.
  - Concatenation and averaging: Basic tool to average (in time or frequency) and perform basic concatenation of MS files **Est. time: 2 months; RM**.
- Fillers
  - Complete VLA filler: Add full data filtering and shadowing calculations. Read SYSCAL and related data. Support the holography format. **Est. time: 1 month; assigned: RM**.
  - Initial VLBA filler: Basic filler for the VLBA Interchange format. **Est. time: 2 months; assigned: AK/new hire**.
- uv-visualization and editing
  - Basic uv-DL components: Implement the existing design for uv-visualization within the DL to include basic display and editing for visibility and calibration data. Use these components in all higher-level synthesis scripts. **Est. time: 3 months; unassigned: AK/new hire**.
  - Automated editing: A basic automated editing tool, using simple algorithms **Est. time: 1 month; assigned: MW**.

- Calibration
  - Expand cross-calibration: Expand the cross-calibration capabilities as follows: i) improved interpolation; ii) simplified synthesis data selection using modified TAQL syntax; iii) new visibility buffer features, including averaging and selection; iv) incremental calibration, v) coupled solvers. **Est. time: 2 month; assigned: AK/WB**.
  - Initial fringe-fitter: Basic fringe-fitting capability, using coherent detection **Est. time: 1 month; assigned: AK/new hire**.
  - Prototype image-plane solver: Basic interface development for image-plane solver, demonstrated for non-isoplanatic phase calibration. **Est. time: 2 months; assigned: KG/TC**
  - Component models upgrade: Add project capability; enhance user interface; fix identified defects **Est. time: 1 month; assigned: RM**
  - uv-component fitter: Initial version of uv component fitter, working from calibrated data. **Est. time: 1 month; assigned: WB**
- Imaging
  - Complete mosaic weighting: Include Sault-type weighting in deconvolution. **Est. time: 0.5 month, assigned: MH**
  - MEM in `imager`: Add MEM deconvolution to `imager`.**Est. time: 0.5 month; assigned: MH**
  - Optimized FFT size: Optimize Fourier transforms for mosaicing `imager`. **Est. time: 0.5 month; assigned: MH**.
  - Initial OTF capability: Design and implement an initial OTF capability in `imager`. **Est. time: 1 month; TC**.
  - Holography support: Expand existing holography capabilities to a general application for single-dish and interferometry. **Est. time: 3 month; JB**.
- Simulation
  - Upgrade `simulator`: Upgrade the existing `simulator` to: i) optionally create a new MS; ii) add simple visibility and image plane terms; iii) first version of OTF data generation. **Est. time: 3 months; assigned: MH**

- Automated imaging
    - Initial automated imaging tool: Initial version of an automated imaging tool for single-field observations, derived from `dragon`. Simple heuristics, including automatic mask generation. **Est. time: 1 month; assigned: AK/KG**.
- Documentation
    - Add synthesis cookbook chapters: Add additional synthesis cookbook chapters and further editing of the existing chapters. **Est. time: 1 month; assigned: AK/others**.
- Testing and defect identification
    - Testing for all instruments: Systematic testing effort to compare results obtained with the existing synthesis code in end-to-end reduction of data obtained from all consortium connected-element instruments. Documentation of results and procedures. **Est. time: ongoing; assigned: ALL**.
- Coordination
    - Planning and coordination for imaging pipelines: All consortium partners currently have plans to develop imaging pipelines based partly or completely on AIPS++. This requires a kick-off planning meeting involving all interested parties, plus continuing regular coordination meetings. **Est. time: ongoing; assigned: AK**.

4 Single Dish (Bob Garwood)

4.1 (a) Current state of the library

The current state is generally good. However, all of the deficiencies in the library outlined in the previous version of the development plan remain.

FITS This continues to drag out. The random access abilities apparently available in the most recent version we received from Ferris (which has yet to be incorporated into the aips++ system due to lack of documentation and a few concerns about the code) would be very useful to use in the GBT filler.

Fitting We continue to have a need for a general purpose multi-component fitting class.

Calibration No work has been done so far on single dish calibration.

MS columns Parkes multi-beam data archived in the SDFITS convention preserves information about the first version of the MeasurementSet. As we migrate to the second version of the MeasurementSet, sdfits2ms needs to be able to do the right thing as it converts this original SDFITS data into the future versions of the MeasurementSet.

4.2 (b) The current state of applications

General single dish applications consist of dish and its plotter, dishplot and the data iteration client which dish uses, sditerator. This client, sditerator, expects the data to be in a flat Table made directly from a FITS binary table following the SDFITS convention. It can not interact with a MeasurementSet. The plotter is built using the pgplotter widget with some additional features appropriate for single dish data. Dish has the following operations:

- limited data selection
- averaging of several spectra within a data set
- smoothing of spectra
- regridding spectra
- fitting baselines to regions of a spectra
- statistics of regions of a spectra
- saving spectra back to an aips++ table or writing out the data to a file in a simple ASCII format.
- browsing data
- a stack-based spectral calculator

GBT specific tools consist of two data fillers, a tool for viewing an interacting with the engineering log data, and a set of utilities for use during commissioning of the GBT. One filler puts engineering log data into a flat Table and the other fills astronomical data into a MeasurementSet. The commissioning tools remain largely untested. The astronomical filler can currently only fill DCR and holography backend data.

4.3 (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

4.4 (d) Deliverables by first release - 1 October 1999

- The speed of the fillers has been greatly improved and seems to be more than adequate.
- Dish is not yet a competent replacement for UniPOPS. Primarily this is because of the lack of single dish calibration tools. The multi-component fitting operation is the other major missing operation in dish. The calculator operation has gone a long way towards making dish a worthy unipops replacement. The dish plotter was successfully changed to use the pgplotter widget underneath.
- Calibration. No single dish calibration tools exist for the first release.
- Holography support. We have supported two holography tests at the GBT. However, no general holography tool is available within aips++. The data has been filled to a MeasurementSet and some preliminary analysis has been done in aips++ but the bulk of the holography analysis for these two tests at the GBT has been in unipops.
- Initial single dish imaging tool. No work has been done on this for the first release.

4.5 (e) deliverables by second release - 1 April 2000

- ready for GBT commissioning and observing. This will be our top priority. The existing commissioning tools satisfy the requirements on paper but they remain largely untested and have not been used by the GBT staff. Due to the difficulty in generating test data for these tools, when commissioning begins it is likely that all of the single dish effort within aips++ will be focussed on these tools for at least the first few weeks of commissioning. It is expected that commissioning will be well under way by the time of the second release. **High Braatz, McMullin, Garwood 8w**

- GBT MS filler. Besides the tools used during the commissioning phase, a large amount of work remains to be done on the astronomical filler. This includes support for MeasurementSet version 2, support for additional backends (especially the spectrometer and the spectral processor) and support for the GBT FITS files which will describe the IF/LO chain (these FITS files have yet to be written or completely defined by the GBT M&C staff). In addition, the filler must be capable of running without significant operator interaction while data is being taken. The output MeasurementSet must be filled automatically. **High Garwood 4w**

- dish ready for regular observing. This includes the following: interact with a MeasurementSet, multi-component fitting, smooth interaction with whatever imaging and calibration tools become available during the next several months, an FFT operation, improve the internals of dish to make it a true tool, and respond to user feedback from the first released version. In addition, some effort will likely need to be expended to ensure that dish is doing the right thing with the GBT spectrometer data. Namely that large bandwidths and large number of channels are handled correctly. **High McMullin 8w**

- Cookbook chapter on single dish processing **Medium, Garwood, 1w**

5 Image Analysis (Neil Killeen, Ger van Diepen)

5.1 Current Status

5.1.1 People

The resources dedicated to image analysis are modest. Currently only about 0.75 FTE (N Killeen) is provided directly on the high level applications. A further 0.3 FTE (G van Diepen) has been available in the last year for low level infrastructure work (lattices, regions, LEL). Finally, there is 0.75 FTE (D Barnes) working on the Display Library and applications, some of which are related to image analysis or used by image analysis applications.

5.1.2 Library

AIPS++ provides access to images via a range of C++ classes. These fall into the following basic categories

Lattices These define and provides access to regular n-dimensional lattices (disk or memory based)

Regions These support image regions-of-interest and image masks.

Lattice Expression Language These classes support calculation of mathematical expressions of lattices.

Coordinates These support non-linear coordinate systems. The sky coordinate conversions are handled with the WCS system (Calabretta and Greisen).

Images These combine lattices and coordinates to define and access images.

FITS classes are provided for converting bewteen AIPS++ images and FITS images.

In addition, there are what we might call analysis classes. The functionality revolves around

- optimal iteration through Lattices with user supplied class to operate on pixels
- computation of statistics and histograms from images
- moment analysis of images
- separable convolution of images
- regridding of images
- Fourier Transforms of lattices
- convolution of lattices
- source model fitting to lattices
- gaussian parameter world/pixel inter-conversion

5.1.3 Applications

The majority of the library capability is available to the user via three *Glish* tools; Image, Regionmanager and Imagefitter. Their actual functionality is listed below under deliverables for the first release.

In addition, there is some small amount of non-library functionality in the Image tool that should be moved to the library:

- concatenation of images along an axis
- placement of an image in another
- simple hanning smoothing of 1 image axis (complements the more general separable convolution function)
- fitting of source models to an image

5.2 Scientific Priorities

Development has been heavily focussed on providing the infrastructure to make application development straightforward. We are now in the position to develop more applications, although some substantial infrastructure work is still required. Of course, development of an application usually means doing the real work in a C++ class with a thin binding to the Image tool, so that application work sometimes appears as infrastructure work.

Image analysis is driven by scientific needs extending over a very broad front; this makes it quite hard to prioritize some of the competing required functionality. However, functionality that will have a large user base is generally developed first.

5.3 Deliverables by First Release 1999/10/01

The image analysis functionality is provided through three tools; Image, Regionmanager and Imagefitter. The first provides basic access and services for images. The Regionmanager manages the creation of image regions-of-interest, and the Imagefitter is a specialized application fitting source models to the sky.

5.3.1 Image tool

The Image tool provides the *Glish* user access to all of the functionality in the Library.

The functionality of the Image tool is as follows.

Conversion Interconvert between AIPS++ images and FITS files Also, more basic structures such as shapes and arrays can be converted to images.

Inquiry Inquire basic information about the image such as its name, shape, type, header information.

Pixel access Access (get and put) directly the pixel and pixel mask values.

Image calculator Manipulate mathematical expressions involving images.

Coordinates Basic access to the coordinate system and inter conversion between pixel and world coordinates.

Convolution Separable convolution by parameterized functions and convolution by a user given kernel.

Analysis Computation of statistics and histograms, moment analysis and fitting of models to the sky.

Display Display the image (raster or contour) with the Viewer. The user can also use AipsView.

Reshaping Images can be reshaped in a variety of ways (subimage, padding, concatenating).

Masks Multiple image pixel masks are supported. They can be set, copied and deleted. Some basic capability is available to change mask values.

Utility A range of utility services are available; open, close, rename, and destroy this tool. View the history, and locking.

Custom GUIs There are some custom GUI interfaces developed for specific functions. These are for handling masks, separable convolution, and moment analysis.

Cookbook chapter on image analysis

Extensive documentation for images module, image, regionmanager and imagefitter tools.

test script for image tool

test/demo programs for the majority of the lattice and image C++ classes in trial. Some classes ready for review.

5.3.2 Regionmanager

The Regionmanager tool manages image regions of interest. It has a custom GUI interface.

The functionality of the Regionmanager tool is as follows.

- Create simple pixel-coordinate based regions
- Create simple world-coordinate based regions
- Create compound regions from simple world and other compound regions
- Create regions interactively (with the Viewer)
- Save and restore regions to and from Tables
- Convert pixel to world regions
- Utility functions

5.3.3 Imagefitter

This tool, written purely in *Glish*, but drawing on many other AIPS++ components (Image, ComponentList, Viewer tools) offers interactive fitting of models to sources in an image.

The functionality of the Imagefitter tool is as follows.

Display The image is displayed with the Viewer providing full control over the display.

Regions Interactive cursor-driven specification of regions-of-interest to be fitted. Regions can be recovered from a Table and automatically refit.

Fits Accepted model fits are stored and returned in a Componentlist tool. Fit parameters are listed and residuals are displayed via an image, histogram and statistics. Model parameters can be held fixed in the fitting process and the fitted model can be subtracted from the image

5.4 Deliverables by Second Release 2000/04/01

The next development cycle will see continued in both infrastructure and applications.

- Improve Imagefitter to handle multiple simultaneous components, and error estimates (using Condon formalism, for example) **High, 4w**
- Develop some very basic dedicated polarimetric analysis tools **Medium, 4w**

- Improve regridding class and bind to image tool **Medium, 2w**
- Develop some more basic services in Image tool (FFT, models) **Low, 2w**
- Implement relative coordinates **Low, 2w**
- Start on unification of coordinates, regions and measures **Low, 2w** ¿

There is some dependency on other people that must be agreed upon. In particular ComponentList [rm] and image overlap a fair amount, and some services I wish to provide in the image tool probably should be supported by ComponentList (the model stuff). Also the ImageRegrid class has been developed by [mh], and partly further developed by him, it may be more effective for him to finish this work than me [nebk] attempt to take it over. Some of [gvd] needed for region and LEL work. Some of [drs] so that pgplotter objects can be passed through to C++ pgplotter.

6 Visualization (David Barnes)

6.1 Current status

6.1.1 Library

Presently, the AIPS++ Display Library (DL) provides an object-oriented approach to data display. The library defines a PixelCanvas interface, of which we have implementations for two devices: X Windows and PostScript. The DL defines and implements a WorldCanvas on top of the PixelCanvas interface. DisplayData objects, of which there are two existing implementations (LatticeAsRaster and LatticeAsContour), can be registered for display on WorldCanvas objects via a mechanism implemented in the WorldCanvasHolder class. A sophisticated colormap management system is available, as well as event handlers for the X Windows PixelCanvas, the WorldCanvas, and the DisplayDatas.

While visualization development in the last 10 months has been mostly directed at applications work, in that time the following improvements have been made to the library itself:

- the implementation of the PostScript PixelCanvas (Harold Ravlin),
- the addition of axis labelling facilities,
- the addition of position and data value reporting facilities, and

- the split of the library into a more modular structure.

6.1.2 Applications

Most visualization development in the last 10 months has been directed towards the delivery of the AIPS++ 'Viewer,' a new image viewer for AIPS++. The Viewer is implemented as a dynamically loaded, Glish/Tk agent-based interface to the DL, and the application ("tool") itself is put together with Glish code. Consequently, the Viewer is available for use as a (quasi) stand-alone tool, or as a module for inclusion in other AIPS++ tools, such as the Imagefitter and Simpleimage. This is what distinguishes the Viewer from previously available visualization programs such as AipsView and MultibeamView.

6.2 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.

6.3 First release: 1999/10/01: Deliverables

For the first release, the objective was to deliver a replacement for AipsView. This objective has mostly been met by the provision of the Viewer tool, which has the following display capabilities:

- multiple display windows, multiple control panels

- display of lattice-based datasets as pseudo-color raster images and contour maps, with overlay capability

- editing of display parameters, eg. line width, line color, colormap, min/max data value, contour levels etc.
- selection and "fiddling" of built-in colormaps
- zooming
- basic animation through planes of >2d datasets
- position and data value reporting
- basic axis labelling and titling
- PostScript output
- Independent and inter-changeable GUI and command-line interfaces, yielding full "plugability" for other AIPS++ applications: eg. simpleimage

The Viewer falls short of full AipsView replacement level in only a few areas, such as animation.

6.4 Second release: 2000/04/01: Deliverables

With the AipsView replacement well in hand, the highest priority for the second release must be to demonstrate that new and innovative capabilities are possible with the existing DL infrastructure. The main goal, therefore, will be the provision of new and interesting DisplayDatas:

1. Design oversight review of UV-related DisplayDatas: A major new thrust for the DL is the development of non-image-based DisplayDatas, as are needed for example in UV plane visualization. David Barnes will be the key design reviewer. **High, ongoing review role, David Barnes**.
2. TrueColor et al. support in X11PixelCanvas. This is essential for acceptance of the viewer as the prime visualization tool. **High, 4w Harold Ravlin?**
3. The addition of HSV and RGB multi-channel overlay facilities for the LatticeAsRaster class is highly desirable. This would enable quite sophisticated representations of data, especially for complex or multi-waveband images. **Medium, 4w David Barnes, 4w Harold Ravlin?**

4. A rudimentary DisplayData for plotting catalog information already exists. Provided the required library enhancements are made, this too could be made available in the second release. **Medium, 4w, David Barnes, 2w Harold Ravlin?**

5. Design and implementation of improved animation capabilities **Medium, 2w David Barnes**

6. Support for masks in viewer. **Medium, 2w David Barnes, Harold Ravlin**

6.5 Deferred

1. The development of an interactive overlay DisplayData would also be very nice. The user would be able to make annotations on the display, and have them drawn correctly as they zoom in or out, and save the annotations to a Table. This DisplayData could be used by various tools to show the location of ImageRegions, or fit parameters, for example.

6.6 Dependencies

- For full implementation of animation facilities, it is important that the target "unification of coordinates measures and regions" be accomplished.

- For progress to be made on the re-worked integration of the various Viewer components within the AIPS++ environment, a global approach to "connecting things up" needs to be developed for AIPS++.

7 Glish (Darrell Schiebel)

The only changes foreseen here for the second release are:

Control of memory use Both glish and glishtk use large amounts of memory. **High, DRS, 2w**

Destructors A mechanism for destruction of glish entities such as closures is needed.**High, DRS, 2w**

Implement binding to TiX widgets . The TiX widgets provide a mechanism for improving the quality of our graphical user interfaces.**Medium, DRS, 4w**

Local/global eval Low, DRS, 2w

8 Measures (Wim Brouw)

8.1 (a) the current state of the library

1. Time, 'infinite', constant directions, 'stable' Earth platform, magnetic field available
2. Moving directions: support for major solar system objects

8.2 (b) the current state of applications

Applications are supportive only (conversion guis; specialised conversion machinery). There are conversion guis available for time, frequency, velocity, positions on Earth and in the sky, and also available where requested (e.g. frequency; uvw).

8.3 (c) scientific priorities

- Non-stable Earth platform: refraction; Earth tides; ionosphere (for VLBI - EVN especially)
- Proper motion
- Ionosphere (for polarisation work)
- minor solar system bodies

8.4 (d) deliverables by first release

See (a).

8.5 (e) deliverables by second release

- Finish comet support **High, Wim Brouw: 1w**
- Refraction **Low, Wim Brouw: 3w**
- Earth Tides **Low, Wim Brouw: 3w**

8.6 (f) assumptions about progress in other areas

- External data support: *i.e.* the data proposal is completed.
- improved vectors/arrays (TJC doesn't know what this means)

9 Infrastructure (Ger van Diepen)

9.1 (e) deliverables by second release (2000/04/01)

The Fitting classes must be finished. These have many different applications.

- Finish documentation **High Wim Brouw: 1w**
- Finish interfaces **Highm Wim Brouw: 2w**
- Glish interface **Wim Brouw: 2w**

A number of changes are needed in the Table system. All of this work will be done by Ger van Diepen unless noted otherwise.

- Prepare TableMeasures for review and move to aips **High, GVD, 1w**
- Adding StandardStMan (disk-based StManAipsIO): From this week on Ger has a student for 5 months who is working on it. **High, GVD, 2w**
- Improving tiled storage manager (performance and functionality): Some inquiry functions are needed (e.g. by Ralph). **High, GVD, 2w**
- TaQL changes to support synthesis selection. **High, GVD, 2w**
- Scaling facility for a Complex column. Needed for data compression of MeasurementSets. **High, GVD, 1w**
- TaQL selection on Arrays and TableRecords. Make the TableExprNode classes suitable for selecting TableRecord's for the HIA ACSIS system. John Lightfoot will write a parser. **High, GVD, 4w**

The following are deferred until the third release (version 1.4)

- Turn gtable into a DO: 2 weeks, Priority 6.
- copying a table (subset) Making a deep copy of a table (subset) is needed for some people. 2 weeks Priority 7

- Allow access to tables without locking: Some people (e.g. TMS) sometimes want access without the locking overhead. 1 week Priority 8
- Implement a ComputedColumnEngine: Needed for tablebrowser. 1 week Priority 9
- TableMeasures must be finished, which has very high priority. (Comments on how much).
- Make addition of real columns to RefTables possible Makes the table system more versatile, but has low priority. 3 weeks
- Full support of removeColumns: Adding a column is possible, but removing not always. Removing a group of columns should also be possible, otherwise one cannot remove a group of columns added with e.g. a TSM. This is needed for Display work. 2 weeks, Priority 3.
- Improve TaQL (support of units, array functions, IIF function, newly created columns). All these items have low priority. Support of units makes life easier for a user. 2 weeks
- Several functions can only be used on a scalar column. It makes TaQL more orthogonal when also applicable to an array. 1 week IIF function is very useful in LEL, thus also in TaQL I think. 2 days Newly created columns make it possibe to do something like SELECT cola+colb FROM ... 2 weeks

10 Documentation (Kate Weatherall, Wes Young)

10.1 Current state

The overall documentation framework is now in place: we have getting started documents, a cookbook, a Glish manual, a user reference manual, a FAQ, background documentation, documentation search, defect reporting and tracking, and a glossary. Virtually of this has been developed by project members, the most notable exception being the "Getting Started in AIPS++" contributed by Anantha.

- Most functionality has associated documentation.
- We have utilities for converting .help files into help atoms/table.
- We have recipes in the form of LaTeX macros.

- The LaTeX to HTML translation is quite fragile and liable to break. We rely upon latex2html which is still undergoing continual revision. Errors in checked in code tend to break the entire build.

- The hyper-links between documents are hard to install and maintain, mainly because of the extra level of indirection in Latex.

10.2 Scientific priorities

- More recipes

- Consortium instrument tutorials

- Technical editing of existing documentation, for consistent terminology

- Improved presentation of Reference manual material.

- Stability and robustness of the documentation building system

10.3 Deliverables by first release

- Clean up of bug tracking system. We changed to use Rational ClearD-DTS for defect tracking. This has been a considerable success and has cost little time.

- Clean up Reference manual in both terminology and presentation

- Remove old documents, duplicates.

10.4 Deliverables by second release

- More stable and robust documentation processing. We expect that this will be based around SGML as the fundamental document source form. We will initiate a pilot project to investigate the implications of this change. The main implication is that our document preparers will have to write in SGML or something that can write SGML. Since little of our documentation has come from outside the current group, we see little to be lost in this change.

- Style guide for writing documentation.

- Check of hyper-links at sites other than Socorro.

- Cleanup and expansion of the cookbook.

- A framework for presenting recipes. This is sorely missing at the moment.
- PDF availablity of all documents. This is straightforward but requires a fair amount of dumb footwork to put in place for all documents.

10.5 (f) assumptions about progress in other areas

Documentation (cookbook chapters and recipes) must come from our users.

11 Quality Assurance Group (Ralph Marson)

Given the limited resources available to individual AIPS++ team members, the QAG has to concentrate its efforts in these areas that will aid in producing a stable system. The major effort will therefore be 'testing'. We view testing mainly as 'pro-active' bug-fixing, rather than the, maybe more visible, 'reactive' bug fixing. We believe that it will have a higher efficiency, and that it has a better chance of keeping homogeneous, reliable code.

This type of testing will include some centralised attending to conformance of rules. The production of test programs can, at a later time, be used during code reviews.

Centralised testing and the writing of test scripts will take a significant fraction of the time of many of the AIPS++ programmers, and in particular the QAG members.

The QAG will not neglect its duties in other areas, namely code reviews and maintenance of coding "rules". But, unlike testing, we do not envisage major new initiatives in these areas over the next six months.

11.1 (a) Current state of the library

Too small a fraction of the current library has been reviewed. Most of the classes (504 to 415) and lines of code (296K to 236K) still reside in trial whereas the majority of the test programs (166 vs 121) are in aips.

11.2 (b) Current state of the applications

Not relevant to QAG.

11.3 (c) Scientific priorities

Improve the reliability of AIPS++, through more rigorous unit testing of glish and C++ code.

11.4 (d) Deliverables by the first release

1. 100% successful completion of all test programs in the aips package for both linux and solaris architectures.

2. Successful completion of the assay for both linux and solaris architectures.

3. Well established code review system

4. Extensive set of templates to guide programmers in what we consider good coding style.

In the above we define linux and solaris to mean RedHat-5.1 and Solaris-2.6. In absence of test machines that are readily accessable to QAG members we cannot guarentee anything about other processors or different versions of the operating systems.

All of the above targets have currently been met with the exception of the second one. There is a continuing problem which we hope will be resolved by the first release.

11.5 (e) Deliverables by the second release

- Initial version of a checklist for testing a release **High**

- List of required hardware and third party software **High**

- guidelines for the writing a glish test script **High**

- test script for every glish function in aips **High**

- successful completion of all glish test script in aips **High**

- test script for 50% of the glish functions in trial **High**

- Complete reference manual documentation for every glish function in aips and trial **Medium**

- Test programs for all classes in the aips package **Medium**

- 80% or better line coverage (and all functions) in the aips package **Low**
- No unexpected errors from Purify for all the test programs in aips **Low**
- Test programs for all classes in the trial package **Low**
- Automated line/function coverage analysis for all test programs. **Low**
- Automated purify analysis for all test programs. **Low**

12 Parallelization (Athol Kemball, Doug Roberts)

12.1 Purpose

The purpose of this document is to update the existing development plan for the AIPS++ parallelization initiative (Kemball 1998) to cover the next six months, which specifically includes the second release planned for 15 March 2000. The current status of the parallelization effort libraries and applications are described, the scientific priorities are listed, and deliverables for the second release are specified. Infrastructure required from the rest of the project is listed separately.

12.2 Current status of the parallelization effort

At present the parallelization initiative has achieved the following goals:

1. Porting of AIPS++ to IRIX and the SGI native C++ compiler. This includes verifying a stable SGI build at the AOC, and ensuring a repeatable stable build at NCSA. Significant progress has been made in the last development cycle to achieve this goal. An Origin200 has been obtained at the AOC, which is used to generate the primary SGI build; parity in IRIX and compiler revisions is maintained between NRAO and NCSA.
2. Development of a applicator/algorithm class structure, which encapsulates the parallel transport layer, for use in the parallelization of existing AIPS++ algorithms. Demonstration of this capability, using MPI transport, for the embarrassingly parallel problem of multi-channel CLEAN deconvolution on an NCSA Origin2000 system.

3. Implementation of fine-scale parallelization of FFT transforms using the SGI parallel library (SCSL), and their evaluation.

4. Implementation of a batch utility and scripter to allow use of the large NCSA SGI Origin2000 systems.

5. Collaboration with the parallel I/O ET team in the NCSA Alliance to successfully instrument AIPS++ I/O using the Pablo libraries. Compilation of I/O instrumentation data for a selection of large-scale runs.

6. Initial imaging of a large multi-configuration dataset taken on M33 by the VLA, to be used as a test case of AIPS++ parallelization capabilities.

7. Initiation of an effort to port AIPS++ to Windows NT, for use of the NCSA NT supercluster. A significant fraction of the AIPS++ standard computational library has already been ported to Visual C++ in this effort.

8. Establishment of a collaboration with the Albuquerque High Performance Computing Center (AHPCC) to allow AIPS++ to be installed on their Linux supercluster in a research project to investigate AIPS++ parallelization performance in cluster environments. This work, along withe the NT port and supercluster work, is part of a CS MS project.

9. Investigation of porting requirements to move to the 64-bit SGI compiler ABI.

10. Profiling of the serial performance of key algorithms targeted for parallelization, including mosaicing and wide-field imaging.

11. Debugging and testing of the wide-field imaging implementation in AIPS++, in preparation for its parallelization.

12.3 Development priorities

The AIPS++ parallelization effort forms part of the Radio Astronomy Imaging Group, which in turn is one part of the Scientific Instrumentation Application Technology (AT) team in the NCSA Alliance. Members of the AT team include NCSA, BIMA and NRAO. The overall goal of the Radio Astronomy team is the development of data reduction capabilities for the demanding imaging applications of radio astronomy. The elements of this goal are:

1. Development of a complete and sophisticated data processing system (AIPS++) for full, final data processing, visualization and analysis.

2. Enhancement of critical tasks in the AIPS++ package to take advantage of high-performance parallel computing environments, such as those available at NCSA.

3. Development of a system for near real-time transfer of data from a remotely located radio synthesis array telescope.

4. Automatic production of first-order images from the observed Fourier visibilities.

5. Archiving of raw data and first-order images.

6. Development of a digital library in which final images are available to the community.

The parallelization initiative has the following overriding goals:

- **Scientific application:** The parallelization effort needs to demonstrate a scientifically useful capability to address the most challenging problems in radio astronomy where supercomputer resources are required. These include wide-field imaging problems at low observing frequencies, mosaicing and the largest VLBI observations, amongst others. In addition, this also includes new algorithms which have not been widely used to date due to limited computing resources.

- **Parallelization infrastructure:** A central goal of the parallelization effort is to ensure that infrastructure is developed within the AIPS++ system as a whole to support parallel and distributed computing without expensive ad hoc modifications. This requires that the parallelization infrastructure be compatible with the overall project design, and also that the mainstream project development consider parallelization when implementing algorithms. It is also imperative that the parallelization capabilities be presented using the same user interface as the conventional package.

- **High-performance computing in AIPS++:** The parallelization effort has a strong vested interest in the serial performance of AIPS++ for problems of the largest size, which are defined to be those with exceptional I/O, memory or CPU requirements. It is considered the responsibility of this group to profile the serial performance in these specialized cases, make any changes required to support these large problem sizes, and optimize overall serial performance in these cases.

12.4 Deliverables for the second release

The deliverables for the second release are listed below:

- Scientific application
  - Production requirements and group allocation: Drafting of a production proposal to the NCSA computer policy committee; subsequent request for a group allocation to accommodate key project processing **Est. time: 2 weeks; DR**.
  - Key project processing: Processing of five key projects (including at least one each of mosaiced, wide-field or large spectral line). Candidates include the existing M33 dataset (Westpfahl), TX-Cam (Kemball), and a selection of low-frequency VLA projects in A-configuration. Generation of user liaison documentation and user support at NCSA **Est. time: ongoing, 1 month per project; assigned: DR/others**.
- Parallelization infrastructure
  - Multi-field parallelization: Implementation of a prototype parallelization for mosaiced or wide-field imaging. Candidates include field- or facet-based gridding, model prediction or residual image computation. **Est. time: 1 month; assigned: AK/KG**.
  - Parallelization of other deconvolution methods: Extend the Clark CLEAN parallelization to other deconvolution algorithms **Est. time: 1 month; assigned: WY**.
  - Statement-level parallelization trial: Trial of SGI OpenMP directives for a test problem. **Est. time: 2 weeks; assigned: DR**.
  - Initial parallel I/O implementation: Implementation of multi-process I/O on the same file, multi-iterator access in a single process, and ROMIO asynchronous I/O using MPI-2. Evaluation of performance in these cases. **Est. time: 1 month; assigned: AK/DR**.
  - Complete NT port of libaips and libtrial: Continue and complete the NT port as described; run a parallel application on the NCSA NT supercluster. **Est. time: 4 months; assigned: PC**.

  - Merging of pimager and imager: Merge `pimager` into `imager`. Requires flawless serial bypass of parallelization code **Est. time: 1 month; assigned: WY**.

- High-performance computing

  - Test suite for large problem sizes: Add a test suite for large problem sizes to improve system reliability for parallelization development; add as a special option to assay.g. **Est. time: 1 month; assigned: DR**.
  - Serial profiling: Continue and document serial profiling of mosaicing, wide-field and spectral line performance for large problem sizes. Identify and eliminate gross serial optimizations **Est. time: ongoing; assigned: KG**.

- System

  - Transition to 64-bit SGI compiler ABI: Continue work to identify and plan for a general AIPS++ transition to true 64-bit operation. Includes use of Insure++ to identify problems of this nature. **Est. time: 1 month; assigned: WY**.

12.5 Development plan: prerequisites

The parallelization development effort may require assistance with the transition to 64-bit compliance throughout AIPS++, depending on the scope of the problem as finally determined.

12.6 References

Cornwell, T., 1997, AIPS++ Note 201, *http://www.aips2.nrao.edu*. Kemball, 1998, previous parallelization development plan, Dec. 1998. Kemball, 1998, Synthesis plan at integration meeting, Nov. 1998.

13 System (Ger van Diepen)

13.1 Current system

AIPS++ can be executed on any UNIX system with X-windows. It has proven to run under SUN Solaris, HP-UX, Linux, SGI/Irix, and Digital Unix on a variety of machines. It requires little extra software. The public domain package Tcl/Tk is needed for the graphical interface provided by glish. Netscape is needed for browsing the documentation.

Developing in AIPS++ requires much more software, all of it available in the public domain area. The following software is needed:

- C++ compiler
- C compiler
- Fortran compiler
- RCS
- GNU make 3.74
- latex and latex2html
- ghostview
- LAPACK
- WCSLIB
- PGPLOT
- Perl 5.003
- Tcl/Tk 8.0

The offical project C++ compiler is egcs 1.1.2, which works fine for all supported UNIX platforms. GNU and egcs have merged again, so the newest g++ compiler is 2.95, which will be adopted in the future by the AIPS++ consortium. New compilers are often based on the EDG front-end. Ports to the KAI and SGI compilers, based on EDG, are mostly completed.
The C++ code is quite heavily templated. Templates are explicitly instantiated, because automatic instantiation led to excessive link times.
AIPS++ has some Fortran77 code, which can be compiled with g77 or a commercial Fortran compiler.

In October 1999 the first release of AIPS++ was distributed on CDs, which contain executables, libraries, documentation, and source code. It is possible to get updates via ftp.
Sites actively developing AIPS++ can obtain and update a development version via ftp.

13.2 Future developments

13.2.1 C++ standard and Compilers

C++ has now an official ANSI/ISO standard. Several new possibilities in the C++ standard are already used in AIPS++.

- Keyword explicit for one-argument constructors.
- True bool data type.
- The local scope of a variable created in a for-statement.
- True exceptions. Note that the AIPS++ code hardly uses AutoPtr to delete objects on the heap. Code should be changed to avoid memory leaks.
- Namespaces.
- Enums and statics in templated classes.

Some other options are of interest and might be used in the future.

- Better templates. They are needed when we want to implement something like Blitz++ to speed up array calculations.
- Standard Library of which STL is an important part. Since all basic classes have already been implemented in AIPS++, the Standard Library is not really needed. However, it may be better to switch to it to make it easy to use future other C++ software. It may also be expected that the classes in the Standard Library are well optimized.
- Casts are done in a better way. They should be used to get safer code.
- Covariant return types. An overloaded function in a derived class can return a derived object.

The AIPS++ consortium sites use the products purify, purecoverage, and quantify to check the code thoroughly. Unfortunately Rational only supports egcs 1.0.3 for these products. Therefore only those C++ constructs can be used that are supported by egcs 1.0.3 or can easily be emulated.

13.2.2 Porting to Windows NT

A port to Windows NT is underway as part of the parallellization efforts at NCSA to use AIPS++ on an NT cluster. Apart from that it is foreseen that NT will be an important platform. It raises the problems how to map the UNIX and X11 functionality to Windows NT.
Windows NT has the POSIX standard implemented, but when using it the Win32 API cannot be used anymore. This limitation is not acceptable.

AIPS++ uses several UNIX system and library calls. The most important are file handling, pipes, sockets, time, environment variables. X11 is used heavily in aipsview. It is also used in the Display Library, but the class structure is such that it should be easy to use classes dedicated to NT.

There are a few public domain and commercially available packages being able to aid in a port. Amongst them are:

- NuTCracker (commercial) supports Unix and X11R5. It can also be used for Windows95/98.
- Exceed (commercial, Hummingbird) supports X11R6 and NFS. It can also be used for Windows95/98.
- Cygwin32 (public domain, Cygnus) provides a UNIX-like API and tools (like gcc, bash). It can also be used for Windows95/98.
- UWIN (public domain, Korn AT&T) is similar to Cygwin32. Its source code is not available.
- Interix (commercial, formerly OpenNT) supports Unix, X, Motif, and NFS. It has the drawbacks that is does not support Windows95/98 and that it is not possible to mix UNIX and Win32 calls.

When doing the port it has to be decided what is the best way to go. There are a few scenario's possible.

- Rewrite UNIX/X11 specific code. This may cost quite some time.
- Do the port using tools only. This may turn out to be (too) expensive when commercial software is needed.
- A mix of using tools and porting ourselves.
- Starting with tools, but slowly moving towards rewriting. NT system and library calls.

A possible porting strategy could be:

1. Handle parts of the basic library not dependent on UNIX or X11.
2. Handle UNIX-dependent library code.
3. Handle glish. Tcl/Tk has already been ported to Windows NT, so the glish graphics should not be a problem.
4. The final step is porting X11-dependent code.

It has to be decided if and which porting tools will be used. Using Win32 calls directly results in best performance, but tools make the porting easier.

As soon as code development will be done on Windows NT, a code management system (integrated with the UNIX side) and a build system should also work on it.

13.2.3 Code Management

The current configuration, code management, and build system has a few problems:

- Support for checking in/out of entire directories (for tables) is lacking.
- Releases are not supported.
- Partial distribution is not supported.

It has to be investigated whether current subsystems have to be replaced by other packages (public domain or commercial).

Another problem is the size of the libaips library. It is already quite large and will get twice as large once the trial code is moved to aips. It requires that the aips package is split into a few subpackages. A requirement is that they are layered and that a lower level layer is not dependent on a higher level layer. This may be hard for DO implementations which tend to use a lot of classes.

Splitting the library can be done in a simple way by having a few packages at the aips level. A more fundamental approach is to have a subpackage layer under aips. That makes it possible that other packages (nrao, dish) are split in subpackages.

An extra directory level could also be useful in the apps tree. One can think of an application group (with maybe some common source files) and in subdirectories the individual applications.

Shared libraries and dynamically loadable code are used to make dynamically configurable code possible. The glishtk part of glish is relying on shared libraries.
Some experiments have been done with using shared libraries to reduce the size of the system. So far they have not been successful, because too many seldomly used code got linked into the shared libraries. Also the use of static variables gave problems. After splitting the libraries another attempt can be done.

13.2.4 Documentation

The current user documentation system, based on latex and latex2html, is fragile and hard to configure. It needs to be stabilized. At the moment Kate Weatherall and Wes Young are investigating other methods like SGML-based systems..

Care should be taken that no latex constructs are used which are not (yet) supported by latex2html. Having to do very frequent latex2html updates can be a pain for some users.

It should regularly be checked if the search facility for the AIPS++ html pages is selective enough.

The code documentation system is based on the Perl script cxx2html which extracts declarations and comments (with special tags) from the header files. It is doing a very good job, but needs some changes to make nicer html output.

13.2.5 Release support

On a regular basis AIPS++ releases are done on CD. In October 1999 the first release was done. Users can get updates via ftp. In order to reduce ftp traffic a mechanism is needed to make the updates selective, i.e. that only the parts needed are ftp-ed. It would also be nice if the dependencies are such that linking a program is based on changes in the source code and not on changes in a library. In that way a program is only linked when really needed, thus also ftp-ed when really needed.

13.2.6 Support of industry standards

There are several industry standards which are of use to AIPS++. It has to be decided if and how they are supported in AIPS++.

- Multi-threading gets more and more common. In AIPS++ it could be used to have non-blocking DO's and to have read-ahead and write-behind in the table system. Later it could also be used for finer parallellization, but it will require some changes to make the AIPS++ library more thread-safe.
- Standards like CORBA,DCOM,OLE,ODMG and Java are getting more important and make it easier to communicate with other software. They are not vital for AIPS++, but when possible it should be tried to support them.

13.3 Priorities for system

After the release in October 1999 two formal releases are foreseen until the end of 2000. The last release is meant to be used by any programmer who likes to develop code using AIPS++ classes.

Before the next release (1.3) the following needs to be done:

1. Makedefs reconcilation **High GVD 1w**
2. Implement data proposal of Wim Brouw **High WB 2w**
3. Automatic tools for patch generation. *e.g.* Have a better dependency analysis to link based on source code changes instead of library changes **High GVD 2w**
4. Investigate ways and advantages of partitioning libraries. **Medium GVD 2d**
5. Improve using native exceptions. To prevent memory leaks, all variables on the heap should be put into an AutoPtr object until their pointer is taken over. **Low GVD 4w**

Before the following release (1.4):

1. Use threads in DO's to make them non-blocking. It requires that static data is thread-safe.
2. Port to Windows NT.
3. See if automatic template instantiation can be used. This is needed when heavily templated classes in Blitz++ or STL are going to be used.

4. Replace some of our classes by classes from the C++ Standard Library. Candidates are String, Complex, Map, Block. Other classes like valarray should also be looked at.
5. Use Blitz++ or techniques used in it to speed up the Array arithmetic.
6. Investigate how CORBA, DCOM, OLE, XML, ODMG, and Java can be used.

14 Other work (Tim Cornwell)

14.1 Support of HIA ACSIS project

This section outlines the deliverables to be expected from AIPS++ in support of the HIA ACSIS project. This follows a dicussion in Socorro on July 13 between Tim Cornwell, Athol Kemball, Darrell Schiebel (AIPS++), Peter Dewdney, Gary Hovey, and Tony Willis (HIA/ACSIS).

Document XSysEventSource (DRS: 1d, due 1 Sept 1999) Current documented example evidently does not compile

Debug Glish Shared clients (DRS: 3w, due 1 Nov 1999)

1. The link operator does not work for shared client when 'from' client is attached to in a script other than the original launching script.
2. A shared client is launched in script a and then attached to by script b. Script b interacts with the client and then exits. If script a is now exited the client continues to run, unattached to any script. The client should die when all scripts to which it has been attached have finished.
3. general improvements to manual in areas relating to distributed clients.
4. Debug shared clients on original or copy of working system
5. class library needed to set up handles for events from multiple scripts for a shared client - i.e. translate most of 'multiplexer.cc' into a class library that can be used with GlishSysEvent.

Allow Glish out-of-band signals (DRS: 3w, due 1 April 2000) How to get an abort, or stop, to the head of the message queue?

User control of Glish polling (DRS: 3w, due 1 April 2000) Allow user to set how often glish polls other systems to see if they are still running

- ie. override 5 sec default, which is currently fixed in the code. The default seems to generate too many spurious warning messages when clients are running on multiple machines.

Catch of "connection terminated" conditions (DRS: 2d, 1 April 2000)
In glish polling of remote machine, how to catch a polling event and tell tasks that a remote machine has vaporized? How to test 'shared client' or 'handle' case that a remote machine has gone away?

Improved Glish documentation of communications and clients (DRS: 1w, 1 April 2000)
A good description of the communication protocol(s) and routines underpinning GLISH and Glish Clients is required.

Appendix 1 - Detailed Development Areas for Image Analysis

This is a fairly detailed description of the areas of development that are expected for the next year or so in Image Analysis. A subset of these will be targeted for the second release (see above). Some of this will not be of general reader interest. The time estimates are generally bare minimums to get something rudimentary going.

- Coordinates
  - The C++ coordinates classes need to be extended to handle relative coordinates.

    1 week [nebk]
  - A C++ engine to facilitate coordinate conversion between different projections and reference frames needs to be built.

    1 week [wnb]
  - coordinate tool

    Create a *Glish* coordinate tool and manager. Currently we have a higher level tool, the coordinate system.

    2 weeks for something rudimentary [nebk]
  - Gaussian Parameters

    Some more work needs to be done to provide C++ classes for general manipulation of Gaussian parameters (e.g. converting from pixel coordinates to world coordinates). In particular, the issue of getting position angles correct in a non-linear coordinate system needs to be addressed.

1 week [nebk]

- Coordinate conversions from Glish

  With the above work, we also need to improve the packaging of the coordinate conversion functions available to the user. They are not very satisfactory currently.

  1/2 week [nebk]

- Error Handling

  There is currently no statistical error handling in AIPS++. This area needs substantial thought and effort. E.g. automatic propagation of errors.

  Big job.

- Filtering

  A lot of things can come under this general heading (including convolution).

  - Fourier Transform

    Develop a class on top of the Lattice FFT classes to handle FTs of images. Binding to image tool.

    1 week [nebk]

  - Filtering

    Other filtering functions.

    weeks

  - Regridding

    The ImageRegrid C++ class needs further development to handle unlike coordinate systems.

    3/1 weeks [nebk or mh]. mh has developed ImageRegrid beyond its checked in version to make progress towards this.

- History

  The image module needs to write history information as appropriate. This needs to be some central facility in AIPS++ which does not yet exist.

  ?? week [drs or gvd]

- Interfaces

  Some effort needs to be extended to improve some of the image tool interfaces. Possibly more custom GUIs, or some actual repackaging of functionality.

- Lattices

  - Improve copyData functiuons to handle data and/or mask

    Some of this is already coded in DOimage.cc and can be moved out. 1 week [gvd]

- LEL

  - Add degenerate axes

    1/2 week [gvd]

- Masks

  Two items here. The Regionmanager needs to be able to manipulate masks. We must also provide general capability for changing the values of masks (e.g. make the central quarter if this mask True). These two things may well be combined.

  2 weeks [nebk]

- Regions

  - Unify coordinates, measures and regions

    The current scheme by which regions and coordinate systems are combined needs to be reworked. We need to use the above mentioned new Coordinate tool and use it to define regions of interest.

    2-3 weeks [nebk]

  - ambiguity

    There is an (unlikely) ambiguity problem with coordinate systems and regions that must be resolved.

    1 week [nebk]

  - New regions

    Generate new region types; ellipsoid, zonal, point.

    2 weeks [nebk]

  - Improved interfaces

    Some work to improve Regionmanager interfaces, both CLI and GUI. For the latter, extension and concatenation needs improving.

    1/2 week [nebk]

  - Eradicate pixel regions

    Maybe try and eradicate entirely our pixel-coordinate based regions, or improve flexibility of compound regions in their ability to combine pixel and world regions

    2 weeks [gvd]

- Rework Code

  - Image Concatenation

    This code needs to be moved from the image DO into library C++ classes.

    1 week [nebk]

  - ImageStatistics, ImageHistograms, ImageMoments

    These C++ classes were written before the Display Library or any of the *Glish* based pgplot plotting widgets were available. The plotting is done with direction access to the C++ pgplotter object. Some effort needs to be put in to rework these so that the plotting is controllable from *Glish*, or that a *Glish* pgplotter object can be passed through to the C++. 2 weeks [nebk]. Something from drs on the pgplotter.

    In addition, IS and IH should be restructured into a base class that works on Lattices, with a derived class dealing with coordinates for images.

    1 week [nebk]

- Polarimetry

  There is no dedicated polarimetry software in the image analysis suite. Although the image calculator can make up for this deficiency at a basic level, some real development work needs to occur here.

  - Easy computation of traditional polarimetric quantities such as polarized intensity (with debiasing), position angle, rotation measure

  - Provide Display Library componentry enabling displays which aid analysis of polarimetric data
  - Include error propagation
  - Tackle more sophisticated polarimetric approaches such as FFT based rotation measure extraction

- Source Models

  - Model fit parameter conversions to world coordinates

    This functionality must move from the image DO into C++ classes.

    Perhaps it should really become part of ComponentList [rm]

    1 week
  - Image Modification

    C++ classes need to be developed to provide the ability to modify an image by a model (e.g. gaussians, disks etc). Binding to images to be developed.

    weeks Probably this should be done in the ComponentList tool [rm]
  - Convolution by parameterized functions

    C++ classes need to be developed to provide general convolution of images by parameterized functions. Probably also high- level *Glish* tools will also need to be developed matching these classes. Binding to image tool to be developed.

    Should share much development with convolution by functions (above)
  - Automatic Source finding

    Some capability should be developed for automatic source finding in images (bottomless pit of activity).

    2 weeks minimum for something rudimentary [wnb]

- Extend the image DO to complex images

  Currently the image DO only works with real images. It should probably be templated to handle complex images too. There is more work in ensuring all of the functions work with complex images than actually templating the DO itself.

  1-2 weeks depending on trouble with application code [nebk]

- Imagefitter

  The basic fitter functions well. It needs to be extended as follows, in order of priority.

  - Fit multiple simultaneous components (mainly interface work)
    2 weeks [nebk]
  - New models (only Gaussian is available)
    1/2 week with above library class development [nebk]
  - Automatic source finding
  - Joint Stokes fitting

- Spectralfitter

  Develop a tool analogous to Imagefitter (or maybe combined with it) to handle fitting of 1-D spectral components.

  1 month [nebk]. Class fitting development should come from Bob Garwood (for dish).