AIPS++ DEVELOPMENT PLAN: Release 1.4

Athol Kemball and Tim Cornwell (eds.)
NRAO

15 May, 2000

Contents

1 Purpose **2**

2 Release priorities **2**

3 Introduction **2**

4 Single-dish **4**
4.1 Priorities 4
4.2 Targets 4

5 Synthesis **5**
5.1 Priorities 5
5.2 Targets 5

6 Applications integration **8**
6.1 Targets 8

7 Build and code distribution system **8**
7.1 Priorities 8
7.2 Targets 9

8 Glish **9**
8.1 Priorities 9
8.2 Targets 10

9 User interface and tasking **11**
9.1 Priorities 11
9.2 Targets 11

10 Basic library **11**
10.1 Priorities 11
10.2 Targets 11

11 Images **12**
11.1 Targets 12

12 Visualization **13**
12.1 Priorities 13
12.2 Targets 13

13 Parallelization and high-performance computing **14**
13.1 Priorities 14
13.2 Targets 15

1 Purpose

The purpose of this document is to define the development plan for AIPS++ release v 1.4, currently scheduled for October 31, 2000.

2 Release priorities

The highest priority for the project at this time is scientific completeness, and planning for release v1.4 has been undertaken in keeping with this objective. Maintaining a sound infrastructure in the project is important however, and infrastructure work has been scheduled for this cycle carefully; primarily in areas which are vital for the long-term vitality of the project, or which are in the critical path for application development.

3 Introduction

Development priorities and targets are listed separately for each major development area in AIPS++. These targets cover only the development cycle through October 2000, and do not include longer-term items, which are tracked separately for consideration in subsequent development cycles. There are 20 weeks before the expected code-freeze, but deductions for defect correction (20%), user support (up to 20%), and where applicable, science time (25%), have been made. The exact numbers vary by developer and institution, and the best estimate has been made in each case. Variability

in these estimates is accommodated by the inclusion of medium prioirty targets, which may need to be carried over into the next cycle. Each target is assigned a priority as high (**H**), which implies that the target is expected to be completed this cycle, and medium (**M**), which is work to be done on an as-available basis. Note also, that not all developers are available to the project on a full-time basis, and have reduced commitments as a result.

Abbr.	**Developer**	**Affiliation**
AK	Athol Kemball	NRAO
BG	Bob Garwood	NRAO
BM	Barry Maguire	NRAL
DB	David Barnes	Swinburne/ATNF
DK	David King	NRAO
DM	Dave Mehringer	NCSA
DS	Darrell Schiebel	NRAO
GvD	Ger van Diepen	NFRA
GM	George Moellenbrock	NRAO
HR	Harold Ravlin	NCSA
JB	Jim Braatz	NRAO
JM	Joe McMullin	NRAO
JN	Jan Noordam	NFRA
KG	Kumar Golap	NRAO
MH	Mark Holdaway	NRAO
NK	Neil Killeen	ATNF
PT	Peter Teuben	BIMA
RM	Ralph Marson	NRAO
RP	Ray Plante	NCSA
OS	Oleg Smirnov	NFRA
TC	Tim Cornwell	NRAO
TM	Toney Minter	NRAO
TO	Tom Oosterloo	NFRA
WB	Wim Brouw	ATNF
WY	Wes Young	NRAO

4 Single-dish

4.1 Priorities

CLI improvement Improving the CLI and scripting capabilities is of direct and immediate benefit to end-users.

Calibration and imaging capabilities The provision of calibration and imaging capabilities which are integrated with the synthesis infrastructure is a high priority.

Support GBT commissioning The continued development of tools required for GBT commissioning remains a priority during this development cycle.

4.2 Targets

Complete dish as a tool Complete the decomposition of dish as a standard AIPS++ tool. (**JM, H, 0.5 wk**).

Bulk processing in Glish Support bulk processing of data in Glish using the current dish reduction functions. (**JM, H, 0.5 wk**).

Consistent use of flags, weights in averager (**JM, H, 1 wk**).

SD imaging using imager Extend and test the current SD imaging capabilities in imager. (**JM, H, 4 wk**).

MS v2 filler conversion Convert sdfits2ms, ms2sdfits, gbtmsfiller and SDIter to MS v2. Includes feedback from GBT observers interface to gbtmsfiller. (**BG, H, 6 wk**).

Spectrometer backend support (**BG, H, 1.5 wk**).

IF backend support (**BG, H, 1.5 wk**).

Initial SD calibration using calibrater Initial calibration examples using a common infrastructure with synthesis. (**BG, H, 3 wk**).

GBT commissioning assistance Continued part-time assistance of GBT commissioning; total time listed here. Includes work on tipper scripts. (**JB, H, 4 wk**).

Improve velocity/frequency conversions Unify velocity and frequency conversions in the plotter and in dish. (**JB, H, 2 wk**).

Consistent use of flags, weights in dish (**JB, H, 2 wk**).

Replace sdimager (**JM, M, 1 wk**).

5 Synthesis

5.1 Priorities

Scientific completeness Significant synthesis capabilities exist within the package at present, but they need to be more widely used in the scientific community. This problem can be addressed through improved vertical integration of synthesis applications, and in the addition of scientific completeness beyond the current thin-path capabilities. In this area, the full completion of connected-element capabilities takes priority over VLBI.

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 imaging The synthesis infrastructure is sufficiently developed to allow the implementation of basic automated imaging capabilities, applicable to both off-line and real-time imaging. It is appropriate during this development cycle to continue initial work in this area in the main package, and to coordinate similar developments at other consortium sites to maximize component re-use in pipeline development.

Automated testing Expanding the automated testing of synthesis capabilities using simulated data is a priority during this cycle, both at the C++ and Glish level. This has benefits in speeding up release testing, and in daily and weekly checks of system integrity for end-users.

5.2 Targets

MS v2 filler migration Migrate all existing fillers to MS v2. These include: i) VLA filler (**RM, H, 2 wk**); ii) BIMA filler (**PT, H, 4 wk**); iii) WSRT filler (**WSRT staff**); iv) ms2scn, ms2iwos (**GvD, H, 1 wk**).

MS v2 FITS definition Complete definition of MS v2 as an external FITS binary table format. (**AK, M, 2 wk**).

MERLIN filler Implement initial MERLIN filler using existing filler infrastructure in AIPS++. Includes familiarization with the AIPS++ library. (**BM, H, 4 wk**).

Additional VLA filler features Add: (i) shadowing; (ii) Tsys and weighting; (iii) complete output filtering; (iv) new weights scheme currently adopted by FILLM. (**RM, H, 4 wk**).

MS concatenation Complete existing work on MS concatenation and averaging. (**RM, H, 4 wk**).

MS data volume control Implement initial solution to the MS data volume problem caused by the MODEL_DATA, CORRECTED_DATA and IMAGING_WEIGHT scratch columns. (**AK, H, 1 wk**).

MS access classes Refine MS access classes for new filler and calibration developments. (**AK, H, 1 wk**).

Automated editing Complete initial automated editing algorithms, including spectral rejection. (**DK, H, 3 wk**).

New fitting classes in calibration solvers Replace existing Newton solvers in calibrater with the new fitting classes now available in the system. (**AK, H, 1 wk, WB, H, 0.5 wk**).

Investigate coupled solver for ATCA Investigate a coupled solver for ATCA data. Includes a comparison with MIRIAD. (**WB, H, 2 wk**).

Calibration table smoothing and re-gridding Implement an initial calibration table smoothing and re-gridding tool. (**HR, H, 4 wk, DM, H, 1wk**).

Ephemeris-based planet flux density calibration Enable flux density calibration based on planet observations, as currently done in Miriad. (**RP, M, 4wk**).

Holography support in imager Complete general holography support in imager, excluding near-field corrections. (**JB, H, 4 wk**).

Imager testing with MS v2 Test all imager capabilities with MS v2 and implement any necessary changes. (**AK, H, 3 wk**).

Optimal-sized FFT's for mosaicing Complete the implementation of optimal-sized FFT's for mosaiced imaging. (**MH, H, 2 wk**).

Simulator error models Expand the analytic error models in the simulator for use in correctness testing. Also, complete initial support of pointing errors(**MH, H, 3 wk**).

Coarse component search Perform a coarse search for component positions in an image. (**WB, H, 2 wk**).

uv-component fitter Initial component fitter in the uv-plane. (**WB, H, 2 wk**).

Automated imaging prototype Complete the vertical integration of synthesis capabilities in map.g, and provide an initial automated imaging utility, automap.g. (**AK, H, 2 wk**).

Pipeline scientific requirements Complete the scientific requirements document for common pipeline architectures. (**AK, H, 1 wk**).

New Getting Results chapters Add new Getting Results chapters for synthesis reduction for specific instruments, including VLA, WSRT, BIMA and ATNF. (**AK, H, 1 wk, TO, H, 1 wk, MW/FB, H, 1 wk**).

Synthesis tests Complete initial test scripts for all consortium connected-element instruments (WSRT, BIMA, VLA, ATNF). (**TO/JN, H, 2 wk, DM, H, 2 wk, AK, H, 2 wk, FB/MW, H, 2 wk**).

Errors in componentlist tables Add support for simple errors (one per existing parameter) in componentlist tables. (**RM, H, 1 wk**).

Ionosphere support in calibrater Complete FJones support in calibrater, including: i) PIM/GPS/ionosphere note; ii) GPS sub-table access classes; iii) finalize Ionosphere and FVisJones classes; iv) Test PIM-based corrections with WSRT data; v) explore secondary PIM corrections. (**OS, H, 12 wk**).

imager tests against simulated data Add imager tests against simulated data for all imaging modes to imagertest, and assay. (**MH, H, 3 wk**).

Unify ImageSkyModel (algorithm/context) Unify deconvolution and imaging context in ImageSkyModel classes. (**MH, H, 4 wk**).

MSLister integration Complete MSLister integration; includes familiarization with related AIPS++ classes. (**GM, H, 4 wk**).

Calibrater infrastructure development Continue enhancement of calibrater infrastructure in the areas of interpolation and parameterization. (**GM, H, 4 wk**).

VLBI FITS-IDI filler Complete initial FITS IDI filler using existing filler classes. (**BM, H, 8 wk**).

Improve mosaicing interface and functionality Initial work on a high-level mosaicing interface, and related utilities. (**MH, M, 2 wk**).

Initial fringe-fitter Complete initial single-band, coherent fringe fitter. (**AK, M, 4 wk**).

6 Applications integration

This includes development aimed at improving the overall integration of applications in AIPS++ in a common framework, using common interfaces and services.

6.1 Targets

Use guientry.g in regionmanager.g Modify the regionmanager custom GUI to use the services provided by guientry.g. (**NK, H, 1 wk**).

Add WSRT Glish components Add profiler, tracelogger, inspect and glishhelp Glish components to the main package. Adjust to conform with integration guidelines. (**JN, H, 3wk**).

7 Build and code distribution system

This covers all system work, excluding maintenance and support of the basic library.

7.1 Priorities

Stability A fundamental requirement in this cycle is the continued provision of a stable build to maximize application development and testing

efficiency. The project is now operational and build failures have a significant impact on applications development efficiency and the ease of user support.

End-user release support With the project in an operational phase, the provision of end-user installation and release update support is a high priority. This includes a mechanism for assembling release patches automatically.

Initial developer support An initial group of external developers is starting to use the system for exploratory development. This group can be supported in the most cost-effective manner by starting to provide formal support for external development on very restricted platforms in advance of the full developer's release, which is planned for 2001.

7.2 Targets

Finish changelog implementation Complete the implementation of the current changelog system. (**GvD, H, 1 wk**).

Prepare initial developer's release Document the makedefs, and provide an installation FAQ for an initial developer's release, targeted at RedHat and SuSe Linux systems of restricted version numbers. Test the resulting CDs. (**GvD, H, 2 wk**).

Automated patch generation Complete a utility to automate patch generation by tracking library dependencies. (**DS, H, 2 wk**).

Implement global data proposal Implement the existing global data proposal using CVSup. (**DS, H, 2 wk**).

8 Glish

8.1 Priorities

Consolidation The work on Glish is in a stage of consolidation. In general, Glish's capabilities are sufficient both for developers to implement higher-level applications and for users to explore their data. The language is also at about the right level; it is capable enough for complex scripts, while still being approachable by general users. The focus for this cycle will be correcting long-standing defects, particularly in the

area of memory leaks and Glish/Tk performance, and in modest new development in Glish.

8.2 Targets

Resolve reference cycle problems This target involves understanding the specifics of how reference count cycles are introduced in Glish, and in devising a general solution to this question which avoids the memory and performance penalties of garbage collection. Reference count cycles are believed to be responsible for many memory leaks currently found in Glish scripts. (**DS, H, 4 wk**)

Bind first Tix mega-widgets This target involves the manual binding of the tab and combo-box Tix mega-widgets to Glish/Tk. These are immediately useful in the end-user interface. A longer-term solution will require an automated method of making mega-widgets available via Glish/Tk, but this work is not included in this target. (**DS, H, 2 wk**).

Understand speed of TCL/Tk and Glish/Tk There is currently a perceived performance problem in Glish/Tk, which is especially pronounced when drawing large GUIs. The specific origin is unknown. This target involves determining the specific cause by direct comparison with TCL/Tk, including investigating what fraction of the speed differential is caused by byte-compiling. If simple remedies are apparent short of multi-threading, this target includes their implementation. (**DS, H, 2 wk**)

Defect backlog A variety of Glish defects remain which are typically long-standing problems which have been neglected due to the work involved in fixing them. They need attention during this development cycle. Included in this list is an investigation of the memory leaks which affect the GBT weather GUI. (**DS, H, 4 wk**).

Local eval() The current eval() function in Glish operates only in *global* scope. This target covers the implementation of a version operating in *local* scope. (**DS, M, 2 wk**).

9 User interface and tasking

9.1 Priorities

Consolidation and evaluation Views on user interfaces are inherently subjective; a wider pool of scientific users needs to use and comment on the current automated GUI system, as implemented in toolmanager, before undertaking a second revision of the user interface. This feedback needs to be actively sought, and reviewed during this cycle. This factor, combined with the limited available resources for toolmanager development, argues for consolidation only during this development cycle.

9.2 Targets

Enhancement of group meta-information Allow function membership in multiple groups; list functions in toolmanager alphabetically within each group; list reserved standard function names in a pre-defined initial group. (**TC, H, 0.5 wk**).

Tool listing by type Allow selection by type in the listing of current tools in toolmanager. (**TC, H, 0.5 wk**).

Constructor in guientry wrench menu Add a standard constructor option in the guientry wrench menu. (**TC, H, 1 wk**).

Single tool per module shortcut Automatically select a tool in the toolmanager initial interface if it is the only tool in the module, and has no global functions. (**TC, H, 0.5 wk**).

10 Basic library

10.1 Priorities

Consolidation The basic library, which covers all fundamental infrastructure, is mostly complete. Development in this cycle has been scheduled in the area of library maintenance, and in the provision of limited new capabilities required for application development.

10.2 Targets

Complete new standard storage manager Complete the new StandardStMan, which will be the default storage manager. (**GvD - H, 1 wk**).

Increase robustness of Tables This target involves continued work in assessing the robustness of the Table system as currently deployed by the TMS on the WSRT telescope, in tracking down any problems, and in fixing them. (**GvD, H, 3 wk**).

Deep copy and actualDesc() Provide additional information on the storage manager in use and the actual table description; allow deep copy of tables. (**GvD, H, 1 wk**).

Remove end_try and Exception.h Final clean-up of exception handling to meet current standard of using native exceptions everywhere. **GvD, H, 1 wk**).

Time comparison function in TaQL Add a time comparison function to TaQL, to specifications provided by synthesis. **GvD, H, 1 wk**).

Code-copping and test functions for Fitting Complete code-copping of the Fitting classes, and add all examples to tests. **WB, H, 2 wk, NK, H, 0.5 wk**).

Fitting interface constraint setting Allow interface constraint setting (manually) to AIPS++ classes. (**WB, H, 0.5 wk**).

Finalize Measures algorithm split-off To ensure accuracy of inverse non-linear coordinate conversions. (**WB, H, 1 wk**).

Measures C++ programming guide Compile a short AIPS++ note on use of Measures by C++ programmers. (**WB, M, 1 wk**).

Measures chapter for Getting Results Chapter in Getting Results describing end-user Glish use of Measures. (**RP, M, 2 wk**).

11 Images

11.1 Targets

Coordsys editor Add a GUI capability to make and edit a coordinate system. (**NK, H, 3 wk**).

Improve Lattice mask handling Current interface is confusing and error-prone. (**NK, H, 2 wk, GvD, H, 1 wk**).

OTF LEL mask Allow a LEL expression to be applied on-the-fly as a mask. (**NK, H, 0.5 wk, GvD, H, 0.5 wk**).

Improved image convolution Add non-separable convolution in (x,y)(z) by a Gaussian function. (**NK, H, 1 wk**).

Improved component convolution and deconvolution Handle convolution and deconvolution by the primary beam; better integration on this point with fitsky. (**NK, H, 0.5 wk, RM, H, 0.5 wk**).

Clean-up of fitsky Clean-up the fitsky code; includes preliminary determination of steps to add 3-D fitting capability. (**NK, H, 1 wk**).

Prepare LatticeStatistics and LatticeHistogram for review Prepare these classes for code review. (**NK, M, 1 wk**).

Vector DisplayData Implement an initial display data class for plotting vectors. (**NK, M, 2 wk**).

12 Visualization

This includes work in the Display Library (DL), and in applications using this library.

12.1 Priorities

DL development transition David Barnes will be leaving AIPS++ during this development cycle, and securing a smooth transition in development in this area is a high priority. This includes training other developers in the DL, and in distributing design knowledge of the library more broadly.

Provision of uv-visualization services The DL has provided excellent support of image visualization, and continuation of the effort to expand this capability to uv-data visualization is a high priority.

12.2 Targets

Initial DL programmer documentation Provision of an AIPS++ note describing the high-level collaboration of the DL classes, the expected and supported services, and the implementation of the DL/GlishTk binding. (**DB, H, 1 wk**).

Multiple GlishTk use Use multiple GlishTk servers to enhance speed. (**DB, H, 1 wk**).

Multi-WCH support Add support for multi-WCH to allow multi-panel displays. (**DB, H, 2 wk**).

Add coordinate system to WC Add coordinate system support to the world canvas. (**DB, H, 1 wk.**).

Support DL programmer training Provide assistance for new DL programmers; visits, e-mail and teleconferences. (**DB, H, 2 wk**).

Improvements in animation and blinking Support multi-WCH and coordinate systems on the WC in animation and blinking. (**DB, M, 2 wk**).

Review current uv-visualization requirements Review and revise the current uv-visualization requirements document. (**TM, H, 1 wk**).

Complete uv-data TB raster display data Complete familiarization with the DL; implement first uv-data time-baseline raster display data. (**TM, H, 8 wk**).

Initial interactive editor using TB display data Complete the first TVFLG-like interactive editor using the TB display data. (**TM, H, 3 wk**).

DL infrastructure familiarization Become familiar with the DL infrastructure. (**DK, H, 8 wk.**).

13 Parallelization and high-performance computing

13.1 Priorities

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.

13.2 Targets

Cluster Linux and IRIX build maintenance Maintenance of the existing NCSA/NRAO builds under IRIX and on the AHPCC Linux cluster. (**WY, H, 3 wk, DM, H, 3 wk**).

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), TXCam (Kemball), and a selection of low-frequency VLA projects in A-configuration. Generation of user liaison documentation and user support at NCSA. (**RP, H, 4 wk, DM, M, 2 wk, AK, H, 2 wk**).

Complete multi-field parallelization Complete implementation of a prototype parallelization for mosaiced or wide-field imaging. Candidates include field- or facet-based gridding, model prediction or residual image computation. (**KG, H, 4 wk**).

dragon revisions for automated wide-field imaging Refine dragon for full wide-field or deep field reduction. (**KG, H, 4 wk**).

Parallelization of other deconvolution methods Extend the Clark CLEAN parallelization to other deconvolution algorithms. (**KG, H, 2 wk**).

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. (**WY, H, 5 wk**).

Complete NT port of libaips and libtrial Continue and complete the NT port as described; run a parallel application on the NCSA NT supercluster. (**WY, H, 5 wk**).

Modify test suite for large problem size Modify the current bigimagertest to use simulated data. (**DM, H, 1 wk**).

Serial profiling Continue to document serial profiling of mosaicing, wide-field and spectral line performance for large problem sizes. Identify and eliminate gross serial optimizations. (**WY, H, 2 wk**).