

AIPS++ Technical Review

National Radio Astronomy Observatory
Socorro, NM
5-6 March 2003

REVIEW PANEL FINAL REPORT

Roger Brissenden (CfA)
Hilton Lewis (Keck), Chair
Andrew Lumsdaine (U. Indiana)
Dave McConnell (ATNF)
Steve Scott (OVRO)
David Silva (ESO)
Doug Tody (NRAO)
Rick White (STSCI)

Table of Contents

<i>Executive Summary</i>	3
1. Change the focus of AIPS++	3
2. Modify the Development Process	4
3. Strengthen the Project Management and the Project Team	5
4. Further Strengthen the Project Scientist Role	5
5. Short-Term Priorities	6
6. Proceed With the Proposed Technology Changes	7
<i>Introduction</i>	8
<i>Review Process</i>	9
<i>Major Recommendations</i>	10
1. Change the focus of AIPS++	10
2. Modify the Development Process	11
3. Strengthen the Project Management and the Project Team	12
4. Further Strengthen the Project Scientist Role	13
5. Short-Term Priorities	14
6. Proceed With the Proposed Technology Changes	15
<i>Appendix A: Charge to the Review Panel</i>	17
<i>Appendix B: Answers to Specific Questions to the Panel Posed by the Executive Committee, January 13, 2003</i>	18
Q1 Goals:	18
Q2 Design:	18
Q3 Architecture:	19
Q4 Components:	20
Q5 Usability:	22
Q6 Process:	24

Executive Summary

A Technical Review for AIPS++ was held at the National Radio Astronomy Observatory in Socorro on March 5-6, 2003. The eight members of the review panel were: Roger Brissenden (CfA), Hilton Lewis (Keck) (Chair), Andrew Lumsdaine (U. Indiana), Dave McConnell (ATNF), Steve Scott (OVRO), David Silva (ESO), Doug Tody (NRAO) and Rick White (STSCI).

The Panel congratulates the project team on the high degree of preparation for this review. The advance material was comprehensive and accessible, covering the major areas of interest to the Panel. The Panel also greatly appreciates the openness and candor of the project team in discussions at the review. We would also like to thank the project team and the NRAO for their hospitality during the intensive two days of the review.

Our charge was twofold: to review the technical suitability of AIPS++ for current and future radio astronomical data processing and to make specific recommendations on how to make AIPS++ a useful and user-friendly data reduction package. However, any meaningful review of these issues must, of necessity, involve some examination of the sociological environment of the project.

A number of significant technical issues have dogged the AIPS++ program from the earliest days, in particular, incomplete functionality and inadequate performance and robustness. In the past, a number of reviews have recommended changes to the program, but these recommendations have not always been adopted. A number of problems in management and technical development have persisted; however, within the last six months some very significant improvements have been made.

This Panel finds that AIPS++ has the *potential* to become a package suitable for general radio astronomical data processing, for current and future instruments, provided some significant changes are made to the management and development process. If these changes are adopted, there are excellent prospects for AIPS++ meeting the challenges of the next generation of facilities currently under development. We feel strongly, however, that continuing on the current path will not yield this result.

The Panel makes a number of recommendations that we feel are crucial to the future success of AIPS++. These recommendations are summarized below, and are explained in more detail in the *Major Recommendations* section that follows.

1. Change the focus of AIPS++

Until now, the aim of the AIPS++ project has been to develop a general-purpose package for the analysis of radio astronomy data. After 10 years of development, the system is still not yet at an appropriate level of maturity to deliver to the general user community. We believe that the major reason for the lack of delivery has been the focus on producing a general-purpose system at the expense of producing software with specific science goals driven by actual radio astronomy

analysis use cases and data. It is now necessary to change the focus of AIPS++ from a system aiming to be a general-purpose radio astronomy package to a system that is developed to meet the strategic goals of the Consortium.

In order to meet the AIPS++ goals, the approach to development must be driven explicitly by the needs of the current and planned major radio astronomy projects. This implies a modified development process consisting of

- science staff identifying use cases and developing requirements (including performance requirements) from the use cases
- development staff flowing the requirements to the software design and developing a build plan with content and delivery tied directly to the project milestones
- a test process that involves science testers using test data based on the use cases
- software acceptance by the relevant science or project lead

This approach is fundamentally different from the current AIPS++ development paradigm.

ALMA is a high-priority project and will be a major driver of the future software development. We see the task of completing sufficient functionality to demonstrate a full set of VLA use cases and a subset of complementary use cases from other Consortium instruments as high priority, in order to complete a significant part of the core functionality required for ALMA.

The support for external non-Consortium users should be de-emphasized until an accepted system can be completed. We recommend taking AIPS++ off-line until sufficient functionality has been demonstrated.

2. Modify the Development Process

In order to deliver software tied to the specific milestones of the Consortium projects, the present software development approach needs to be modified. We suggest the incorporation of a number of developmental steps, detailed later in this report. Although many of these elements are currently present, they are not focused on the delivery milestones derived from the projects.

A critical aspect of developing the AIPS++ build schedule is creating a master schedule tied to the Consortium projects. The AIPS++ master schedule should be consistent with the major project schedules (e.g., GBT, ALMA, LOFAR, ATCA, EVLA).

Explicitly tying the content and deliveries to a master schedule detailing the needs of the projects may result in the abandonment of the current 6-month routine delivery cycle. Such a delivery cycle is more suited to a system in long-term

maintenance rather than development. The recent adoption of mini-deliveries each on 4-6 week timescales is a move in this direction.

3. Strengthen the Project Management and the Project Team

The project team is clearly highly talented, energetic and dedicated. Nevertheless, it is evident that there are a number of weaknesses in the management of the project that must be addressed.

Project management must be strengthened in a number of areas.

- More formal project management techniques should be adopted.
- The Project Manager role is a full-time position requiring a dedicated person; it cannot be shared with other duties and responsibilities. The current practice of sharing the Project Manager role between several individuals, none of whom is fully dedicated to the role, cannot work for a program of this magnitude.
- A clear customer focus must be fostered among the project team.
- Every effort must be made to fill current staff vacancies. The team should be strengthened by the addition of professional software engineers. One of the project staff positions should have the role of software architect, responsible for the overall integrity and consistency of the core classes.
- More effort must be made to utilize full-time FTEs. Fractional FTE allocations should be consolidated wherever possible into full-time staff.

Although this program could benefit from additional personnel, the Panel believes that the current team (utilizing all currently open positions) should first demonstrate their capability to use what resources they have more effectively. We also note that several Consortium partners are currently contributing significantly fewer staff than that required of them, in some cases to the point where their contributions are ineffective. This is a matter that the AIPS++ Executive Committee should resolve.

4. Further Strengthen the Project Scientist Role

The Panel applauds the appointment of a strong and active AIPS++ Project Scientist. The Panel also acknowledges the recent activities of the NRAO AIPS++ User Group (NAUG). The interaction of the NAUG and the project in recent months has been clearly beneficial and demonstrates the need and power of actively involving users in the development process.

The Panel believes the role of the Project Scientist is critical and needs to be expanded in a number of ways. Analogous to the Project Manager, the Project Scientist should be appointed at the Consortium level and be responsible to the AIPS++ Executive Committee. Individual Consortium sites and projects should appoint individual Site Scientists whose activities would be coordinated by the Project Scientist.

In coordination with the Site Scientists, the Project Scientist should:

- Develop science-based use cases for all major Consortium instruments.
- Based on use cases, derive project science requirements (including performance requirements).
- Identify and appoint Consortium staff scientists as use case testers.
- Organize and prioritize tester feedback for project action.
- Verify that project deliverables fulfill use-case-derived requirements.
- Organize development of high-level User's Manuals ("cookbooks").

In coordination with the Project Manager, the Project Scientist should:

- Provide scientific prioritization for all project activity.
- Assure that general project activity is aligned with scientific requirements.
- Certify and approve for public release any new version of AIPS++.

Technical decisions should continue to lie with the Project Manager. The Project Scientist will generate use-case-based requirements, but it is up to the Project Manager to decide how to achieve those requirements.

5. Short-Term Priorities

Development efforts over the next 12 months should focus on the following:

- Significant improvements in reliability and stability
- Performance improvements approaching the performance of AIPS
- Synthesis imaging capability covering many common use cases of the VLA
- Reduced emphasis on graphical user interfaces

The Panel endorses the current systematic approach to investigation of performance issues by benchmarking of components. It seems likely that continuation with the same program will increase performance to the point where it approaches that of AIPS.

There are defects in the design of the many of the GUIs, some related to the underlying implementation of Glish and others to usability issues. However, script-driven processing is adequate to demonstrate the core functionality of AIPS++ at this stage. A Viewer GUI is essential for this mode of operation, but improving other graphical interfaces should have a lower priority for now.

Future releases must include user "cookbooks" for processing data with validated use cases.

The VLA provides an ideal test bed for AIPS++. The effort to identify the VLA scientific use cases needs to be completed along with their systematic testing. Successful achievement of this opens the door for an AIPS++ VLA pipeline, the routine operation of which would give much needed credibility to AIPS++, as well as confidence that it could fulfill its Consortium obligations.

6. Proceed With the Proposed Technology Changes

The current architecture is built around Glish, which implements many features central to AIPS++. However, this is a custom system supported by just one developer; it is not easily able to benefit from the many developments in the rest of the software world. In fact, many of its functions might be better performed by standard components developed over the last 10 years.

A new technology has been proposed to change AIPS++ to an industry-standard open system, capable of using open-source software. This approach can address many issues of robustness, performance and usability that plague the current implementation.

The Panel finds that the case for changing the technology to support future development of AIPS++ is compelling. The proposed architectural change to the framework has technical merit and a proof of concept should be pursued at this time, in parallel with current development. However, it is critically important that this *not* be allowed to distract the AIPS++ team from completing the current development.

Since no design was presented, the Panel was unable to estimate the effort and cost required to make the change, nor indeed whether the candidate technology will in fact meet the needs of the Consortium projects. A design review will need to be held at some future time before the upgrade project can proceed.

Introduction

A Technical Review for AIPS++ was held at the National Radio Astronomy Observatory in Socorro on March 5-6, 2003. The eight members of the review panel were: Roger Brissenden (CfA), Hilton Lewis (Keck) (Chair), Andrew Lumsdaine (U. Indiana), Dave McConnell (ATNF), Steve Scott (OVRO), David Silva (ESO), Doug Tody (NRAO) and Rick White (STSCI).

The Panel congratulates the project team for the high degree of preparation for this review. The advance material was comprehensive and accessible, covering the major areas of interest to the Panel. The Panel greatly appreciates the openness and candor of the project team in discussions at the review. We would also like to thank the project team and the NRAO for their hospitality during the intensive two days of the review.

Our charge was twofold: to review the technical suitability of AIPS++ for current and future radio astronomical data processing, and to make specific recommendations on how to make AIPS++ a useful and user-friendly data-reduction package. However, any meaningful review of these issues must, of necessity, involve some examination of the sociological environment of the project.

A number of significant technical issues have dogged the AIPS++ program from the earliest days: incomplete functionality, inadequate performance and robustness, the inability to perform end-to-end data reductions and difficulty with usability and programmability. This Panel notes that there have been a number of major reviews of the AIPS++ program in the past, panels that have provided clear recommendations for change and warnings of problems that will arise if such change was not made. Several of the significant recommendations of the past have not been adopted and problems in management and technical development of the AIPS++ package have persisted. However, we also note that there have been a number of very significant improvements made within the last six months. In particular, the interactions and effectiveness of Steve Myers (Project Scientist) and Joe McMullin (Project Manager) have been particularly important in this regard.

This Panel finds that AIPS++ has the *potential* to become a package suitable for general radio astronomical data processing, for current and future instruments, provided some significant changes are made to the management and development process. If the recommended changes are adopted, there are excellent prospects for AIPS++ meeting the challenges of the next-generation facilities currently under development. We feel strongly however, that continuing on the current path will not yield this result.

The Panel makes a number of recommendations that we feel are crucial to the future success of AIPS++. The charge to the Panel and answers to questions posed by the Executive Committee are included in the appendix.

Review Process

The material for the AIPS++ Technical Review was made available to the Panel on February 21, 2003. This material, which was very comprehensive in nature was reviewed by the Panel and resulted in a number of email exchanges prior to the review.

The review itself was held at the Array Operations Center in Socorro, NM on March 5 and 6, 2003. The Panel met on the evening of March 6 to outline the report and then convened in executive session on the morning of March 7 to produce a first draft. This draft was the basis of a verbal briefing later that day to the AIPS++ Executive Committee (Harvey Butcher, Tim Cornwell, Richard Crutcher, Phillip Diamond, Ron Ekers, Fred Lo (chair) and Gianni Raffi), as well as Steve Myers (NRAO project scientist) and Joe McMullin (project manager).

An interim version of the report was submitted to the chairman of the AIPS++ Executive Committee, Dr. Fred K.Y Lo, on 17 March 2003.

The final report was submitted on April 12, 2003 to Dr. Lo, for further distribution to the rest of the AIPS++ Executive Committee.

Major Recommendations

1. Change the focus of AIPS++

Until now, the aim of the AIPS++ project has been to develop a general-purpose package for the analysis of radio astronomy data. The intent has been to provide tools to allow the analysis and processing of data from current and future radio telescopes, including the VLA, GBT, ALMA, LOFAR, ATCA and EVLA. After 10 years of development, the functionality of the system is approaching that required to analyze a subset of existing VLA science cases, but with performance, usability and some code defect issues. The system is not yet at an appropriate level of maturity to deliver to users.

We believe that the major reason for the lack of delivery has been the focus on producing a general-purpose system at the expense of producing software with specific science goals driven by actual radio astronomy analysis use cases and data. It is now necessary to change the focus of AIPS++ from a system aiming to be a general-purpose radio astronomy package to a system that is developed to meet the strategic goals of the Consortium.

In order for the Consortium to meet its analysis and data-processing software goals, the approach to developing AIPS++ must be driven explicitly by the needs of the current and planned major radio astronomy projects. This implies a modified development process consisting of

- science staff identifying use cases and developing requirements (including performance requirements) from the use cases
- development staff flowing the requirements to the software design and developing a build plan with content and delivery tied directly to the project milestones
- a test process that involves science testers using test data based on the use cases
- software acceptance by the relevant science or project lead

This approach is fundamentally different from the current AIPS++ development paradigm and implies tight coupling between a master schedule and the AIPS++ software deliveries.

ALMA is a high-priority project and will be a major driver of the future software development. In principle, it is preferable to examine high-priority use cases from all Consortium projects and derive an integrated set of requirements and build content from these. In practice, however, this diffuse focus, coupled with a small and distributed project team, is unlikely to succeed. We propose instead that the task of completing sufficient functionality to demonstrate a full set of VLA use cases be viewed as a high priority, not in order to satisfy users (who are generally using AIPS), but in order to complete a set of core functionality required for ALMA. A subset of complementary use cases of other Consortium instruments

(such as ATCA and BIMA) should also be incorporated in order to match remaining core ALMA requirements. Examples of these requirements are complex spectrometer specification and configuration, mosaicing and linear polarization.

We see the completion of the VLA use cases and a carefully selected subset of complementary use cases from other Consortium instruments as an essential part of the ALMA development.

The support for external non-Consortium users should be de-emphasized until an accepted system can be completed. We recommend taking AIPS++ off-line until sufficient functionality has been demonstrated per use cases. At present, VLA users are generally using AIPS, and the GBT users could be considered as the first target users of the system.

We believe that adopting these recommendations can, in time, lead to the AIPS++ goal of providing a general-purpose radio astronomy package, a package that will provide a general-purpose toolkit, cater for a variety of instruments and provide a good environment for algorithm development.

2. Modify the Development Process

In order to deliver software tied to the specific milestones of the Consortium projects, the present software-development approach needs to be modified. Specifically, we suggest considering incorporation of the following elements in future development:

- Develop use cases for the major project drivers
- Derive requirements from use cases (including performance)
- Develop build schedule with explicit release dates with incremental functionality tied to key project milestones
- Develop design and hold design reviews
- Develop tests, test data
- Develop code and documentation
- Peer review code
- Test using data tied to use cases
- Acceptance of release by Project Scientist

Many of these elements are currently present, but are not focused on the delivery milestones derived from the needs of the major Consortium projects.

Tying the content and deliveries to the projects may result in the need to abandon the current 6-month routine delivery cycle, which is more suited to a system in long-term maintenance than one undergoing significant development. The recent adoption of mini-deliveries each on 4-6 week timescales is a move in this direction.

A critical aspect of developing the AIPS++ build schedule is creating a master schedule tied to the needs of the Consortium projects. The resultant AIPS++ master schedule should be consistent with the major project schedules (such as GBT, ALMA, EVLA).

This approach would allow the answer to the question “What’s AIPS++ needed for” to be something like:

- Perform analysis of {BIMA, GBT, ALMA, LOFAR, etc.} data per the use cases and requirements documented in w, x, y and z
- Support pipeline processing of {VLA, ATCA, etc.} data per document x, y

The approach would have an associated set of processes that would be applied to all development projects, e.g., a process for generating a use case, by whom, how to ensure its part of full coverage, etc.

3. Strengthen the Project Management and the Project Team

The project team is clearly highly talented, energetic and dedicated. Although these are essential ingredients for success, by themselves they cannot result (and indeed have not resulted) in achieving the goals of the AIPS++ Project. In particular, it is evident that there are a number of weaknesses in the management of the project that must be addressed.

Project management must be strengthened in a number of areas. The Panel was struck by the lack of standard project management methodology and reports. Suitable formal project management techniques should be adopted, both for managing the project and communicating the state of the project to the Consortium management and partners. Key metrics such as earned value and progress against project milestones should be adopted, in order to show progress in a meaningful way and to allow for adjustment in scope and schedule as warranted.

The current practice of sharing the Project Manager role between several individuals, none of them on a full-time basis, presents a problem. For a project of this magnitude the Project Manager role must be recognized as a full-time position, one that cannot be shared with other duties and responsibilities.

Project management must foster a clear customer focus among the project team members, where the customers are the major Consortium programs requiring AIPS++. This is a distinct change from serving an amorphous general user community. The development of an AIPS++ master schedule tied to the Consortium projects will assist in this regard.

Every effort must be made to fill current vacancies. The team should be strengthened by the addition of professional software engineers; sufficient

astronomical expertise is already present in the existing project staff. One of the project staff positions should have the role of software architect, responsible for the overall integrity and consistency of the core classes. This is especially important, as significant further development will be required in the future.

More effort must be made to utilize full-time FTEs. Staff allocated at a small fraction of their time over an extended period are inefficient, and in the case of marginal involvement (<10%, of which there are several) they often amount to a drain on overall project resources.

Although this program could benefit from additional personnel, the Panel believes that the current team (utilizing all currently open positions, of which there are several) should first demonstrate their capability to use what resources they have more effectively. We also note that several Consortium partners are currently contributing significantly less staff than that required of them. This is a matter for the AIPS++ Executive Committee to resolve.

4. Further Strengthen the Project Scientist Role

The Panel applauds the appointment of a strong and active AIPS++ Project Scientist. It is clear that this has already had a significant and positive impact on the project. The Panel also wishes to acknowledge the recent activities of the NRAO AIPS++ User Group (NAUG). The interaction of the NAUG and the project in recent months has been clearly beneficial, and demonstrates the need and power of actively involving users in the development process.

The Panel believes the role of the Project Scientist is critical to the success of the project and, therefore, needs to be expanded in a number of ways.

Analogous to the Project Manager, the Project Scientist should be appointed at the Consortium level and be responsible to the AIPS++ Executive Committee. Individual Consortium sites and projects should appoint individual Site Scientists. The activities of these Site Scientists would be coordinated by the Project Scientist. This is similar to the user group structure presented by Steve Myers during the review, but formalized in the following ways.

In coordination with the Site Scientists, the Project Scientist should have the following responsibilities:

- Develop science-based use cases for all AIPS++ supported major Consortium instruments.
- Based on use cases, derive project science requirements. Note that science requirements should include baseline performance requirements.
- Identify and appoint Consortium staff scientists as use case testers. It is expected that these use case testers will be active scientists with research experience with their home instruments.
- Organize and prioritize use case tester feedback for project action.

- Verify that project deliverables fulfill use-case-derived requirements.
- Organize development of high-level end-user User's Manuals ("cookbooks") for major scientific threads.

In coordination with the Project Manager, the Project Scientist should have the following authority and responsibilities:

- Provide scientific prioritization for all project activity, based on clearly defined scientific use cases, and subject to the high-level project priorities established by the Consortium Executive Committee.
- Assure that project activity is aligned with the scientific requirements.
- Certify and approve for public release any new version of AIPS++, after the Project Manager is satisfied that a build is technically ready for release and after the satisfactory testing of the scientific use cases.

Technical decisions should continue to lie with the Project Manager. The Project Scientist will generate use-case-based requirements, but it is up to the Project Manager to decide how to achieve those requirements.

5. Short-Term Priorities

Development efforts over the next 12 months should focus on the following specific issues:

- Significant improvements in reliability and stability
- Performance improvements approaching the performance of AIPS
- Synthesis imaging capability covering many common use cases of the VLA
- Reduced emphasis on graphical user interfaces

The Panel endorses the current systematic approach to investigation of performance issues in AIPS++ presented at the review. Benchmarking performance of components and feeding back problems to the developers for resolution has been quite productive. It is important that the planned "best practices" manual cautioning developers against performance-robbing practices be produced soon. It seems likely that continuation with this program will increase performance to the point where it approaches that of AIPS.

There are defects in the design of many of the present GUIs, some related to the underlying implementation of Glish and others to usability issues. However, script-driven processing is adequate to demonstrate the core functionality of AIPS++ at this stage. A Viewer GUI is recognized to be essential for this mode of operation. Other graphical interfaces are required in the longer term, but should have a lower priority for now. Redesign of existing GUIs will be required, but should await resolution of the underlying GUI performance issues. The project intends to achieve the needed level of performance through a re-implementation of significant parts of the current technology. However, other less ambitious options are also available (e.g., based on the Qt widget library), and should be

considered if the implementation of the new technology takes longer than planned.

Future releases must include user-level “cookbooks” for processing data with validated use cases. Given the toolkit nature of AIPS++, such cookbooks are the most effective way for non-expert users to use the system successfully and productively.

In the case of the NRAO, the VLA provides an ideal test bed for AIPS++, not only because of the many common observing modes, but also for organizational and sociological reasons. The Array Operations Center co-locates many of the AIPS++ developers and Project Scientist as well as a large pool of scientific users of the VLA. The recent increase in involvement of the scientific staff is very encouraging as is the quick turnaround on bug fixes. The effort to identify the VLA scientific use cases needs to be completed along with systematic testing of these use cases. Successful achievement of these efforts opens the door for an AIPS++ VLA pipeline, which would complement the proposed archive. Routine operation of such a pipeline would give much needed credibility to AIPS++ and confidence that it could fulfill its Consortium obligations.

6. Proceed With the Proposed Technology Changes

The current AIPS++ architecture is built around Glish, which serves as the software bus for task communications, the task control system, the scripting language for high-level applications, the command-line interface (CLI) for user interaction and the GUI system. Since the AIPS++ system has its own custom approach to each of these needs, it is effectively a closed system that cannot easily benefit from the many developments in the rest of the software world. In fact, many of these functions might be better performed by standard components developed in the software community over the last 10 years.

The proposed new technology appears to be a sound basis for changing AIPS++ to an open system that uses industry standards where appropriate (e.g., CORBA, Java) and allows the use of robust open-source software in place of AIPS++ software (e.g., Python instead of Glish). By moving Glish out of the core of the system, a major bottleneck for improving the performance of GUIs is also eliminated. Using the ALMA Common Software as the basis for the bus is also appealing, since that software is already in use within the project.

The case made for a new component-based distributed architecture is strong. A preliminary analysis of implementation technologies has been performed. This analysis should continue and should be the major focus of any proof-of-concept. In particular, the Alma Common Software (ACS) appears to be well aligned in terms of architecture and technology. However, the current ACS implementation was designed for a different purpose and may not be suitable for use within a data analysis framework without significant rework and extension. Collaboration with

ESO on a common "ACS-light" core could be beneficial to all concerned, but there may be difficulties in reconciling the development and release needs of the ACS in support of ALMA with the needs of the AIPS++ project. The project should prepare for the possibility that they will be responsible for the work to integrate ACS (or other technology identified by the feasibility study) into the AIPS++ system. A common software framework is desirable, but it may prove to be impractical to share this core software with other systems.

The first step in considering the technology change should be a feasibility study that evaluates alternative technologies relative to the long-term project requirements. It is essential that this evaluation not interfere with the ongoing improvements in functionality, robustness and performance required for the AIPS++ tools and applications. Parallel efforts focusing separately on completing the current AIPS++ science data-processing capabilities and prototyping a new system framework are therefore called for. However, the staffing for the current AIPS++ applications group must not be reduced; moreover the effort of evaluating and implementing the new technology will require staff with a strong software engineering background, rather than that of astronomical data processing. These requirements point to the need for additional resources, probably carried out by a group that is independent of the existing project team.

Appendix A: Charge to the Review Panel

The AIPS++ package has been under development as a general radio astronomical data analysis package for about 10 years and has established some significant capabilities. Nevertheless, the AIPS++ package has not reached the desired final state of being the data analysis package of choice, and there are some adverse perceptions of AIPS++ in the user community. These perceptions include:

- AIPS++ runs too slowly.
- AIPS++ is hard to learn and is difficult to use.
- After 10 years of development, AIPS++ functionality still significantly lags that of older packages.
- It is difficult to program in AIPS++, leading to long development timescales and initial reliability problems.

Objectively, the changes that need to be made to the project to arrive at the final state of AIPS++ can range from minor modifications to major revisions, or even to seeking an alternate approach altogether, but any proposed solution to the current problems cannot be open-ended in time and resources. Consequently, the AIPS++ Executive Committee has agreed that a Technical Review of AIPS++ is needed as soon as possible to inform decisions.

The purpose of this AIPS++ Technical Review is thus twofold:

1. To evaluate the technical content and suitability of the AIPS++ software package as a radio astronomical data processing program for current and future instruments; and
2. To make specific recommendations, wherever possible, on how to make AIPS++ a useful and user-friendly data reduction package.

In its report, the panel is asked to estimate the consequences in schedule and resources of any substantial changes recommended. The panel's final report will be used by the AIPS++ Consortium Executive Committee to help determine the future status and course of development for the AIPS++ project. The panel is asked to present its final report by no later than 2 months following the review meeting.

AIPS++ Executive Committee
January 13, 2003

Appendix B: Answers to Specific Questions to the Panel Posed by the Executive Committee, January 13, 2003

Q1 Goals:

Are the project goals (e.g. the mission statement, development plans) clear and appropriate for the needs of the AIPS++ consortium and the international community? Are the near and long term goals well defined and aligned with the needs of the current and future customer base? Are the goals achievable on timescales acceptable to the customers? Does the AIPS++ Project have the correct vision for the future evolution of the package?

- The answer to the first two Q1 (Goals) questions is “no”. Recommendations about high level project goals and directions are discussed in Recommendation 1. Recommendations about short-term priorities are provided in Recommendation 5. In summary, the Panel believes the project focus must be changed from the goal of developing a general-purpose image processing facility for radio astronomy to developing a system that meets the strategic goals of the AIPS++ Consortium. In the short-term (next 12 months), this means focusing development activity in three key areas: (1) class library reliability and stability; (2) package performance; and (3) VLA synthesis imaging use cases. Development and/or improvement of graphical user interfaces should be given lower priority until these three main tasks are completed.
- We re-emphasize here that targeting the correct customers is the one of the biggest challenges facing the Project. Our recommendation is to focus on meeting the needs of the key Consortium projects as the first priority, and to de-emphasize providing support for external general community users until such time as system robustness and performance reaches acceptable levels for specific and well-documented use cases.
- Are these goals achievable on timescales acceptable to the customers (Q1.3)? Without following the major recommendations of this review, the Panel believes the answer is “no” (see the Introduction).
- Does the Project have the correct vision for future evolution (Q1.4)? As discussed in Recommendation 6 (and below), the answer is a qualified “yes” based on the concepts presented. A real feasibility study and implementation plan is needed before this question can be fully answered.

Q2 Design:

Is the design of AIPS++ sound, such that it will fulfill the requirements of the projects of the consortium partners over the projected timescale of the current decade?

- The design of the AIPS++ package is based on Object Oriented principles and makes use of an architecture comprising distributed objects on a “software bus”. Such an approach allows both the proper encapsulation of specialist algorithms

and functionality, as well as the ability to upgrade selected parts of the package (e.g. graphics capability) in the face of changing technologies. This is a widespread and successful approach used in modern software engineering. The Panel considers this to be a sound basis for the software design. However, in order to fulfill the requirements of the Consortium partners over the current decade, a number of important technical and management changes must be made. Without these changes, the design will not be able to meet the expected needs. The specific recommendations for change are articulated elsewhere in this report.

Q3 Architecture:

Is the underlying software architecture sufficiently flexible and maintainable? Does it allow easy upgrade paths, such as parallelization, use over the Internet, or use of grid computing? Will AIPS++ be flexible enough to handle heretofore-unforeseen data processing problems, such as those posed by consortium projects such as ALMA, EVLA, LOFAR, CARMA, and GBT? Will the AIPS++ component parts be inter-operable over the immediate lifetime of the project? If there are architectural or design deficiencies, what level of rework is required to correct them?

- While the basic distributed, component-based architecture of the current AIPS++ system is sound, the implementation is deficient primarily due to the dependence upon Glish for all aspects of the system framework, and the strong coupling between subsystems which results. The major system components (software bus, distributed object interface, scripting language, Table system, etc.) are all custom, with private interfaces between subsystems. The scalability of the system is limited, mainly due to the limitations of the Glish software bus and distributed object implementation, and to the uniprocessor oriented design of the Table system. The flexibility and scalability of the system and the ability to upgrade subsystems as new technology becomes available would be greatly enhanced if the system had a more open implementation based on open standards.
- Given the current dependence of the AIPS++ system upon Glish it is hard to see any way to address these problems without major changes to the current system. The new system framework proposed by the Project appears basically sound and has the potential to address these problems. While much can be gained by the use of modern technology based upon open standards, the challenge will be to integrate such a range of technologies and languages without producing something that is unwieldy and overly complex.
- Handling heretofore-unforeseen data processing problems from future instruments will require that the system be both flexible and scalable. Flexibility is required particularly in the area of data processing algorithms, since the real data processing requirements of future instruments are not yet understood. This can be achieved by an open system, which allows components implementing varied processing algorithms or approaches to be incorporated. Scalable computation is addressed by the proposed new system framework. Basic scalable I/O can be provided by existing cluster computing technology such as parallel file systems.

- Changes to the AIPS++ Tables interface may be required to provide scalability to future very large (multi-Gigabyte) datasets or to enable cluster computing. High-performance computing technology such as MPI-IO and HDF5 may provide part of the solution. Another approach might be to preserve the current Tables API largely intact, but reimplement this as a shared-access component to provide scalable concurrent access to large Table datasets for cluster computing. The elements of a large dataset could be stored as separate files in a parallel file system, using a conventional database such as MySQL or PostgreSQL to manage shared access to Table metadata as well as provide efficient multiprocess, indexed I/O for large tables. Open standards such as FITS and XML could be used to format individual file elements.

Q4 Components:

Are the core components of AIPS++ (e.g. Glish, Table system, MeasurementSet, MeasurementEquation, User Interface) suitable for the current and future requirements of the package? Is the core calibration and imaging software design sufficient to handle the increased demands, e.g. dynamic range, of the new instruments? If the components are unsuitable, what changes are recommended?

It is convenient to consider the core components in two groups: specialized (those providing specialist scientific or astronomical function such as the Table system or the Measurement Equation), and general (such as the scripting system or graphical user interface).

1. Specialized components: With the exceptions noted below, these components do seem to fulfill the requirements of the package.
 - Table system: The logical model for data organization provided by the Table system appears sound. However, there are a number of issues concerning performance and scalability for future instruments.

The Project is already addressing questions about performance of the Table system and its use in applications. The Table system is built on a “storage manager” layer, which provides the I/O functionality. Performance problems may lie in the storage manager itself, or more likely in the usage of the Table system by applications. A detailed end-to-end analysis of selected Table-based applications is needed, to determine how I/O is performed. This includes the order in which processing steps are carried out by the application, the use of buffering within the application and Table system and the interaction with system virtual memory, and the use of tiling within the storage manager to optimize I/O. This analysis should result in a set of design patterns for I/O intensive applications (specifying the order in which operations are performed, the optimum tiling and buffering strategies, etc.) that application programmers can follow to ensure that reasonable performance is achieved. Benchmarking and tuning of the resultant applications should then be

performed to confirm that adequate performance has been achieved. Where appropriate, the system software should automatically adapt to the environment, to make the best use of resources such as the total physical memory available.

Scalability for future instruments, for example to support extremely large datasets or to enable cluster computing, is likely to be an issue in the long run but is not as high a priority as making the current system and applications perform adequately. This issue is addressed further in the discussion of system architecture elsewhere in this report.

- **Measurement Set (MS):** This is used to hold telescope data and so plays a central role in the package. To cater for the variety of Consortium instruments, it is necessarily complex. There is evidence that the complexity has led to difficulties for the applications programmer, particularly in providing “fillers”, which are procedures to write native telescope data into the Measurement Set. To fulfill the potential of AIPS++ to cater for a wide variety of instruments with possibly evolving functionality and data outputs, the process of producing filler software must be made as straightforward and routine as possible. This can be accomplished by ensuring that the data model and interfaces are comprehensive and stable with changes made only infrequently, by providing good documentation, and by provision of template or sample fillers which can be taken as the starting point for adding a filler for a new instrument.
- **Measurement Equation (ME):** The approach of using a common MS/ME and data model to provide uniform processing for a range of instruments appears sound. Basic processing for a range of existing instruments has been demonstrated. Basic correctness has been shown by spot-checking of the results of processing artificial data or by comparison with the outputs of other existing packages. Whether or not a single approach can work for all future instruments is less clear; the approach could become unwieldy as a wider range of instruments with complex and demanding processing requirements are added. Ultimately it may be necessary to be able to support several different approaches within the same framework.

2. General components:

- **Glish:** The Panel recognizes the shortcomings of Glish as the basis for both the software bus and the GUI toolkit, although it seems to provide an effective scripting language that is popular with users. The difficulty of long-term support for Glish was also highlighted, especially given the availability of widely used alternatives. In the long term Glish should be replaced as the system framework for AIPS++, as discussed in the answer to the previous question about the architecture. In the immediate future, however, scripting using Glish should be continued, as we believe it is adequate both to

demonstrate and further develop the core AIPS++ science data processing functionality.

- **User Interface:** The toolkit capability of AIPS++ is one of the most powerful aspects of the package and should be retained. However the toolkit should not be the primary user interface. Users should be able to reduce data end-to-end for common use cases using a high level command or GUI interface, without having to resort to toolkit level programming. This could be accomplished by layering application scripts or GUIs on top of the existing toolkit components, providing both an easy-to-use interface for routine usage as well as an example of scripting for advanced users. The requirements for an interactive command language are somewhat different than those for a scripting language (e.g. command line recall), and it may prove worthwhile to provide an advanced CLI as well as enhanced GUI capabilities in the user interface.

Q5 Usability:

There are serious concerns about the usability of the AIPS++ package, including performance, ease of use, and programmability. What specific actions should be taken to verify and quantify these concerns and to remedy the real problems?

The committee found that the development team and the scientific users and testers provided verification and quantification of many of the usability issues. Our interpretation of the important areas and recommended remedies follows.

- **Performance:**
 - The current AIPS++ performance is inadequate and must be addressed. There are serious performance problems in several areas, with reported degradation in some areas by factors of as much as twenty compared to other packages. In some cases this degradation is due to such simple factors as not utilizing enough memory and can readily be rectified. The generality of the approach taken by AIPS++ appears to cost little in performance (less than a few tens of percent) and is the correct approach to adopt.
 - The systematic analysis (by benchmarking) and mitigation of performance issues that was presented at the review is the correct approach. The production of the proposed “best practices” manual to ensure that performance is not degraded by incorrect implementation practices should be done soon.
 - It must be emphasized that performance is a critical issue that will be viewed by users as a show-stopper. Efforts to enhance performance should not diminish until AIPS++ is within 10% to 20% of the performance of AIPS.
- **Ease of use:** There are three issues here: defects, documentation, and GUI’s.
 - Defects have been a significant deterrent to the adoption of AIPS++ and more emphasis must be placed on their elimination. The recently established loop of developers and testers in the AOC is an excellent way to quickly eliminate critical defects.

- End user documentation is critical for user satisfaction and productivity. The use case driven “cookbook” approach taken at the AOC is excellent and should be continued as functionality is expanded. Given the toolkit nature of AIPS++, such cookbooks are the most effective way for non-expert users to use the system successfully and productively.
- It is clear that many of the AIPS++ GUI’s are unsatisfactory. Large GUI’s suffer from sluggish performance caused by the overhead of large numbers of Glish events and the “auto-GUIs” appear ineffective. The long term solution for the Glish event problem is the proposed technology change. This new framework would also support a new GUI environment, making work in the present environment of doubtful long term value. Driving AIPS++ with Glish as a scripting language should continue to provide a powerful scientific platform for the motivated astronomer. Good GUI’s are not easy to design and are expensive to implement. Given the cost, benefit, and long term plan, the use and development of GUI’s should be de-emphasized in the short term. If the technology change schedule is such that GUI development work must be done before the introduction of the new framework, it may be useful to consider alternatives such as *Qt*.
- Programmability:
 - There were reports of programming at the C++ level being difficult. Some of these are rooted in the fact that object oriented programming does not come naturally to those with a procedural background; programmers with an object oriented background find AIPS++ quite accessible.
 - Provision of better programmer level documentation, as well as design patterns for effective use of complex components (e.g. for I/O intensive applications involving the Table system) is needed. A greater reliance on widely used components (as opposed to custom AIPS++ system components) and the proposed change to a more open architecture will also go a long way to simplifying the programming environment.
 - The model of migrating contributions at the scripting level from scientific users to code by AIPS++ programmers is valid and should be encouraged.
- Finally, stability of the core classes and their interfaces is the key to promoting understanding of, and confidence in, the system among users and developers and to increase their productivity in using the AIPS++ package.

Q6 Process:

Are the project management structure and development methodology suitable for the development, construction, and maintenance of a software package the size and scope of AIPS++? Are there sufficient tools, processes and management in place for quality assurance of the package deliverables? Is the management and process structure appropriate for the distributed nature of the project? Does the development process identify and adequately incorporate the input of its current and future customer base?

- The AIPS++ team has developed a sound code management process that supports the distributed development of the system. We do not see a need for substantial changes to this process. However, the software test and release process needs more attention, as evidenced by the amount of code in “trial”. Code reviews, testing, and programmer documentation need to be given more emphasis, to reduce the introduction of new defects.
- There is, however, a need to strengthen the project management processes as discussed in Recommendation 3, including:
 - ensuring a dedicated, full-time Project Manager;
 - use of more formal project management techniques,
 - use of full-time FTEs rather than fractional FTEs,
 - changing to a more customer-oriented focus.
- Recommendation 2 highlights the need to change to a use-case driven, master schedule based build-and-deliver cycle. Finally, Recommendation 4 endorses the recently re-introduced role of Project Scientist and discusses how this role must be strengthened, especially in the area of gathering customer input via use-case development.