NOTE 209 – AIPS++ OPERATIONS

Tim Cornwell and Gustaaf van Moorsel

18 July 1997//Revised 18 November 1997

Contents

1 Purpose

The purpose of this document is to lay out a vision for operating AIPS++ in the long-term once the first release is made. This document should serve as a starting point for discussion in a number of different forums: the AIPS++ Project, the AIPS++ Executive Committee, and the AIPS++ Scientific and Technical Advisory Group.

2 Introduction

In the latest development plan for AIPS++ (Note 202), a goal is that it be a functional equivalent to the existing consortium packages by late 1999. At that point, AIPS++ will be a fully functional system, in use for production at a large number of astronomical (and non-astronomical) sites. To get some idea of the scale of the distribution that is plausible, we can sum the existing sites for consortium packages: AIPS, Unipops, SDE, Miriad, Newstar. This is in excess of 200 active sites. Presumably the majority of the active users will be astronomers doing science, but we should not forget that if AIPS++ is successful, then some fraction of these sites will host active AIPS++ programmers. AIPS++ will be the mechanism for observatories to provide support for astronomers using their telescopes, much in the same way that IRAF is used as a vehicle for data reduction packages for a number of organizationally distinct telescopes. This is different from the existing AIPS model, where one observatory, NRAO, has control. A close, but much smaller scale, model inside the consortium is the use of MIRIAD by BIMA and the ATNF. It is thus important that AIPS++ be clear from the beginning what type of role it will play in astronomy. We see it mainly as a platform or vehicle for a number of clients to use in their work.

Let us turn, therefore, to focus on the clients that AIPS++ will have in this operational phase. These clients split naturally into a number of categories.

Observatories Observatories will use AIPS++ as the vehicle for providing support for particular telescopes. This will presumably have a number of components:

- **Software** for observing preparation, monitoring of observations, and reduction of data.
- **Distribution** of ancillary data for the telescope *e.g.* source lists, antenna coordinates, instrument history, physical parameters (such as ionospheric menasurements), *etc.*
- **Support** for astronomers using the telescope.

Astronomers Astronomers will use AIPS++ for access to a common telescope reduction package, and for analysis tools that are not the specific responsibility of any one observatory. One can also envisage that AIPS++ will become a mechanism for collaboration and for publication, but we pay less attention to these possibilities here.

Programmers Programmers (who may be active astronomers) will use AIPS++ as a resource for tools to solve problems and as a distribution mechanism for their products.

The overall goal of the AIPS++ package must be to aid science. As a working assumption, we will take it that all of these classes of clients are equally important in furthering that goal, and thus are equally deserving of support by AIPS++. We should thus aim to be as responsive as possible to all of these clients. Perhaps the most difficult problem of operations will be to find suitable compromises between the imperatives of all these different clients. Finding the right balance between conservatism and innovation will require a firm hand. We return to this problem below.

In the rest of this document, we describe the operations of AIPS++ in more detail. We see the operations of AIPS++ as splitting into a number of components. First, there are *core operational functions* that must be performed in order for the package to be installable and usable at any consortium site. Second, there is *astronomer and programmer support*, needed to support the use of the system by astronomers and programmers alike. Third, there must be a *quality assurance program* that ensures and tracks the overall quality of the system. Overseeing, coordinating and monitoring all of these functions is a *management and oversight* component. These are discussed in turn in the following sections.

3 Core operational functions

The core operational functions for any package are *maintenance, distribution*, and *development*.

3.1 Maintenance

Maintenance of the package is needed to discover and eradicate bugs, and to track environmental changes such as operating systems changes and compiler changes.

Bugs In any system, a continuing effort is needed to uncover, track, and eradicate bugs. For the mechanics of bug tracking, we currently use the GNATS package from the Free Software Foundation. This seems to be adequate currently but we may have to consider migration to another package if the current approach does not scale up. The current system has two potential bottlenecks. First, all bug reports currently go through the Project Center, and second, the bug reports are then classified by one person. We have to work to eliminate both these bottlenecks once the flow of bug reports diversifies and increases. A quick response to bug reports is vital to avoid the "stale bugs" problem whereby a delay in responding to bug reports means that many bugs have been fixed or are no longer relevant.

Tracking environmental changes The environment under which AIPS++ executes will inevitably change over time as the underlying operating systems evolve and as improved version of tools such as compilers become available. An extreme example of an environmental change that must be monitored is the advent of Java and all it portends for computing in the future.

3.2 Distribution

System In the current setup, the core AIPS++ system is maintained at the Project Center in Socorro, New Mexico.

System distribution for developers occurs via two mechanisms. First, periodically the system is distributed to sites over the Internet, either incrementally or cumulatively. This is usually done in a batch job, and uses ftp for transmission. Second, changes to the master are performed via an NFS mount of the master disks onto a local machine. The first function, downloading, is highly reliable and essentially presents no significant problems. The second function, uploading, is more problematical since the Internet is sufficiently congested that the necessary NFS mount is hard to acquire and maintain. This will be alleviated with the forthcoming modifications to NFS to use TCP/IP in place of simple UDP datagrams. Inside NRAO, we rely upon our own intranet in place of the Internet. Outside NRAO, we have the option of bypassing the mounting process by ftp'ing files directly to an NRAO computer.

For end-users, the system is currently distributed as both binaries or source by anonymous ftp. Our beta testing has shown this to be a simple, effective and reliable approach. About 90% of AIPS distribution is via ftp. However to service some communities, we will also distribute via two hardcopy forms: CD-ROM and various tapes: 8mm and DAT initially.

Data Data in AIPS++ fall into a number of categories. First, there are data from telescopes. Second, there are data needed for the software to execute. Examples would be databases needed for the Measures system. Third, there are databases that aid the user in some process. Examples would be source parameters, antenna locations, scheduling information, *etc.* Fourth, there are data needed for testing and demonstration purposes. We are currently determining requirements for the latter three, and plan to put in place mechanisms for users to acquire such data either from the AIPS++ Project Center, or directly from the original source.

Knowledge Knowledge about astronomical processing and analysis should be one of the prime commodities that AIPS++ can help provide to its clients. We should distinguish between knowledge generated by AIPS++, and knowledge funneled through AIPS++ mechanisms. In the former category we see conventional documentation such as reference manuals, cookbooks, tutorials and migration guides (*"AIPS++ for AIPS users"*). In the latter category, we think that AIPS++ should provide mechanisms for users to talk to other users, via exploders, listservers, news-letters, *etc.*. The major value added by AIPS++ in this latter category would be a common framework in which the discussions can be couched, and an archive which can be searched (a database for knowledge). Thus, for example, the NRAO Synthesis Imaging workshops should be strongly coupled with AIPS++ documentation.

Schedule The update schedule will be one of the most contentious aspects of operating AIPS++. We can imagine that a dynamic group in a phase of rapid development of some sub-system in AIPS++ will push for a rapid release schedule whereas more established groups with less rapidly evolving needs will prefer a conservative approach to releases. For help in understanding this issue, it is instructive to consider the evolution of AIPS during the development of software for the VLBA. During this time, the AIPS software was evolving rapidly in response to a deluge of new data (and concommitant problems to be solved) from the VLBA correlator. Groups tightly tied to use of the VLBA therefore wished to update their own working copy of AIPS more often than allowed by any reasonable formal release schedule. The answer was for them to be placed on the AIPS Midnight Job whereby the working version is updated every night. The advantage of this is obvious. The disadvantages are that the system is then more unstable, and, more seriously, that it is impossible to determine the state of the system on any given date. However, if an actual release is also available then these bad effects can be reduced. This may form a good model: the center should release stable, well-tested releases a couple of times per year, and then allow users to be placed on a periodic update schedule as dictated by their needs.

3.3 Development

From experience with other systems, it is certain that development of AIPS++ will continue until (or even after) it is supplanted by another system. Historically, development of an existing production system has been an extremely delicate proposition. To add new capabilities without compromising those existing capabilities was difficult, and the degree of success that AIPS++ has in the area will be the key measure of the success of the object-oriented design of AIPS++. It is apparent from our (and industry) experience that object-oriented methods work well if high-level interfaces do not have to drift over time.

Core library Any significant continuing development of the core library after the initial programmer's release will need careful consideration. Additions to a widely-used interface are often acceptable but the cost of changing an interface can be very large. This means that the core at the time of the first developers release, the core libraries should be up-to-date and state-of-the-art. In practice, if a core library *is* changed, we think that the resulting cost of changing code has to be borne by the Project Center since otherwise the cost to end-user programmers of tracking continuing changes will be a strong disincentive to use the system.

Applications Development of applications will continue throughout the lifetime of the system, and is clearly to be encouraged! Here the most difficult problem is deciding which pieces of an application can be moved to the core library. Advice on priorities for this process can come from our clients but detailed advice on what to migrate when must come from a technically knowledgable group. Integration of a new package will be viewed most favorably if the code obeys our coding and documentation rules, and if the expected lifetime of the package is more than one year.

System Changes in the implementation of the system library will be necessary over time for a large number of reasons: operating systems change, compilers improve, new devices come along (*e.g.* better long-term storage devices, improved visualization devices, *etc.*). Determining which configurations will be supported is a function of the Project Center, but advice is particularly important here.

Development entails hard choices. These choices have to be made bearing in mind the needs of potentially all the clients, and making sensible guesses as to which lines of development should be followed with which priority. As we said above, this will probably be one of the most contentious areas in operating AIPS++. We think that a diverse but tightly knit group would be best suited to advising AIPS++ management on such issues. We return to this issue below.

4 Astronomer and Programmer support

4.1 Help

We see help for clients of AIPS++ as being layered. We strongly suggest that each AIPS++ consortium site have a designated contact person who can also provide immediate, in-person advice to both astronomers and programmers on how to use the system. For the next layer of help or for queries from non-consortium sites, there are a couple of possibilities: either set up a special group to provide detailed help, or adopt something similar to the *Designated AIP* program of the AIPS system whereby such duties are rotated around the active programmers in the Project, perhaps to three at one time, one per 8 hour shift? The latter has a number of attractions: it keeps the programmers in touch with the users, the level of advice is high, and it spreads a difficult and demanding job amongst a group of people. The downside is that it takes some fraction of the time of the best people. Overall, we prefer the latter approach and recommend that it be adopted.

4.2 Education and training

Users workshops For the first few years of operation of AIPS++, it will be important to hold training sessions for astronomical users. This is probably best considered as a responsibility of the consortium partners so that, for example, NRAO would offer a workshop for the users of NRAO telescopes. The role of the AIPS++ Project might be to send core staff to help with those workshops.

Tutorials and demonstrations Computer-based tutorials and demonstrations are an effective way to train end-users both in concepts of astronomical observing and in AIPS++ data reduction. Computer-based conferencing may also eventually allow effective training to proceed year round, but this needs some investigation.

Developers workshops Programmers need training at a number of levels: programming in Glish, in C++ using our libraries, adding to our libraries, and finally sub-system development. We envisage that all of these would be covered in annual developers workshops.

4.3 Feedback

We see user feedback as coming via a number of mechanisms. Immediate feedback comes via email, phone calls, and bug reports. This feedback and any related replies should be logged in a generally accessible form. Thus for purposes of logging, we will prefer feedback via e-mail. Longer term feedback comes via User Group meetings at which representative clients of AIPS++ can comment on current capabilities and future development priorities.

4.4 Library and Contributed code

The code in AIPS++ is organized into a number of conceptually different types of repositories. Admittance of code to these repositories is controlled by a number of different policies. The classes of repositories are:

core libraries *aips, synthesis, dish, vlbi, doc* These contain the core library code and documentation of AIPS++. Admittance of code and documentation to one of these directories requires passing our code review process. The AIPS++ Project is responsible for maintaining code in these libraries.

documentation library *doc* The core documentation source is kept here. We currently have no policy on admittance to this area. This is a topic that we will have to review, and possibly move to package level documentation *package/doc.*

consortium libraries *atnf, bima, nfra, nrao* These are repositories for consortium site specific code. AIPS++ has no policy on whether these are subject to code-review, but we do demand that coding, documentation and other standards be obeyed. The consortium sites are each responsible for maintenance of this code.

contributed library code *contrib* This is a repository for contributed code that is either unsupported or supported by an individual author. A minimal set of code standards probably needs to be enforced *e.g.* some programmer documentation in our standard format, and some user documentation (if appropriate), optionally in our standard format. We currently have no code in this category.

ad hoc library code *trial* This is a way-station repository for code that is in development by consortium programmers and is not yet ready for admittance to a core library. There is no admittance limitation, and the individual programmers are responsible for maintenance.

It is in the interests of AIPS++ that good quality, useful code migrate from the peripheral libraries such as *contrib* and *atnf, bima, nfra, nrao* into the core libraries *aips, synthesis, dish, vlbi, doc.*

5 Quality Assurance

Traditionally quality assurance for software packages has been either diffused over an entire organization or, more commonly, done by the users. In keeping with the large scope of the AIPS++ Project, we have decided to designate a Quality Assurance Group that has special responsibility for ensuring the quality of official AIPS++ products. The main task for the QAG is assuring the high quality of AIPS++ products, principally via testing, including execution of software, reviewing code and documentation.

5.1 Testing

Unit testing Currently the code review process requires that along with submitted code, unit-test software is also submitted. The complete suite of unit tests is run once a week as a check of the basic integrity of the system. We have found this to be extremely useful, and see no reason not to continue it.

Application testing Application testing is currently weak. Some test scripts are present in the system but these are few in number. We think that two orthogonal approaches are needed. First we must perform the equivalent of periodic unit testing checking known inputs against expected outputs. Second, we should assign a group to perform blind-testing of applications, to trap unexpected behavior not caught by unit-testing.

Package testing Does a given package do all that is required? Is the model used an appropriate one? Are there better models available? This type of high-level review will require a collaboration between the QA group and an expert in the field. One might think of this as being similar to a review of a technical textbook.

Benchmarking The AIPS Project found that benchmarking of new systems using a pre-determined set of applications was extremely useful both for determining possible computer purchases and for uncovering systematic performance problems. They use a set of Dirty Dozen Tasks to characterize the performance.

The Quality Assurance Group will be initially consist of three people, with the following division of responsibility:

Code Cop
- Ensuring that code is reviewed in a timely manner
- Ensuring that module and code documentation is kept up to date
- Ensuring that unit testing is performed
- Updating external libraries when necessary
- Maintaining the RCS and Change log entries so that they are useful
- Maintaining the directory hierarchy

Rules Boss
- Establishing new coding standards and removing outdated ones
- Maintaining templates used for aips++ development
- Controlling development of the AIPS++ directory hierarchy
- Maintaining and policing the file extensions used
- Controlling inclusion of external libraries

Chief Tester
- Responsible for user level documentation

- Ensuring that that applications testing, both automated and interactive, is performed
- Maintaining benchmarks and performing new ones
- Has final sign-off on AIPS++ releases

The head of the QAG will report directly to the Project Manager.

5.2 Code Reviews

Our code review process has been in place for some time now and seems to work well for the current size of the Project. The process is documented in Note 167. A *code cop* is currently responsible for overseeing the code review process. As the Project grows, this role would probably better be served by a group of people, perhaps in the above-mentioned Quality Assurance Group.

We need to augment the code review process with design reviews to catch problems early on in the design-implement-review-use cycle. We may also wish to use code walkthroughs or inspections.

6 Management and Oversight

It is our view that the current management structure works well, and strikes a reasonable balance between autonomy for and oversight of the Project Manager. Thus any change to the structure must be motivated by the transition to operations. Lines of command and responsibility must remain clear, both inside the consortium sites and within the Project. Thus the reporting line of *Executive Committee - Project Manager - Site Managers - Project staff* should be maintained if at all possible. We note that the lack of a formal line of responsibility of Site Managers to Project Manager works satisfactorily in most cases. It will be a challenge to maintain this aspect under the increased pressure of operations.

Although we hesitate to suggest an overly structured management scheme, we do think that some formal recognition of the de facto sub-structuring of the Project is probably overdue. We have group leaders in various areas: System, Core library, Synthesis, Single Dish, Visualization, *etc.* Currently most activities in these groups are monitored at a detailed level by the Project Manager. In the long-term, this is not feasible and we need to move to a situation where the sub-group leaders take more explicit responsibility for setting, assigning, and monitoring targets. A common philosophy and framework for management within the sub-groups can be imposed by the Project Manager.

In addition, we need to add explicitly an Operations Manager who oversees the day-to-day operations, freeing the Project Manager to concentrate more on longer-term issues. This Operations Manager would report to the Project Manager.

Advisory and oversight roles should be added to this proposed structure. From the discussion in the previous sections, we believe that two advisory groups are required:

AIPS++ User Group This group of representative users of AIPS++ meets annually (or perhaps bi-annually initially) to advise the AIPS++ Project management. It should be a large group with representatives of all three classes of clients: Observatories, Astronomers and Programmers. Moderate and long term issues are discussed in this group.

AIPS++ Technical Working Group This is an internal group that meets often, perhaps monthly or quarterly, to advise the Project Management on detailed technical issues. It is here that short-term development issues are discussed. We recommend that this be composed of no more than a dozen people drawn from the inner circles of the Project and thus limited to those with deep knowledge of working inside the system.

These two advisory groups report directly to the Project Manager. Note that these roles are currently fulfilled by the Scientific and Technical Advisory Group that reports directly to the Project Manager. We think the STAG should evolve into these two groups over a few years.

To provide oversight on overall project management issues, we suggest that the Executive Committee may wish to appoint another group that reports directly to it on management of the project. We suggest that this group be small and be composed of more senior people with direct experience in astronomical software and astronomical software management.

7 Resources and Timescales

The current AIPS++ staff is quite large: in aggregate, we have 25 people contributing just over 18 FTEs to the AIPS++ Project. Of these, 11 are employed by NRAO, and contribute 9.1 FTEs. The numbers for the other partners are: ATNF 4 and 2.45, BIMA/NCSA 5 and 3.25, NFRA 5 and 3.25.

We think that some of these staff should participate in the general shift of the Project from development to operations, but we hesitate to prescribe how this should be done. The new resources needed to implement this plan are:

- Operations Manager: providing the Operations Manager is probably best seen as an NRAO responsibility that is part of hosting the Project Center. Such a person should be appointed as soon as possible. The initial prime responsibility of the OM will be to implement this operations plan. The Operations Manager will also handle the mechanics of distributing AIPS++ via the mechanisms described above.
- Documentation group, headed by an astronomer, to oversee the production and maintenance of high quality documentation, including demonstrations and tutorials.
- Quality Assurance Group: could be located at any location. Location at the Project Center would have some minor advantages. Establishment of a QAG could proceed over the next year or two.
- Support people at each site: this requires a migration towards a different type of expertise, more astronomically based than currently true. We recommend that the consortium sites review their staffing in light of this need.

8 Summary

The development of AIPS++ is ambitious. We have chosen to be similarly ambitious in our plans for operation. An estimate of the staffing required is about 6-10 FTEs to operate as we have described. *This is in addition to the staff needed for any ongoing development of AIPS++, which we anticipate to be substantial.* Many of these people can be drawn from existing operational staff at the various observatories. A bump in staffing around the time of the transition into operations is probably inevitable as we finish development of the core systems, and start the preparations for operations considered here. We anticipate that the development staff will drop in number as the core solidifies. Many of these staff can be moved onto either Operations or to development for specialized telescopes such as the NRAO MMA or the SKAI.

The Quality Assurance Group will be set up immediately (November 97). The initial assignments will be Code Cop: Ralph Marson, Rules Boss: Wim

Brouw, Chief Tester: Jan Noordam. Ralph Marson will direct activities of the QAG, reporting directly to the Project Manager.