# E2E PROJECT BOOK

**VERSION: 1.1 (DRAFT)**

**JULY 12, 2002**

[ED. TIM CORNWELL]

http://www.nrao.edu/e2e/documents/e2eprojectbook_11.doc

## REVISION CHART

| Version | Primary Author(s) | Description of Version | Date Completed |
|---|---|---|---|
| Draft | Tim Cornwell | Initial draft created for distribution and review comments: complete when scientific requirements have initial text | TBD |
| 1.0 | Tim Cornwell | First version including description of core areas, and outline of Cycle 1. Note that many areas are sketchily filled out. | 2001-11-27 |
| 1.1 | John Benson, Tim Cornwell, Boyd Waters, Honglin Ye | Update for the end of development cycle 1 | 2002-07-???? |

## CONTENTS

## LIST OF FIGURES AND TABLES

# 1. INTRODUCTION

***Author: Tim Cornwell***

***Revision date: 2001/11/19***

***Status: good version***

## 1.1 Project Overview

The objective of the e2e project is to provide integrated end-to-end processing for observations made on NRAO telescopes. The ends are proposal submission (start) and scientific analysis (finish). The software to be delivered must handle:

- Proposal submission
- Proposal management
- Observation planning and preparation
- Real-time observing
- Scheduling of sets of observations
- Archiving of observations
- Pipeline processing including calibration and imaging

These are to be treated as a whole, so that information flows through the above steps without loss due to *e.g.* format changes, omissions, *etc.* For this to be possible, some changes in current operations may be necessary. In addition, some existing tools will be deprecated in favor of more uniform tools.

Our plans laid out in this document are for 2005, with re-evaluation and re-scoping based upon experience up to that time.

An initial **straw man** set of deliverables is:

- Operational Model
- Proposal Submission Toolkit
- Proposal Management Toolkit
- Observation Scripting Toolkit
- Telescope Simulation Toolkit
- Observation Evaluation Toolkit
- Remote Observing Toolkit

- Observation Scheduling Toolkit
- Archive Toolkit
- Pipeline Toolkit
- Calibration Source Toolkit

A more detailed description of each of these deliverables is given below, each in a separate chapter. Note that this breakdown is given mainly to focus discussion, and it may be revised as a result of subsequent work. Each deliverable must be delivered in some form, but the content of each deliverable is on a prioritized, best-effort basis. Scientific requirements distinguish between essential, highly desirable, and desirable features. It is intended that at a minimum all essential functions be delivered.

The resources required for the e2e project are a project manager, a project architect, a project scientist, scientific programmers, and scientists. Computers and other software development tools are required. Additional hardware access is needed for prototyping archives and pipelines.

Reuse of existing packages or concepts is strongly preferred if possible. The "front end" toolkits (PST, OET, OST, RTOT, ST) may be outsourced or developed from another project such as VLBA OMS or Gemini. Some parts, in particular the OET and the PT, may be implemented as a native facility in AIPS++.

The e2e project is a service to other NRAO projects and telescopes: ALMA, EVLA, GBT, and VLBA. The project formally started 1 July 2001, and is expected to run until deployment of EVLA and ALMA, but with much of the development completed by 2006. A strong timeline driver from the EVLA is the plan to have the Control and Monitor system deployed by 2005.

## 1.2 Motivations

The motivations for the e2e package are to:

- Streamline the use of NRAO telescopes by scientists.
- Archive scientific results from NRAO telescopes for general access by scientists.
- Aid operation of NRAO telescopes by NRAO staff.

## 1.3 Exemplars

In developing the e2e package, we intend to draw upon the extensive previous work by NRAO and other organizations. Specifically, we have looked at:

Overall architectures:

- VLBA Observation Management System
- Existing VLA proposal and archive processing systems.
- ESO data flow system http://www.eso.org/projects/dfs/

Frontend:

- GBT Observe http://www.gb.nrao.edu/~rfisher/Glish/gbt_gui_proto.html
- Gemini Proposal Tool http://www.gemini.edu/sciops/P1help/p1Index.html
- Gemini Observing Tool http://www.gemini.edu/sciops/OThelp/otIndex.html
- HST RPS2 Remote Proposal Submission
- NGST Scientist's Expert Assistant http://aaaprod.gsfc.nasa.gov/SEA/
- NOAO proposal web page http://www.noao.edu/noaoprop/noaoprop.html
- SIRTF proposal kit http://sirtf.caltech.edu/SciUser/C_PropKit/SSC_C.html
- STScI proposal submission http://www.stsci.edu/observing/proposal_process.html

Pipelines:

- ACSIS pipeline http://www.drao.nrc.ca/science/jcmt_correlator/
- OPUS pipeline http://www.stsci.edu/software/OPUS/
- JCMT pipeline http://www.jach.hawaii.edu/JACpublic/UKIRT/software/oracdr/

Archiving:

- MAST archive at STScI http://archive.stsci.edu/
- StarView database browser http://starview.stsci.edu/html/

## 1.4 Assets

An assay of relevant NRAO assets is available in e2e memos 2 and 3 at http://www.nrao.edu/e2e/memos/.

## 1.5 Assumptions

The following assumptions have been made in planning the e2e project:

1. All NRAO telescopes (EVLA, GBT, VLBA) will be served with the same tools, with deviations as scientifically necessary.
2. The current operation of all NRAO telescopes will change in accordance with the e2e common operational model, again with deviations as scientifically necessary.

3. The operation of all NRAO telescopes will follow a mostly identical process, as far as e2e is concerned, with scientifically necessary deviations.
4. The AIPS++ package will form the basis for data reduction. Some changes in AIPS++ core functionality may be requested.
5. Glish will be used as the primary scripting language.
6. Pipeline processing to a calibrated dataset and reference image is feasible for a large fraction of scientific observations with NRAO telescopes.
7. Observers will always want access to the original, unprocessed data.

## 1.6 Process

The software development process to be followed is iterative:

1. The Data Management Scientific Working Group will draw up scientific requirements, with attached priorities. These will be sent for review to various bodies.
2. The e2e project architect and other e2e staff will perform analysis of the requirements.
3. A set of goals for a 9 month development cycle will be developed and followed.
4. After each development cycle, the process will be iterated.

This project book will be filled out as this process progresses. Initially, some areas will be poorly described, but we expect that to improve as the cycle's progress. We plan to use this iterative approach because the scientific requirements must evolve in response to the developments. Similarly, our ability to plan a complete 5 year development is limited by lack of prior experience in this area. We do expect, however, that subsequent cycles will be more tightly focused.

As stated above, our plans are targeted for the year 2006. We do not believe that it is prudent to plan for 2009 (the date of deployment of the EVLA), but prefer instead to evaluate progress in 2005.

## 1.7 Straw man design concepts

In this section, we collect the straw man design concepts. In Figure 1 we show a sketch of the overall architecture.

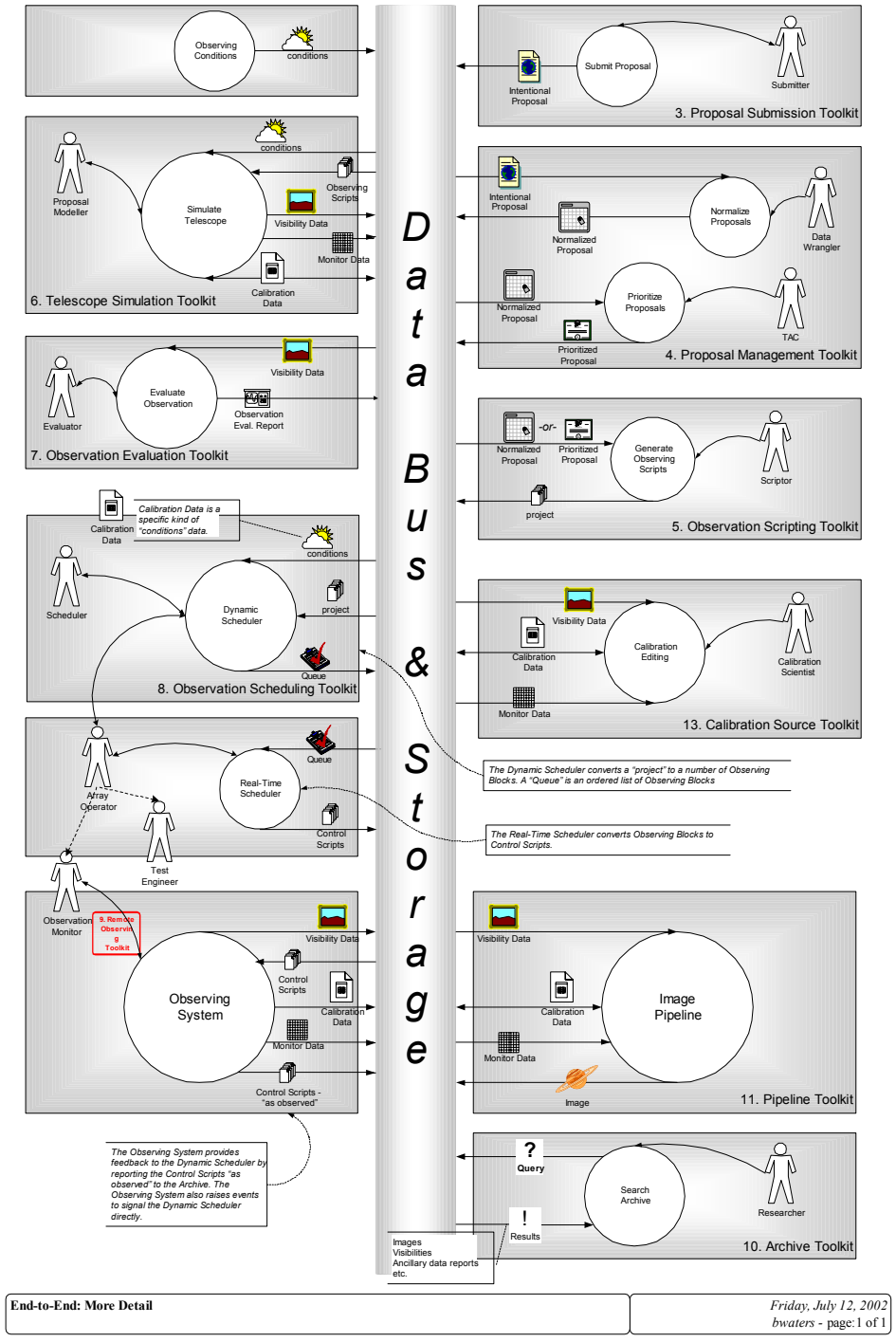



**Figure 2 Overall system architecture**

To emphasize that reuse of existing components is desired and possible, in the next table we describe the nascent ideas for how the different sub-packages are to be realized. We have also denoted the overall priority, which reflects scientific, organizational, and developmental imperatives.

**Table 1 Prototyping**

| From Package | How to prototype? | Priority |
|---|---|---|
| Operational Model | Document | High |
| Proposal Submission Toolkit | Web form or Java-based tool | Medium |
| Proposal Management Toolkit | Java-based tools plus database | Medium |
| Telescope Simulation Toolkit | AIPS++ tools | High |
| Observation Evaluation Toolkit | AIPS++ tools | Medium |
| Observation Scripting Toolkit | GBT Observe, GUI editor | High |
| Remote Observing Toolkit | AIPS++ tools | Low |
| Observation Scheduling Toolkit | OMS + local adaptations | Low |
| Archive Toolkit | AIPS++ plus rdbms? | High |
| Pipeline Toolkit | AIPS++ tools | High |
| Pipeline heuristics | Glish scripts | High |
| Calibration source toolkit | OMS? | High |

## 1.8 Reference Materials

*List all the documents and other materials referenced in this document. This section is like the bibliography in a published book.*

## 1.9 Definitions and Acronyms

*Provide definitions or references to all the definitions of the special terms and acronyms used within this document.*

**Table 2 Definitions and acronyms**

| *Name* | *Definition* |
|---|---|
| AIPS++ | *Astronomical Information Processing System* |
| ALMA | *Altacama Large Millimeter Array* |
| Archive | *A combination of software, hardware, procedures, and operators that stores data from NRAO telescopes for later access and retrieval.* |
| Ancillary data | *Data other than astronomy data (see below) relevant to the time range of interest, such as operator logs, monitor data, open work orders and real-time system SPRs, slowly changing instrumental parameters (e.g. VLA baselines and pointing model parameters), ionospheric data extracted from GPS network, etc.* |
| API | *Application Programming Interface: A language and message format used by an application program to communicate with the operating system or some other system or control program such as a database management system (DBMS) or communications protocol.* |
| Astronomy data | *Whatever is on the telescope's standard raw output data tape. For the VLA, it is correlations, various calibration data (sys/cal, etc.) flags, odds and ends (e.g. weather data), and meta-data as defined below.* |
| Catalog | *Database of radio source properties* |
| EVLA | *Expanded Very Large Array* |
| GBT | *Green Bank Telescope* |
| Glish | *High-level scripting language used in AIPS++ and at the GBT* |
| Heuristic | *A usually speculative formulation serving as a guide in the investigation or solution of a problem* |
| Image | *Multidimensional pixellated representation of sky brightness with axes position, frequency, Stokes parameter, and possibly others. Can mean a simple spectrum.* |
| MeasurementSet | *AIPS++ format for data from an observation using a telescope* |

| *Name* | *Definition* |
|---|---|
| Meta-data | *A description of the telescope setup, containing, but not limited to, the relevant time range, source parameters, receiver setup parameters (especially LO frequencies, and the switch settings used to select bands and filters), the observing proposal in aid of which the observation is made, the observing procedures followed.* |
| Observing script | *A script giving a detailed description of the telescope setup and sequences of sources to be observed with a telescope.* |
| Observation | *A sequence of measurements made by a telescope with some common goal* |
| Pipeline | *A combination of software, hardware, procedures, and people that processes data from NRAO telescopes to some improved form.* |
| Proposal | *A request to the NRAO for telescope time, including scientific justification, target sources, telescope setup, etc.* |
| Reference Image | *An image constructed by applying standard calibration and imaging procedures to a data set.* |
| Schedule | *A sequence of observation scripts to be observed using a specific telescope.* |
| Scheduling | *The process of arranging a schedule of observations for a telescope.* |
| Simulation | *Calculation of the data to be expected in an observation using theoretical and empirical models of calibration effects.* |
| VLBA | *Very Long Baseline Array* |

# 2. OPERATIONAL MODEL

***Author: Tim Cornwell***

***Revision date: 2001/10/25***

***Status: Sufficient for further discussion***

## 2.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The Operational Model is an essential ingredient of the e2e package: it specifies how NRAO telescopes are to be operated in the light of the e2e project. The OM is not a software product but a description of a workflow to be followed when using e2e software products.

The Operational Model includes both commonalities in operation between NRAO telescopes, and specific differences between operations. The purpose is to unify those parts that can be united but not to force agreement is areas that are different for sound reasons.

## 2.2 Background

The NRAO telescopes (VLA, VLBA, GBT) have been developed by different groups of people, but with some overlaps (*e.g.* between VLA Monitor and Control and VLBA). Furthermore, the ways of observing with the telescopes are necessarily different (compare *e.g.* VLBA earth rotation synthesis to GBT single point deep spectrum). Finally, the operation staffs have been quite independent of each other. All of these factors have lead to significant differences in operations, ranging from proposal submission through observation calibration to archiving.

The role of the Operational Model is to specify in detail commonalities and differences in the operations of NRAO telescopes. Since one of the goals of e2e is to bring a consistent look-and-feel to the use of NRAO telescopes, inevitably some of these operational differences must disappear. The Operational Model prescribes operations in the future, when e2e is fully deployed.

## 2.3 Commonalities in operations

The Operational Model is driven by a number of top level decisions *e.g.* the supported modes of observing, the desired data products from the pipeline.

### 2.3.1 Proposal submission and handling

1. Proposals are prepared using one tool for all NRAO telescopes.
2. Proposals are submitted to a single reception point.
3. Proposals are composed of cover information and scientific justification. Cover information (and some other information) is entered into a database, tagged by a proposal code.
4. Proposals are assigned a subject code, from which the pool of referees is derived. A common pool of referees is used by all NRAO telescopes.
5. Referees are assigned to projects, avoiding conflicts of interest, and material sent to the referees for review.
6. Reviews are gathered and entered into the database. Summaries are sent to the Telescope Allocation Committee.
7. There are separate Telescope Allocation Committees for the GBT and for the VLA/VLBA/Global VLBI network.
8. The appropriate person on the TAC negotiates time allocations, and sends tentative schedules to telescope operations periodically. The tentative schedule specifies which programs are to be run, with priorities, and with fixed time slot specifications as appropriate.
9. Telescope operations (or data analysts) then contact the observer requesting an observing script. The observer may either request that an observing script be made by operations staff using the information in the proposal or submit an observing script prepared using NRAO provided tools.
10. Telescope operations then check the script to ensure that it is targeted at the agreed scientific goals.
11. The script is entered into the observing queue for subsequent observation.

### 2.3.2 Observing scripts

An observing script describes the details of observing on a telescope:

- Telescope setup (*e.g.* antennas, receiver selection, LO settings).
- Correlator control (*e.g.* number of channels, polarizations, filtering)

- Sequences of sources to be observed, *etc.*
- Data acquisition (*e.g.* tape recording)

Observing scripts may be generated in two different ways:

- By the observer, using standard NRAO-provided tools.
- By telescope operations, from information in the proposal. The observing script will necessarily use standard calibration procedures.

### 2.3.3 Scheduling

The observing script describes how the data are to be taken. In addition, the TAC or the observer may wish to specify under what conditions the data can be taken. This various additional criteria must be available to describe the conditions under which a script can be observed. These include but are not limited to:

- Universal Time range (joint or periodic observations)
- Local Sidereal Time range (exactly known UV coverage)
- Telescope configurations (VLA configuration)
- Image dynamic range (UV coverage)
- Mosaic dynamic range (pointing, UV coverage)
- Solar RFI (low frequencies)
- RFI from satellites, balloons, other sources
- Polarization observations (parallactic angle coverage, ionosphere)
- Atmospheric transparency (zenith angle)
- Coherence at high frequency (API RMS phase, calibration cycle time)
- Coherence at low frequency (ionosphere)
- Pointing: high frequency mosaicing (reference pointing, wind)
- Spectral dynamic range (bandpass calibration)
- Absolute gain calibration accuracy
- Absolute astrometry
- Bright sources (Van Vleck, Solar cals/attenuators)
- Solar system (ephemeris)
- Pulsars (phased array)

- VLBI (phased array/single dish)

Note that with this definition of observing criteria, there is no meaningful distinction between dynamic and fixed scheduling.

In addition to these criteria, the Telescope Allocation Committee may attach an assessment of scientific priority. For the VLBA currently, the priority of a program depends on the priority assigned by the scheduling committee (only three levels), the referee ratings, an additional increment in priority for subsequent epochs of monitoring programs after the first segment has been scheduled, an additional increment for programs likely to be difficult to schedule (*e.g.*3mm programs), and, for monitoring projects, how close the suggested time comes to the requested interval.

Telescope operations take the observing scripts as submitted by observers and schedules them for execution on the telescope, using a number of criteria to determine the order of execution.

### 2.3.4 Calibration and imaging

A key operational question is how calibration and imaging will be performed. Traditionally, calibration has been largely the observer's responsibility, with the exception that slowly varying terms (baselines, delays, noise tubes) are calibrated by operations. However, this approach is disadvantageous for short, snapshot observations. Instead, we assume that calibration observations will be regularly scheduled as part of telescope operations. The goal will be to provide sufficiently good calibration for typical observations. Very demanding observations (*e.g.* high continuum or spectral dynamic range) may still require dedicated calibration observations.

To limit the range of possibilities that will be supported, we will use templates. Thus an observer can propose to observe a given source, using one of a number of standard templates (*e.g.* continuum image, collection of snapshots, continuum-free spectral-line, on-the-fly continuum image, single-pointing integration, *etc.*) If such a template is used, NRAO will take responsibility for the calibration and imaging. An observer may elect to observe without a template in which case, she will be responsible for calibration and imaging.

The calibration pipeline will produce copious amounts of information on the behavior of a telescope. Telescope operations may choose to mine this information to improve operations. However, this is not (yet) explicitly part of e2e.

### 2.3.5 Interactive observing

Interactive observing is expected to important for the GBT, but less so for the VLA, and VLBA. Interactive observing requires:

- Real-time display of the telescope operation and data acquisition.

- Quick-looks at the data, either as raw data, calibrated data, or quick look images/spectra.
- The ability to change schedules quickly in reaction to results and conditions.
- Communications with operations.

### 2.3.6 Telescope data products

The data products produced by the telescope and associated instruments (*e.g.* weather stations, water vapor radiometers, GPS, *etc.*) determine the type of processing possible. The different telescopes have different definitions of the core astronomical data and the ancillary instrumental data.

- For the VLA, the "data tape" contains coherence measurements, instrumental setup information, flagging information, *etc.* Data from various monitor points around the array is written to a separate "tape" and called monitor data. Operational data such as antenna locations, pointing model, are written to a disk area controlled by operations. Still other data relevant to the telescope, such as the Atmospheric Phase Interferometer, are written to separate disk location.
- For the VLBA, the "data tape" contains more information than for the VLA: tape read statistics and correlation statistics, for example. Other data from the correlator are kept on disk on the *aspen* computer.
- For the GBT, all instruments write data as binary FITS tables. A small number of operational results are written elsewhere.

In general, the distinction between core "astronomical" data and extra "ancillary" data is hard to make and is not a distinction worth building upon. In particular, the meta-data may be drawn from both. Examples of meta-data that could be used are:

- Calibration source type:
    - Band pass, phase, gain, polarization position angle,
    - Astrometric accuracy
- Calibration source data:
    - Parallactic angle coverage for D term calibration,
    - UV spacings vs. source structure for absolute gain
    - Calibration
    - Source models
- Atmospheric opacity: TIPPER or ELINT-type corrections
- Data selection:

- UV coverage
- Elevation
- API rms phase
- GPS rms phase
- Atmospheric opacity (dedicated tipper or VLA tipping scans)
- Solar angle
- Wind velocity
- Cloud cover?
- Reference pointing quality
- Source elevation
- Number of pointings (mosaic)
- Number of channels (line)

### 2.3.7 Pipeline processing

All observations will be processed through a pipeline. The pipeline will provide:

- Basic calibration (*e.g.* for the VLA, amplitude, position (phase), polarization, and bandpass; for the VLBA, also fringe-fitting; for the GBT, on-off, frequency switching).
- Automated (non-interactive) editing, according to the best rules and information available.
- Reference Image/Spectrum
- Observing/Processing History
- Quality assessments There are multiple contexts in which the pipeline can be run:
- During observing, for a quick look at intermediate results, either by telescope operations or by the current observer.
- At conclusion of observing, with standard or with user-specified parameters, for insertion into the archive. These results would be regarded as a cache.
- In batch reprocessing, such as would occur when processing procedures are revised.

- On an archive request, with standard or with user-specified parameters. If the cache contents were up to date, then those would be used, otherwise the cache would be first updated.

### 2.3.8 Best practices

The calibration and imaging procedures used in the processing pipelines will be the "best practices" of data reduction as determined by NRAO scientists and other experts (as captured, for example, in the AIPS Cookbook). The actual procedures and parameters used in any circumstance will be determined from the observing scripts. Accountability of the processing procedures and software configuration (*i.e.* versions of key software) will be maintained.

### 2.3.9 Final products

An observer will have access to a standard set of final products from an observation. The final products from e2e processing will be:

1. Original telescope data files
2. Flagging information
3. Calibrated data
4. Calibration tables
5. Reference images or spectra
6. Observing logs
7. Observing scripts
8. Processing scripts
9. Quality assessments

Final products can be accessed over the Internet or, on request, may be sent on suitable media.

### 2.3.10 Quality assessment

The pursuit of a standard level of quality for the data products is premature. Instead the quality of the reference image or calibration data will be quantified and attached to products. This allows scientific use of the archive but is less demanding of the pipeline scientific performance. Examples of the image quality measures are:

- Image noise compared to theoretical (especially in signal-free regions, polarizations, or channels)

- Dynamic range
- Deconvolution statistics (*e.g.* convergence information)

For calibration tables:

- Calibration solution statistics (*e.g.* rms fit per solution)

For data quality:

- Auto-flagging statistics

These quality measures would be attached to results stored in the archive (*e.g.* an image would have image noise and dynamic range attached as meta-data).

### 2.3.11 Archive use

There are a variety of ways in which the archive will be used:

I. To return the original data independent of the pipeline.

II. To return the reference image for a final scientific result by the original proposer.

III. To return the reference image for a final scientific result by someone else (after the proprietary period).

IV. To return postage stamp previews.

V. To provide images for a finding chart in *e.g.* StarView or an equivalent.

VI. To provide images for an impromptu survey (*e.g.* all observations of Mars, Lockman hole, time-variable source)..

## 2.4 Differences in operations

Nothing here yet.

## 2.5 Scientific requirements

**Table 3 Common Operational Model: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 2.1.1 | 0 | All NRAO telescopes will be operated using a common model |
| 2.1.2 | 0 | Processes, tools, databases, and other resources will be shared between all NRAO telescopes |
| 2.1.3 | 1 | Access to every capability of NRAO telescopes must be maintained |
| 2.2.1 | 0 | All NRAO telescopes will allow interactive observing |

| Ident. | Pri. | Description |
|---|---|---|
| 2.2.2 | 1 | Criteria for scheduling will be telescope dependent |
| 2.3.1 | 0 | All observations from NRAO telescopes will be pipeline processed into a reference data set |
| 2.3.2 | 1 | All pipeline processed observations will be assigned a quality measure |
| 2.4.1 | 0 | All observations will be archived |
| 2.4.2 | 2 | All archives will be made available via the National Virtual Observatory |
| 2.5.1 | 1 | All large-scale surveys with NRAO telescopes will use standard pipeline facilities |

## 2.6 References

# 3. PROPOSAL SUBMISSION TOOLKIT

***Author: Honglin Ye, Tim Cornwell***

***Revision date: 2002/07/11***

***Status: Sufficient for cycle 2 prototyping***

## 3.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The proposal submission toolkit allows scientists to submit proposals to NRAO telescopes, ensuring that the information necessary to the review of the proposal is included and, as much as possible, is correct.

The PST does not require detailed description of the observation (a responsibility of the OST), and it does not have capabilities for detailed planning and evaluation of observations (a responsibility of the OET). However, in simple cases, it must be possible to go from the information in the proposal to a valid observing script.

An observer will use the proposal submission toolkit to construct, verify, and submit a proposal for one or more of the NRAO telescopes. This may be done collaboratively with other scientists at other locations.

## 3.2 Background

For the VLA/VLBA, Barry Clark summarizes the situation:

> *The process starts with the VLA and VLBA observational status reports.*
>
> *They are currently TeX documents, each maintained by a system scientist. They are available on the Web. There is no software associated (a calculator for setups and sensitivities might be nice, but there as been no serious effort at implementation.) Also a component of this early cycle is the VLA configuration announcement in each NRAO Newsletter. I am not aware of the equivalent documents for GBT, but I presume they exist.*
>
> *The next component of the system as currently practiced is the VLA and VLBA coversheets. These are TeX templates, available from the Web. As well as the lines asking for information we need to know to be able to schedule the observation, there are lines asking for computations, to encourage them to catch for themselves impossible observations of one sort or another, and otherwise to think about what they really need. Some people fill in the blanks completely. The GBT equivalent of the proposal coversheets is the Proposal Submission Tool.*

*The current coversheets have evolved over the years and probably contain all or perhaps more than is needed.*

## 3.3 Scientific requirements

*Requirements are to be inserted into a table. A unique number must identify all items. If an item is deleted after the draft, it should be struck out rather than removed from the list. The priority must be chosen from: 0: essential, 1: desirable, 2: if possible. The description should be concise.*

**Table 4 Proposal Submission: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 3.1 | 0 | Produce and submit compliant proposals for all NRAO telescopes |
| 3.2 | 0 | Allow basic user checking of parameters |
| 3.3 | 0 | Allow collaboration between different users in preparing a schedule |
| 3.4.1 | 0 | Provide a Graphical User Interface as prime means of interaction |
| 3.4.2 | 0 | PST runs on local machine. Necessary information is provided at installation. Submission is via email. |
| 3.4.3 | 1 | PST runs on server via web or client interface. Necessary information is downloaded as necessary. Submission is via client-server interface. |
| 3.5.1 | 0 | PST should produce information that is easily put into a database |
| 3.5.2 | 1 | PST should produce information that is editable directly (*e.g.* XML) |
| 3.6 | 2 | PST should provide novice and expert modes |

## 3.4 System requirements

The proposal submission toolkit allows scientists to submit proposals requesting use of the NRAO telescopes for astronomical observations. A proposer (or proposers potentially located everywhere) will construct a proposal and send it to NRAO in electronic form. As such, proposers will use PST over the internet.

A proposal includes a cover page, an abstract and a full text scientific justification. A submitted proposal will be stored in a database.

A user must register with the NRAO to submit a proposal.

**Table 5 Proposal Submission: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | N/A |
| Proposal Management Toolkit | Database based |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | Generate simple scripts from information in proposal |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |
| Pipeline Toolkit | None |
| Pipeline heuristics | None |
| Calibration source toolkit | None |

## 3.5 Design concepts

The PST will use java servlets and HTTP forms. The servlets will run on a web server as a middle layer between requests coming from a web browser and database or applications on the HTTP server. A proposer use HTTP forms to submit the proposal cover page information and upload a full proposal postscript file.

### 3.5.1 Proposal States and Proposal Handling Activities

The proposal submission and management tool kits will be used by 4 types of users. They are proposers, staff scientists (Time Allocation Committee), referees system administrators. The primary states of a proposal include (1) preparation, (2) verification, (3) referee, (4) rating and (5) time allocation. PST and PMT provide appropriate user interfaces for different users and control the data access according to the user roles. PST and PMT handle the data flow and transaction at each of the states. The following figure depicts the major proposal states and user activities.

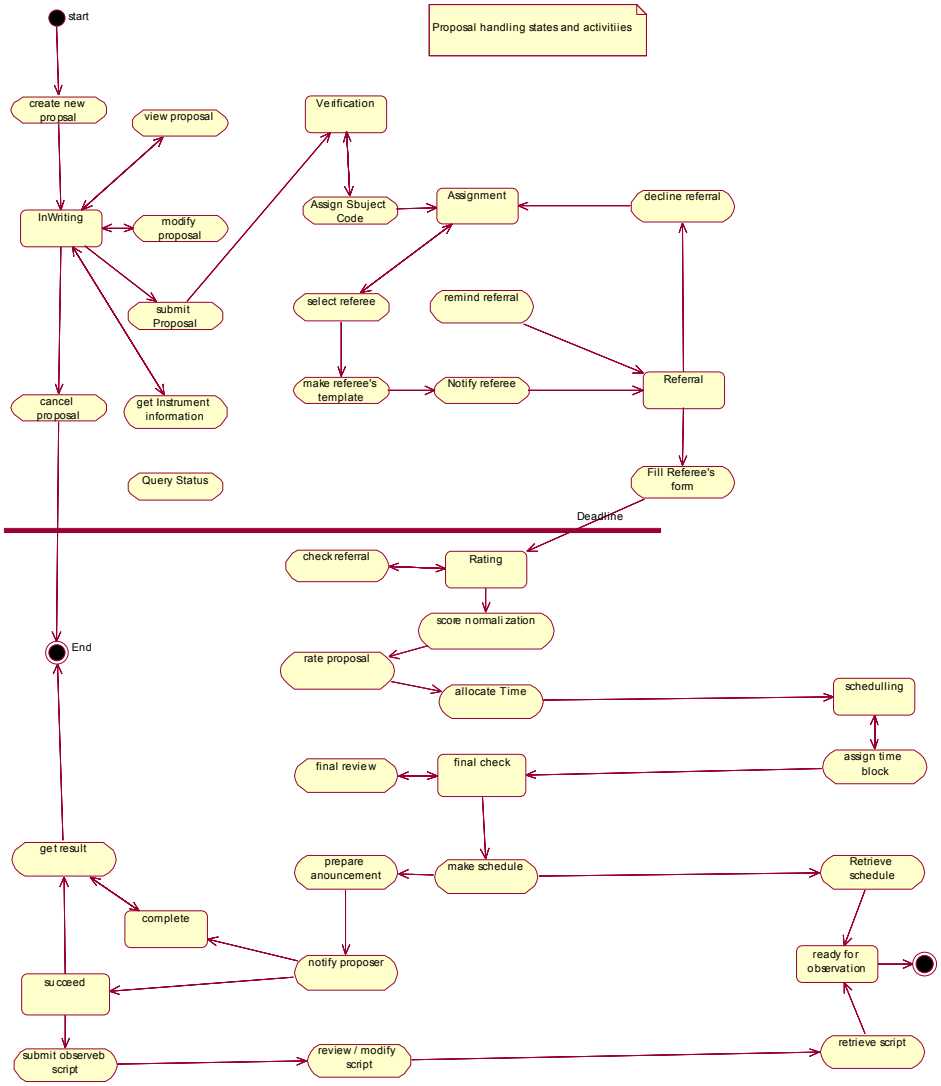



**Figure 3 Proposal states and user activities**

### 3.5.2 User Registration Use Cases

A user is required to register to the NRAO user information database. User access to the proposal handling system and data archive will require a valid user ID, a password and a set role. The user information will be incorporated into a proposal as needed.

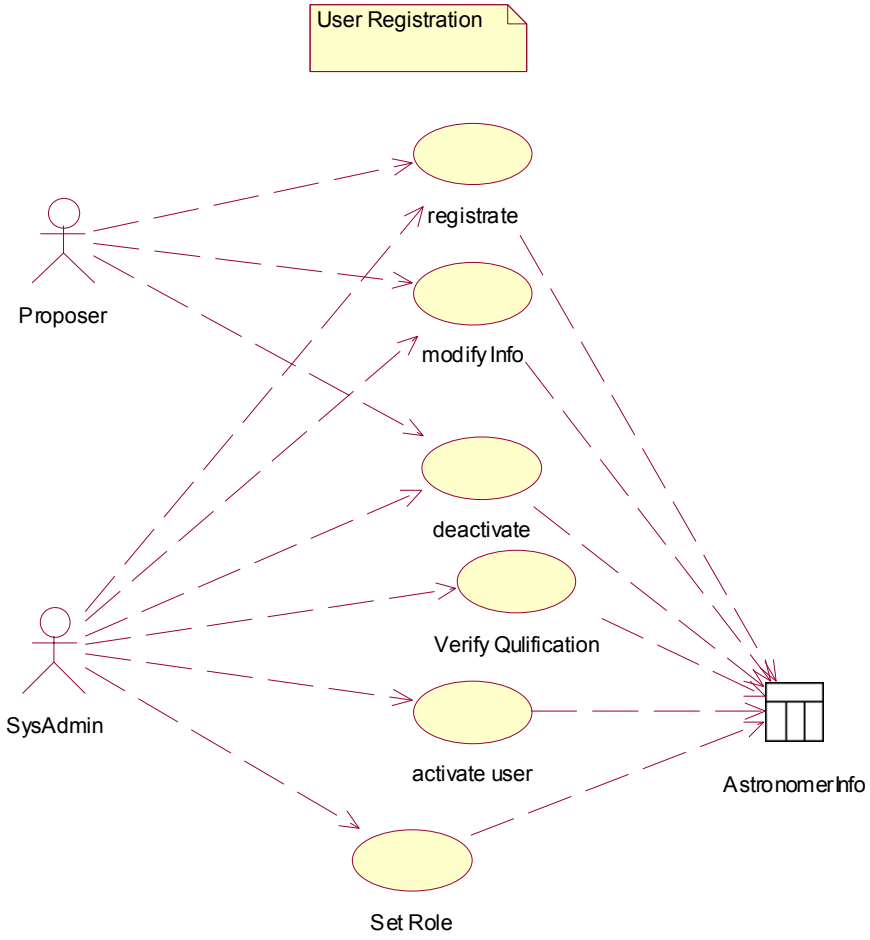



**Figure 4 User registration use cases**

### 3.5.3 Proposal Submission Use Cases

A proposer can create a new proposal by fill a HTTP form. A proposer can attach a full proposal text file. The attached file is required to be in postscript format. The PST does not provide functionality to prepare or to view postscript file. Proposers listed in a proposal header can check proposal status, and make modifications before a proposal is submitted.

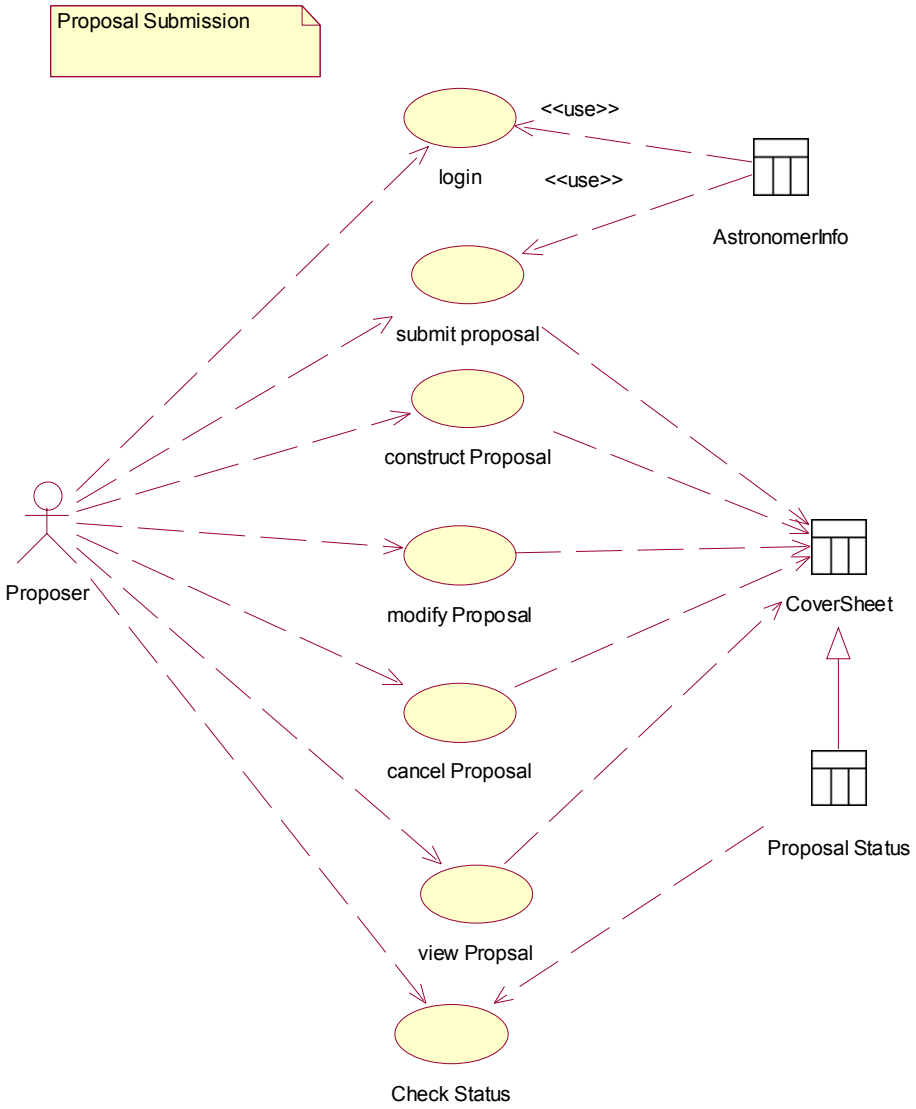


**Figure 5 Proposal submission use cases**

### 3.5.4 Astronomer Information

The stored user information is for communications initiated by the system. A user's referee indicator corresponding to the proposal scientific subject code and is used to select referees.

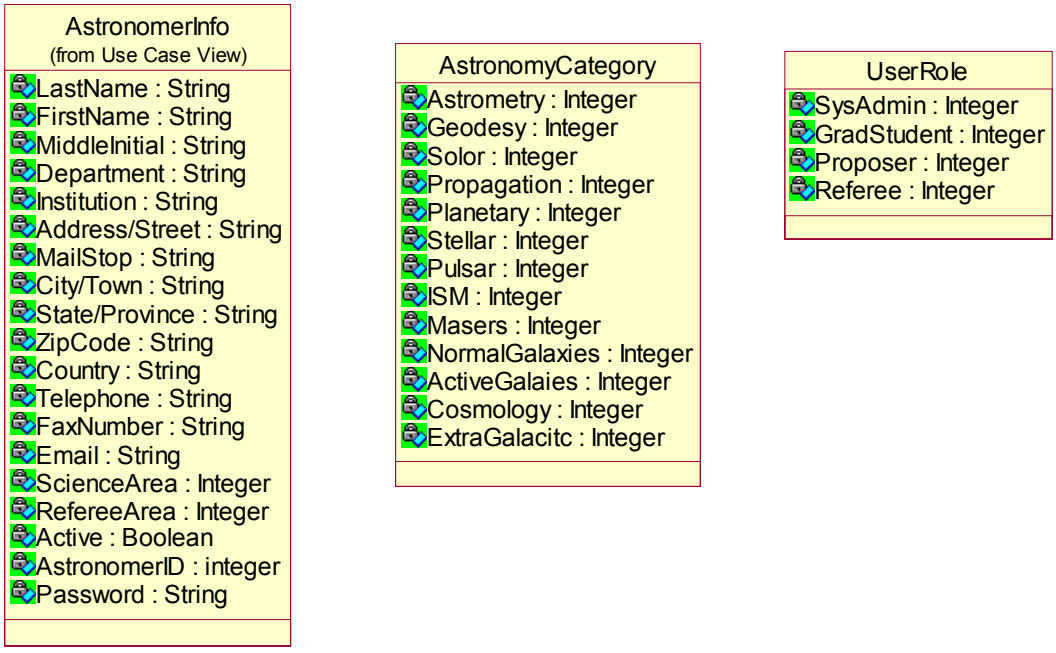



**Figure 6 Astronomer information**

### 3.5.5 Proposal Contents

A proposal includes header information and telescope specific entries. The information required for making observation scripts will not be in the proposal. That information will be submitted later only if the proposal is granted observation time.

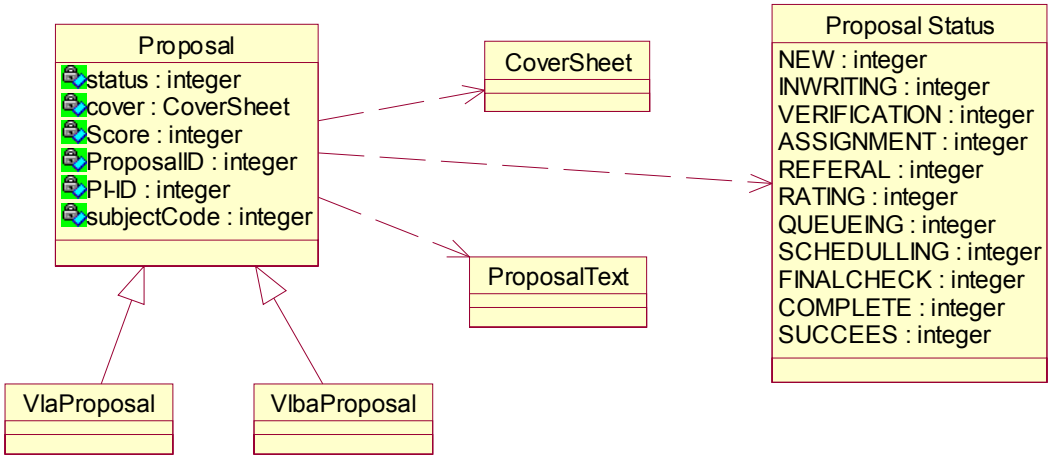



**Figure 7 Proposal concepts**

## 3.6 Implementation

## 3.7 References

# 4. PROPOSAL MANAGEMENT TOOLKIT

***Author: Honglin Ye, Tim Cornwell***

***Revision date: 2002/07/11***

***Status: Sufficient for cycle 2 prototyping***

## 4.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The proposal management toolkit allows facilitates handling of proposals to NRAO telescopes.

## 4.2 Background

For the VLA/VLBA, Barry Clark writes:

> *Proposals for the VLA, VLBA, and Global VLBI Network are sent to Socorro by E-Mail in postscript format. This is a non-negligible impact on net bandwidths, and only in the last couple of years have disks gotten big enough to absorb the load without major pain. I believe that the GBT proposal text is a postscript attachment to the content-parsed material provided by the PST.*
>
> *Material below is for VLA/VLBA/Network; I am most unsure about the GBT equivalents.*
>
> *Proposals are printed as soon as possible after arrival, among other reasons, to make sure they will print on our printers. At this time they are assigned a proposal code.*
>
> *Proposals are given to a staff astronomer. He assigns one or two subject codes, which correspond to referee panels. Paper copies are made for each referee. (Referees were polled about whether they would be happy with an all-electronic system, ala ApJ Letters, but the majority indicated a preference for paper.)*
>
> *Lori Appel enters information about each proposal into a database system (DBase). This includes title, authors, date received, the subject codes mentioned above, and indications of the time requested, bands, VLA configurations (or, for VLBA, non-VLBA antennas requested).*
>
> *Lori then runs a program (written in the obsolete DBase data language), which converts the subject codes into referee panels, and then compares the referee names with the author's names and eliminates referees who are also authors. It then prepares the referee's template form, with a listing of each proposal he is being*

*asked to referee, and contains the proposal code, title, lead author, space for a numerical rating, space for a recommendation of percentage of the requested time to be given, and space for comments. (The rating and time recommendation are regarded as orthogonal - a referee may find a proposal very good science and still recommend little or no time for it, or may find the science god awful, but that if it is to be scheduled it should get 100% of the request.) These templates are sent to the referees by E-Mail. The filled-in templates are to be returned by E-Mail.*

*Lori then types in the source positions given on the proposal coversheets.*

*The current rules are that source lists longer than 30 objects are ignored.*

*(This version of the source list is used to check for conflicts among the proposals, to call up old observations of the same sources, and in some cases, to construct the time allocations for the proposal).*

*I proceed to check the database entries that Lori makes, and edit the proposal titles to a more or less uniform standard. (These titles are what appear in the monthly observing and utilization reports. The titles proposers give are frequently unsuitable for the purpose. There is a tendency to give obscure but catchy remarks that the author thinks will titillate the referees. And some authors are fond of inserting several meaningless words before getting to the substance ("A sensitive VLA observation of....").)*

*I (or sometimes Peggy Perley) use the proposal coversheets to update a database table containing institutional affiliations and E-Mail addresses. The institutional affiliations are used in the monthly observing and utilization reports. The E-mail addresses are essentially my private address list; the primary usage of interest is to get current E-mail addresses to let people know when their observations are being extracted from the archives.*

*I create a database table listing the requested time blocks for each proposal. Some proposers provide this information in adequate form, but they are a minority. And those who do often do not do so in a form sufficiently clear that a secretary could transcribe it; I find it easiest do this myself.*

*When the referee reports are returned, I scan them, primarily to enforce the policy that proposal confidentiality requires that the proposers of competing proposals not be explicitly identified. But I also make minor corrections in spelling and grammar.*

*I then run a C program that converts the referee reports into a Dbase table format.*

*When all referee reports are in, I run a program (in the DBase data language), which calculates for each referee a mean and rms of the numerical ratings. The ratings are normalized to mean 2 and rms 1, and the ratings for all referees for each proposal are averaged.*

*I then run a database report for the scheduling committee, which comprises the following items for each proposal:*

*Proposal code*

*Title*

*Mean referee rating (see above)*

*List of authors*

*List of time blocks requested, by configuration Times previously allocated and scheduled to the proposal. Any remarks by the scheduling committee from previous trimesters Referee code, rating, time recommendation, remarks for each referee List of previous observations of the requested sources (optional)*

*At the time of the scheduling committee meeting, time blocks are tentatively allocated, with a code indicating the order of reconsideration if we allocate too much. At the end of this process, we simply add up the hours allocated for the VLBA fixed and dynamic queues, and compare with what we think we can support. For the VLA we run a program that produces a draft schedule for the configuration, and thereby ensure that the allocated programs will fit.*

*After the scheduling committee meeting, it takes a day or two to review the actions to see if they make sense. Some of the more egregious errors are corrected at this time. And it takes another couple of days to compose comments for the proposers. (If we deviate seriously from the recommendations of the referees, we find it incumbent to explain why; similarly if we make large cuts in time.) When the above are entered, a Dbase report is run which contains essentially the information above. A copy is generated for each person on the author list (the list of previous observations is sent to the lead proposer only, essentially to save paper). A somewhat informative memo is enclosed with each author's referee reports. Lori Appel creates envelops for mailing these reports from the BARS master address list.*

*For those proposals approved for the VLBA dynamic scheduling queue, an E-mail is sent to the contact person, containing a note explaining how to send in their observe files, a repeat of the allocation information sent with the referee reports, and for some proposals, the name of a contact person (experienced observers are not assigned a contact person at this time). At this time too, a program is run transferring the program properties and priorities from the Dbase database to the Ingres database used in VLBA operations. Intervals for monitor programs and preferred dates are added at this time.*

## 4.3 Scientific requirements

*Requirements are to be inserted into a table. A unique number must identify all items. If an item is deleted after the draft, it should be struck out rather than removed from the*

*list. The priority must be chosen from: 0: essential, 1: desirable, 2: if possible. The description should be concise.*

**Table 6 Proposal Management: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 4.1.1 | 0 | Manage proposals for all NRAO telescopes |
| 4.1.2 | 0 | Maintain confidentiality of proposal content, reviews, *etc.* |
| 4.2 | 0 | Proposal can contain cover information and scientific content |
| 4.3 | 0 | Associate subject codes with proposal |
| 4.4 | 0 | Derive suitable referees from subject codes, eliminating self-referees. |
| 4.5 | 0 | Attach multiple reviews to each proposal |
| 4.6 | 1 | Check consistency of addresses with other NRAO databases |
| 4.7 | 0 | Add minimum scheduling information to proposal (*i.e.* requested time blocks) |
| 4.8 | 0 | Provide review summary information for TAC |
| 4.9 | 0 | Attach time allocation information to each proposal |
| 4.10 | 1 | Support mailing to submitters |
| 4.11 | 0 | Provide schedule to telescope operations |

## 4.4 System requirements

**Table 7 Proposal Management: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | N/A |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | None |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |

| From Package | Derived system requirements |
|---|---|
| Pipeline Toolkit | None |
| Pipeline heuristics | None |
| Calibration source toolkit | None |

## 4.5 Design concepts

The PMT will use Enterprise Java Bean technology. One of the reasons to build EJB component is to accommodate both web users and client application program users if it turn out to be necessary.

### 4.5.1 Proposal Verification Use Cases

A staff scientist can check the completeness of the proposal and assign a subject code. The system will use the subject code to find match referees in the astronomer information database. A number of scientists are selected as referees and assigned to evaluate the scientific merits of the proposal. The selected referees are notified through email use the contact information in the astronomer information database. The system will periodically check the status of the proposals under referee and remind referees to complete the assignment.

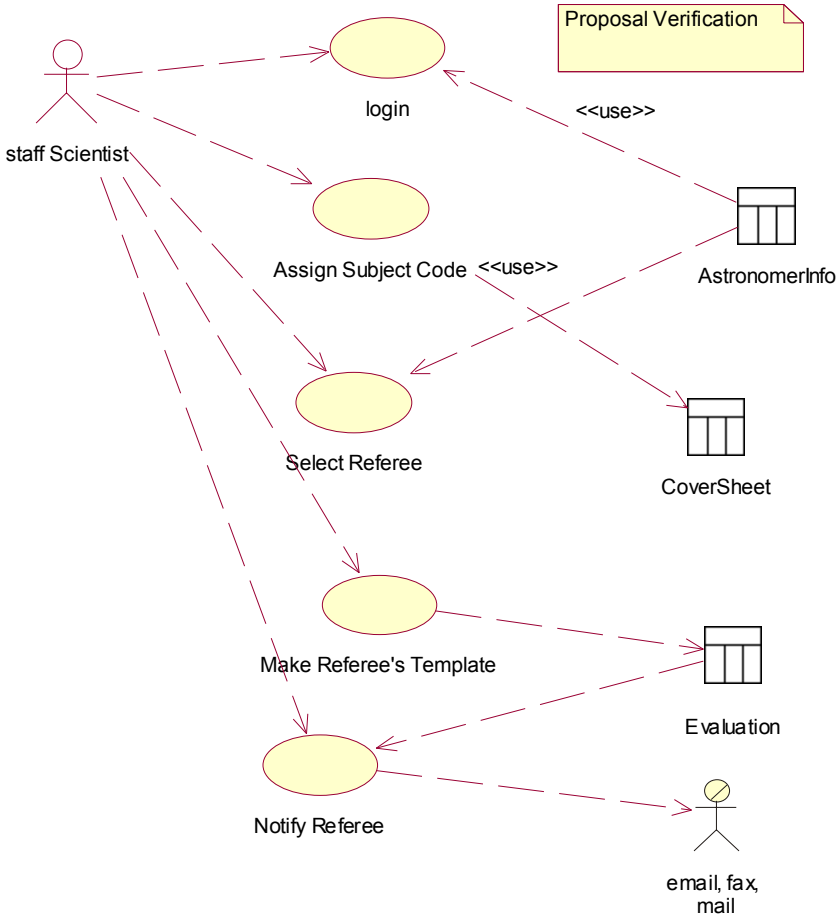



**Figure 8 Proposal verification use cases**

### 4.5.2 Proposal Referee Use Cases

A referee has access to the proposals assignment to him for evaluation. A referee will review the proposal the fill out a referral form. The results from a referee are stored in the database.

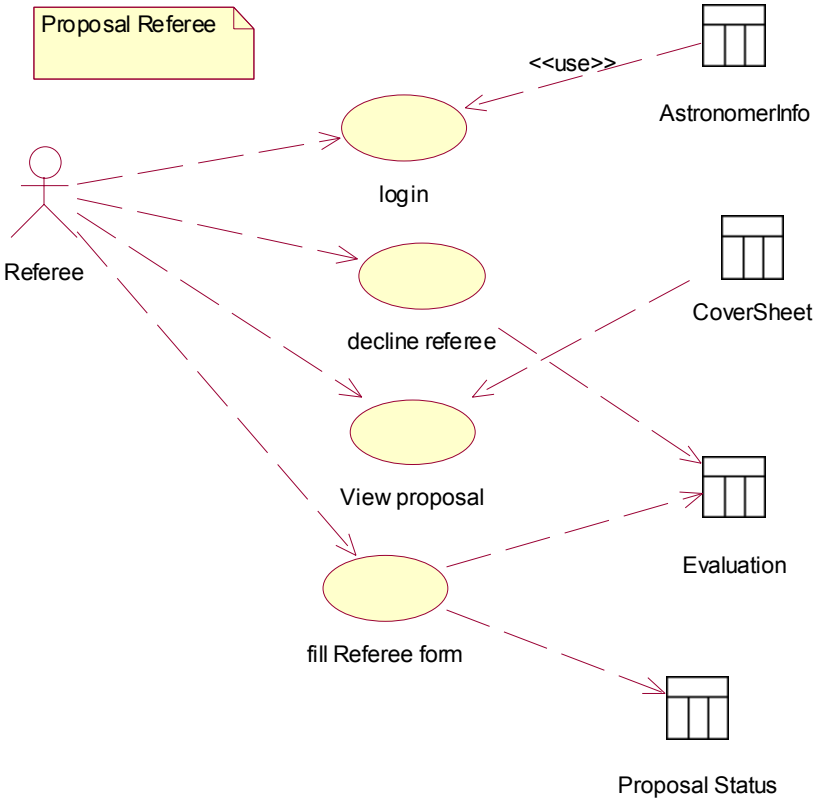



**Figure 9 Proposal referee use cases**

### 4.5.3 Proposal Rating Use Cases

At a preset deadline, rating scores by each referee is normalized against the assemble average. The normalized rating score for each proposal is then averaged and each proposal is given a overall evaluation score. Total required observation time is calculated and highest priority proposals are allocated observation time. Other proposals are to use the available time block according to the priority score. The time allocation at this stage is tentative. The result is a report to the Time Allocation Committee. The results are also stored in a time allocation table.

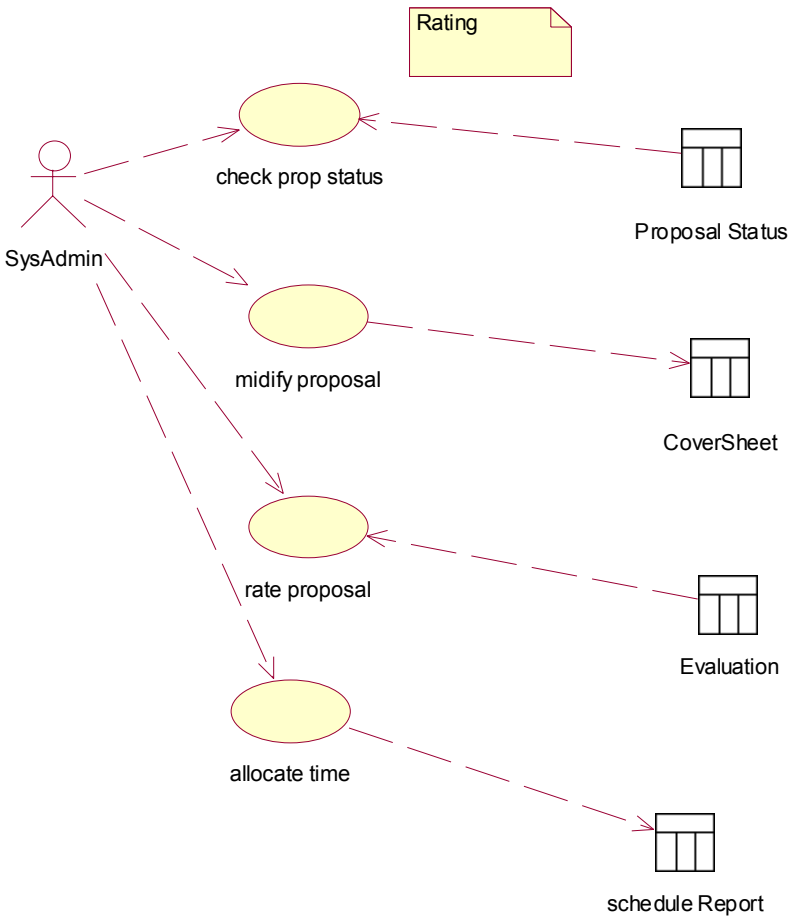



**Figure 10 Proposal rating use cases**

### 4.5.4 Proposal Time Allocation Uses Cases

After the TAC meeting, a staff scientist will set final time allocation based on the TAC decision. The system then generates notification email to all the proposers. The successful proposers are advised to submit observe scripting information or submit an observe script.

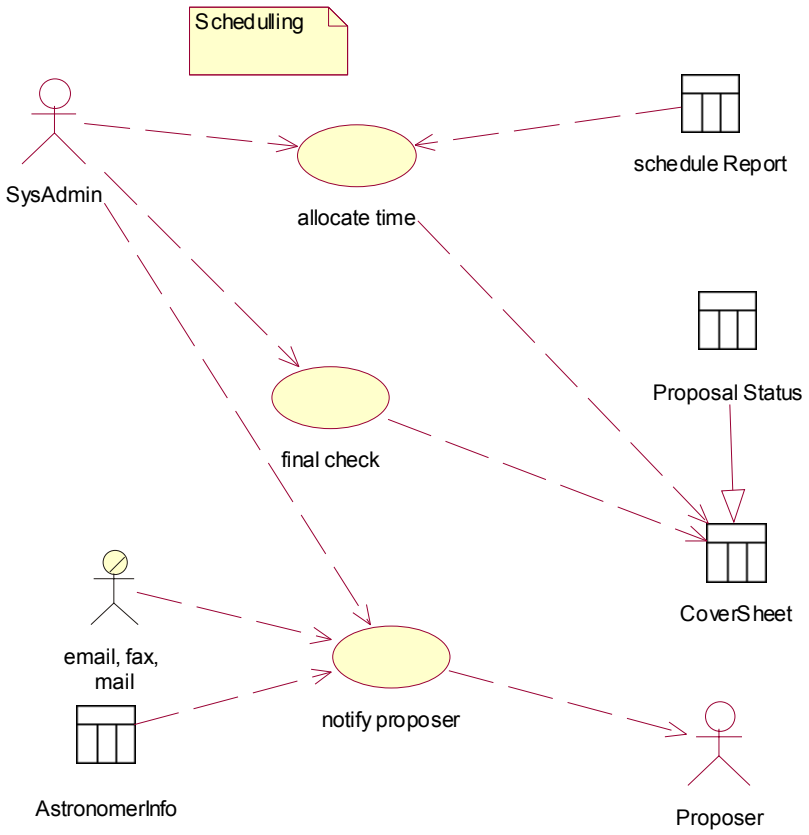



**Figure 11 Time allocation use cases**

## 4.6 Implementation

## 4.7 References

# 5. OBSERVATION SCRIPTING TOOLKIT

***Author: Boyd Waters, Tim Cornwell***

***Revision date: 2002/07/12***

***Status: more developed but still early draft***

## 5.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The observation Scripting toolkit allows description of observations to be made with NRAO telescopes. The result is called an observing script. The script from the OST can be used either for an observation or as input to the Observation Evaluation Toolkit.

The OST is a natural successor to existing tools like JObserve, SCHED, and the GBT Observe. The key innovations over these tools are

1. One tool serves all telescopes.
2. The tool will be able to drive the observation evaluation toolkit.
3. The OST will provide a sufficiently high level description of the observation that the pipeline toolkit can derive suitable processing heuristics.

The script will be in a high level scripting language (possibilities are Glish or the simple GBT observation scripting language).

## 5.2 Background

Classically, synthesis radio telescopes have been scheduled using static ASCII files obeying some well-defined format. Capabilities such as specific shape sky sampling patterns are typically added as extra definitions known to the telescope monitor and control system. Single dish telescopes have been more interactive, and some include provision for writing simple scripts in some command language.

## 5.3 Scientific requirements

The recent work of Rick Fisher for the GBT Observe program shows one straightforward way to proceed (see *e.g.* http:/www.nrao.edu/~rfisher for more detailed examples). He defines a simple scripting language ("observing tables") that is translated into Glish for execution when observing. We sketch the layering of software in Figure 4.1. The Proposal Submission Toolkit may generate a simple observing table, or a user may generate a table using a specially written GUI, or, as an interim measure, using JObserve

or Sched and a translator. New observing procedures may be specified as Glish functions, and accessed as such from the observing table.

It is not clear that the GBT Observe can be adopted directly but it does show an approach to layering software that can be used.

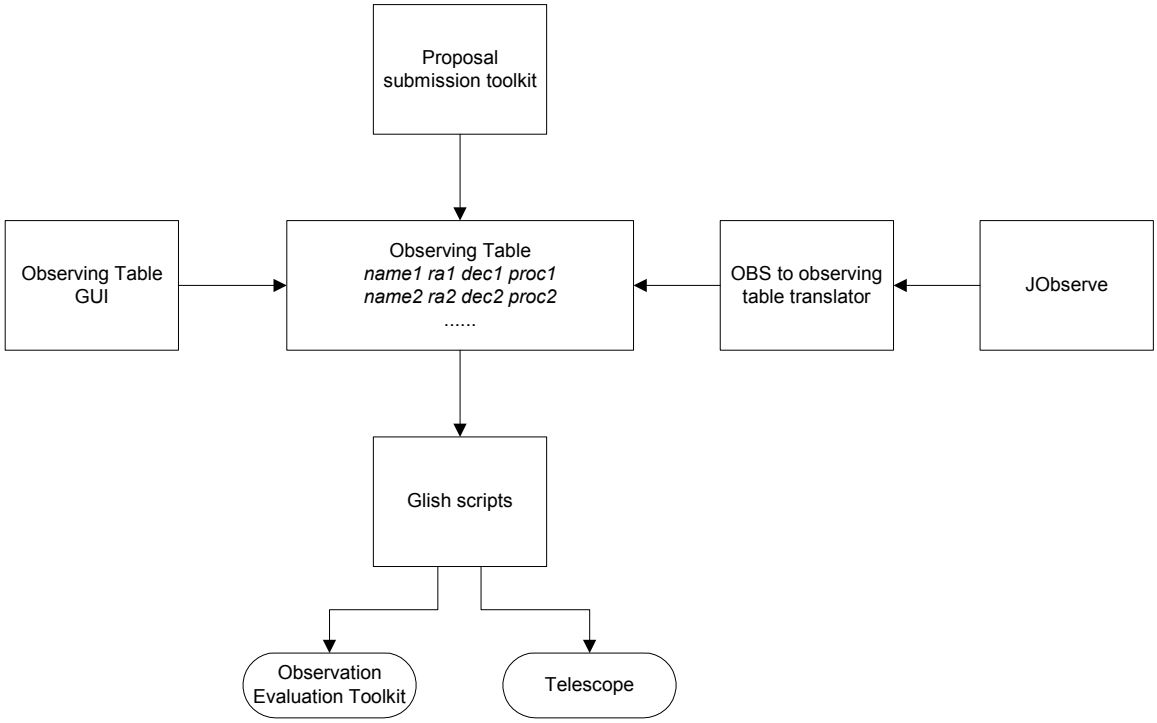



**Figure 11 Layering of software if GBT Observe is adopted.**

The OST must also check any schedule for correctness in a number of areas:

- LO settings
- Slew times
- Elevation limits
- Shadowing
- Internal radio frequency interference
- External radio frequency interference (well known sources)
- Data rate limitations
- Proximity of strong sources (e.g. Sun, planets, Cygnus A, *etc.*)

The Observing script must allow specification of criteria for scheduling (see section 7 for details) and information for pipeline processing (see sections 10 and 11).

**Table 8 Observation Scripting: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 5.1.1 | 0 | Describe observations with NRAO telescopes |
| 5.1.2 | 1 | Provide one interface for all NRAO telescopes |
| 5.1.2 | 1 | Text-based, human-readable, and human-editable format |
| 5.1.3 | 1 | Graphical user interface for novices |
| 5.1.4 | 2 | Expert mode for experts |
| 5.1.5 | 0 | Verify observing scripts for correctness |
| 5.1.6 | 1 | Check observing scripts for minimum standards (*e.g.* minimum time on source) |
| 5.2.1 | 0 | Allow specification of a range of criteria for scheduling |
| 5.2.2 | 0 | Allow specification of pipeline processing |
| 5.2.3 | 0 | Allow use of default or user-defined observing procedures |
| 5.2.4 | 0 | Allow use of default or user-defined observing setups |
| 5.2.5 | 0 | Allow use of default or user-defined processing procedures |
| 5.3.1 | 0 | Allow use of information from existing NRAO catalogs and images |
| 5.3.2 | 1 | Allow use of existing non-NRAO catalogs and images |

## 5.4 System requirements

**Table 9 Observation Scripting: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | Generate simple scripts from information in proposal |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |

| From Package | Derived system requirements |
|---|---|
| Observation Scripting Toolkit | N/A |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |
| Pipeline Toolkit | Observation scripts can be digested by pipeline |
| Pipeline heuristics | None |
| Calibration source toolkit | None |

## 5.5 Design concepts

### 5.5.1 GBT Observe as a Model

We will adopt and/or adapt Green Bank Telescope Observe. The format for schedules is an observing table. GUI-based tools will be written to write such scripts. The observing table can call upon Glish-based functions.

### 5.5.2 Other Approaches Considered

In keeping with a "simplest is best" approach, at this time we are assuming the descriptions will be implemented following the design of GBT Observe. But here are some other ideas:

#### 5.5.2.1 Scripts

At the moment it appears desirable to describe the observations in a simple, text-based table model. Such a representation could be thought of as a scripting language with a very simple syntax; this simple representation gets "compiled" into a computer program implemented in another language (Glish).

We could implement the Observation Description in a "full", extant scripting language such as Glish or JPython, but in practice we would use function libraries so that such a representation would be tantamount to our intermediate table format anyway.

We have not yet identified a compelling reason to pursue such a "full-blown" scripting language as the standard representation for the Observation Description.

If "black belts" wish to implement custom routines in such a script, they may do so and then call the scripts from the observing table.

#### 5.5.2.2 Objects

The ALMA Observing Tool (http://www.ovro.caltech.edu/scott/ALMA/OT/) attempts to provide a rich, graphical environment for the novice array user. ALMA OT will be implemented in Java.

It is possible that the observation descriptions are themselves Java objects; they can be stored on disk using the standard Java Development Kit "serialization" methods (http://java.sun.com/j2se/1.4/docs/api/java/io/Serializable.html) or by developing a custom format (such as an XML binding). Such a design emphasizes the run-time representation of the Observation Description; such representations are not to be modified outside of the Toolkit.

An interesting point: the Observation objects could be directly invoked by a telescope scheduling system, perhaps using a remoting mechanism such as CORBA or Java RMI.

#### 5.5.2.3 XML

Put simply, XML is a standardized way to represent structured textual data in a manner that is possible to be read by people and computers.

XML is almost certainly less readable, as a "raw" format, than the text-based observing table.

Extensive experimentation with XML leads us to believe that the benefits of XML (representation of tree structures, standardized parsers and data transformations), while interesting, are not sufficiently compelling to require XML's adoption for the Observation Description.

## 5.6 Implementation

## 5.7 References

# 6. TELESCOPE SIMULATION TOOLKIT

***Author: Tim Cornwell***

***Revision date: 2002/07/12***

***Status: early draft***

## 6.1 Summary

The Telescope Simulation Toolkit simulates observations with one of the NRAO telescopes, working from a script generated by the Observation Scripting Toolkit.

## 6.2 Background

## 6.3 Scientific requirements

The AIPS++ simulator is designed to simulate the data collected by a radio telescope. Modeling of the observation process is via the Measurement Equation formalism in AIPS++, and the data are stored in an AIPS++ MeasurementSet. Definition of the observing setup and data collection strategy is currently hard-coded into the interface for the simulator. This will be evolved to match the Glish used in the Observation definition.

The simulator must be able to handle the following:

- UV coverage
- Thermal noise
- Source models (Gaussian, points, disks, images, catalogs)
- Calibration errors (both systematic and random)
- Pointing errors
- Mosaics
- Spectral line calibration effects (*e.g.* band pass ripples)

The following are desirable:

- Realistic simulations of the sky source distribution
- Tropospheric phase errors
- Ionospheric phase errors
- Interference

Current versions of the AIPS++ simulator are driven by various input parameters that can invoke various observing scenarios. It would be more useful if a template-based observing script could drive the simulator.

**Table 10 Observation Simulation: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 6.1.1 | 0 | Verify observing script for correctness |
| 6.2.1 | 0 | Simulate error-free MeasurementSet from target |
| 6.2.2 | 1 | Simulate simple noise processes in MeasurementSet from target |
| 6.2.3 | 1 | Simulate calibration errors in MeasurementSet from target |
| 6.2.4 | 2 | Simulate second-order calibration errors in MeasurementSet from target |
| 6.3.1 | 0 | Scale standard test images and spectra appropriately |
| 6.3.2 | 1 | Generate test image or spectrum using a variety of algorithms |
| 6.3.3 | 2 | Generate test image or spectrum acquired from external source |
| 6.4.1 | 0 | Available as native facility in AIPS++ |
| 6.4.2 | 2 | Available via web service hosted on NRAO computers |

## 6.4 System requirements

**Table 11 Observation Simulation: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | N/A |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | Can execute from Observation Scripts |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |

| From Package | Derived system requirements |
|---|---|
| Pipeline Toolkit | None |
| Pipeline heuristics | None |
| Calibration source toolkit | None |

## 6.5 Design concepts

The current AIPS++ simulator is limited in its ability to generate arbitrarily complex MeasurementSets by the user interface. One answer is to use a scripting language to specify the observations. Since this is similar to the role of the Observation Scripting language, we plan to use the same solution for both purposes. This has the advantage that complete observations can be simulated and then analyzed prior to any actual telescope monitor and control system being able to digest observing scripts.

## 6.6 Implementation

## 6.7 References

# 7. OBSERVATION EVALUATION TOOLKIT

***Author: Tim Cornwell***

***Revision date: 2002/07/12***

***Status: Sufficient for prototyping in cycle 2***

## 7.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The observation evaluation toolkit allows evaluation of observations using NRAO telescopes. It is a collection of tools that may be used by a scientist in conjunction with the Observation Scripting Toolkit to aid the planning of an observation. Factors relevant to the scientific success of an observation are calculated and displayed.

The OET must work in close conjunction with the OST and TST to allow a scientist to explore many options easily.

## 7.2 Background

## 7.3 Scientific requirements

*Requirements are to be inserted into a table. A unique number must identify all items. If an item is deleted after the draft, it should be struck out rather than removed from the list. The priority must be chosen from: 0: essential, 1: desirable, 2: if possible. The description should be concise.*

**Table 12 Observation Evaluation: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 7.1.1 | 0 | Verify observing script for correctness |
| 7.1.2 | 0 | Provide first level statistics of telescope use |
| 7.2.1 | 0 | Available as native facility in AIPS++ |

## 7.4 System requirements

**Table 13 Observation Evaluation: System Requirements**

| From Package | Derived system requirements |
|---|---|

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | Results from evaluation can feed back to simulation |
| Observation Evaluation Toolkit | N/A |
| Observation Scripting Toolkit | None |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | Results from evaluation can be stored in archive |
| Pipeline Toolkit | None |
| Pipeline heuristics | Results from evaluation can be used by heuristics |
| Calibration source toolkit | None |

## 7.5 Design concepts

The Observation Evaluation Toolkit summarizes an observation (or result of merging several observations), and then publishes the result in some format.

Assessing the quality of an observation is difficult. Ideally one would like a simple ranking, such as a number from 1 – 10 measuring the number of performance metrics met. This would be simplest if the user has specified performance metrics that must be met: *e.g.* noise < 1.5* thermal, dynamic range > 10000, spectral dynamic range > 1000, flux calibration error < 2%, image fidelity, polarization position angle error < 3 degrees, *etc.* We can calculate each of these via some standard approach and then attach each to a project summary, aiming for good coverage of all aspects of an observation. This is the approach that we use in this prototype.

Useful performance metrics can be split into a number of categories: Project, Observation, Editing, Calibration, and Imaging.

**Table 14 Metrics for a project**

| Project Metric | Nature of metric | Method of calculation |
|---|---|---|
| Completeness | Dimensionless number in range 0 to 1 and a bit | From TAC |

**Table 15 Metrics for an observation**

| Observation Metric | Nature of metric | Method of calculation |
|---|---|---|
| Time observed / scheduled | Dimensionless number in range 0 to 1 | Need observing script and simulator |
| Wind | Fraction of time wind > 10 m/s | Weather station data |
| Phase stability | R0/Baseline length | API |
| Transparency | Atmospheric optical depth | Tipping radiometer |

**Table 16 Metrics for editing of an observation**

| Editing metric | Nature of metric | Method of calculation |
|---|---|---|
| Quack | Fraction | Fraction of data lost to Quac |
| Time consistency | Fraction | Fraction of data lost to time consistency editing |
| Frequency consistency | Fraction | Fraction of data lost to frequency consistency editing |

**Table 17 Metrics for calibration of an observation**

| Calibration Metric | Nature of metric | Method of calculation |
|---|---|---|
| Calibration | Dimensionless number in range 1 to ??? | Median (calibration fit / theoretical noise) |
| R/L consistency | Dimensionless number in range 0 to 1 | Min (R,L calibration)/Max (R,L calibration) |
| Flux calibration consistency | Dimensionless number in range 1 to ??? | Median (calibrator flux / error) |
| Polarization angle error | Angle ~ few degrees | From ???? |

**Table 18 Metrics for images**

| Image Metric | Nature of metric | Method of calculation |
|---|---|---|
| I noise | Jy/beam or K | Robust statistics in blank region of Stokes I image |
| V noise | Jy/beam or K | Robust statistics in blank region of Stokes V image |
| Noise/Theoretical | Dimensionless number in range 1 - ?? | I noise, and simulations |
| I/V noise ratio | Dimensionless number in range 1 – 10 | Ratio of I to V noise |
| Dynamic range | Dimensionless number in range 5 – 100000 | Peak flux / I noise |
| Deconvolution stability | Jy/beam or K | Robust statistics of differences between MEM and CLEAN images |
| Image fidelity | Dimensionless number in range 1 – 100 | Median (Flux / Deconvolution stability) |

It should be emphasized that this is only an initial list. Other candidate measures will certainly arise over time.

The metrics should be stored in tables linked to the other catalogs.

## 7.6 Implementation

## 7.7 References

# 8. OBSERVATION SCHEDULING TOOLKIT

***Author: Boyd Waters, Tim Cornwell***

***Revision date: 2002/07/12***

***Status: more developed but still early draft***

## 8.1 Summary

The Observation Scheduling Toolkit allows NRAO telescope operators to schedule a telescope from submitted observing scripts. A telescope operations staff will receive observation scripts from observers. The staff will then feed these to a queue for subsequent observation. The OST is not controllable by an observer but the current queue may be viewed by anyone.

## 8.2 Background

For the VLA/VLBA, Barry Clark writes:

> *The VLBA dynamic scheduler is a C program named scenario. It produces a large number of possible scenarios of observations for a few days ahead, and orders them by a figure of merit based on the priorities of the programs they contain. (The operator can select a scenario other than the first ranked, in case he knows something the program doesn't.) The priority of a program depends on the priority assigned by the scheduling committee (only three levels), the referee ratings, an additional increment in priority for subsequent epochs of monitoring programs after the first segment has been scheduled, an additional increment for programs likely to be difficult to schedule (e.g. 3mm programs), and, for monitoring projects, how close the suggested time comes to the requested interval. The scenario program is run through OMS via a Java servlet. OMS can display the various scenarios, and can allow the operator to pick the next program to observe, and to run sched to produce the crd files for the stations.*

## 8.3 Scientific requirements

Telescope observing scripts are constructed using the Observation Scripting Toolkit. These scripts are then sent to telescope operations for observing. Various criteria may be used to determine when a script can be observed. These include but are not limited to:

- Universal Time range (joint or periodic observations)
- Local Sidereal Time range (exactly known UV coverage)
- Image dynamic range (UV coverage)

- Mosaic dynamic range (pointing, UV coverage)
- Solar RFI (low frequencies)
- RFI from satellites, balloons, other sources
- Polarization observations (parallactic angle coverage, ionosphere)
- Coherence at high frequency (API RMS phase, calibration cycle time)
- Coherence at low frequency (ionosphere)
- Pointing: high frequency mosaicing (reference pointing, wind)
- Spectral dynamic range (bandpass calibration)
- Absolute gain calibration accuracy
- Absolute astrometry
- Bright sources (Van Vleck, Solar cals/attenuators)
- Solar system (ephemeris)
- Pulsars (phased array)
- VLBI (phased array/single dish)

In addition to these criteria, the Telescope Allocation Committee may attach an assessment of scientific priority. For the VLBA currently, the priority of a program depends on the priority assigned by the scheduling committee (only three levels), the referee ratings, an additional increment in priority for subsequent epochs of monitoring programs after the first segment has been scheduled, an additional increment for programs likely to be difficult to schedule (*e.g.*3mm programs), and, for monitoring projects, how close the suggested time comes to the requested interval.

**Table 19 Observation Scheduling: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 8.1.1 | 0 | Aid telescope operations staff in constructing observing schedule from submitted observing scripts |
| 8.1.2 | 0 | Allow multiple scheduling criteria per script |
| 8.1.3 | 0 | Present multiple, ranked scenarios |
| 8.1.4 | 0 | Allow operator to select next program to observe |
| 8.2.1 | 0 | Provide Graphical User Interface for operation |
| 8.3.1 | 0 | Allow operator to |

| Ident. | Pri. | Description |
|---|---|---|
| 8.3.1 | 2 | Allow use of standard scheduling engines (*e.g.*. SPIKE or equivalent) |
| 8.4.1 | 0 | Summarize current status of projects: partially or completely satisfied |
| 8.4.1 | 1 | Display results to web |

## 8.4 System requirements

**Table 20 Observation Scheduling: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | Observation Scripts can be scheduled |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |
| Pipeline Toolkit | None |
| Pipeline heuristics | None |
| Calibration source toolkit | None |

## 8.5 Design concepts

### 8.5.1 Dynamic Scheduling

The term, “dynamic scheduling” means that we seek to structure an observation so that we can respond to events on a short time scale. Dynamic Scheduling allows the system 1) to optimize the course of the observation execution for particular observing conditions (e.g., weather), and 2) to respond to targets of opportunity (e.g., a Gamma-Ray Burst).

### 8.5.2 Observing Blocks

In order for the telescope to respond to changing conditions, an observation is split into a stream of discrete *observing blocks*, nominally covering a 20-minute time range.

It is important to note that an observing block does not know when it will be run on the telescope! Therefore the observing block holds information about the conditions necessary for it to be run, including instrument configuration and other constraints.

An observing block is a series of instructions for the telescope, called the observing block *body*, along with a *preamble* (executed at the beginning of the block, before the body), and a post-amble (executed at the end of the block, after the body).

An observing block also contains information that allows the dynamic scheduler to order a series of blocks into a prioritized queue: the scheduling constraints and the time allocation committee ranking. Finally, a block knows which observing program it belongs to.

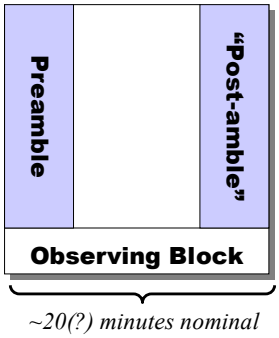



**Figure 12 An Observing Block.**

An observing block – the block body, preamble, and post-amble – are each comprised of Glish scripts. In general, these scripts will be standardized libraries stored in a script template repository.

The system will maintain a repository of "template" blocks for standardized tasks, including standard pointing runs for use by observers and "debugging" blocks for use by telescope engineers.

At this point it is easiest to think of a scheduling block as a row in a table, with the block components as columns, as shown in Figure 12.

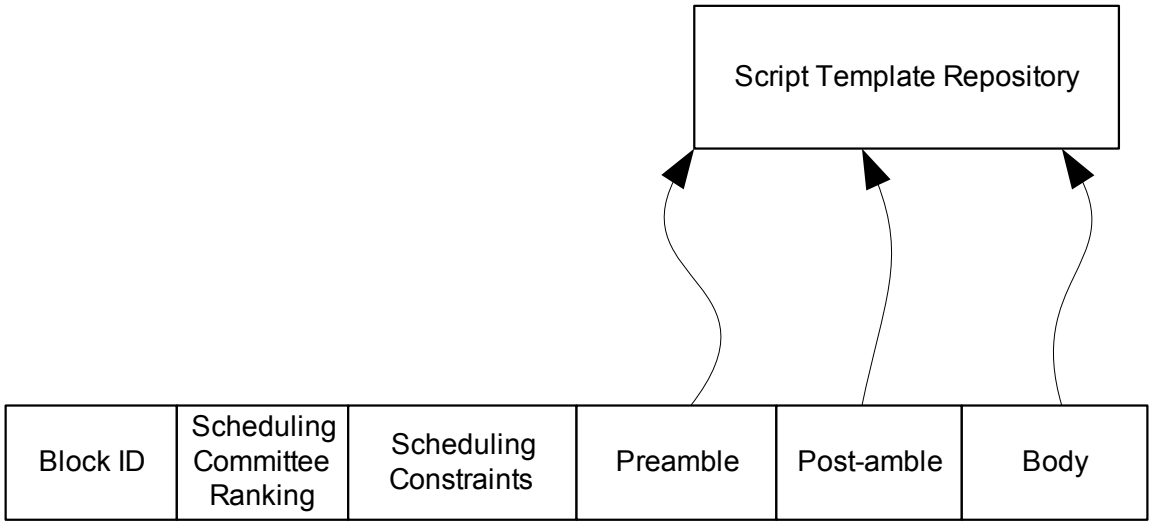



**Figure 13 Observing Block Detail**

#### 8.5.2.1 Block ID

The Block ID is a unique identifier for the observing block. The Block ID will include a reference to the block's observing program.

#### 8.5.2.2 Scheduling Committee Ranking

The Time Allocation Committee (TAC), discussed in the Proposal Management Toolkit, is responsible for ranking observing programs. An observing block generally inherits the ranking of its program. (Therefore it may be that the reference to the observing program is sufficient. But we include the ranking explicitly here at this stage in the design.)

#### 8.5.2.3 Scheduling Constraints

Constraints may be placed on the execution of an observing block. Some illustrative examples:

##### 8.5.2.3.1 ALGORITHMIC CONSTRAINTS

"Run this block until the calibrations converge"

Blocks may have dynamic run-time requirements. Although a block may nominally cover a 20-minute range, a block's actual "wall-clock time" when run on the telescope will generally be more than twenty minutes.

#### 8.5.2.3.2 INTER-BLOCK CONSTRAINTS

"I can't run unless the previous block has run successfully"

It may be that, rather than enforce inter-block constraints, we simply collect all such blocks into a single block.

#### 8.5.2.3.3 TIME CONSTRAINTS

"I MUST run at 16:42:30 GST on 30 May 2002"

 Fixed, or "non-dynamic", scheduling is dynamic scheduling with time-domain constraints.

### 8.5.2.4 Preamble

This is the series of commands which are to be run before the body of the script. Example: a flux calibration routine.

### 8.5.2.5 Post-amble

Like the preamble, but run after the body of the block has finished execution. Example: polarization calibrations.

### 8.5.2.6 Body

The body of the observing block contains routines germane to the observing program.

### 8.5.3 Scheduling Phases

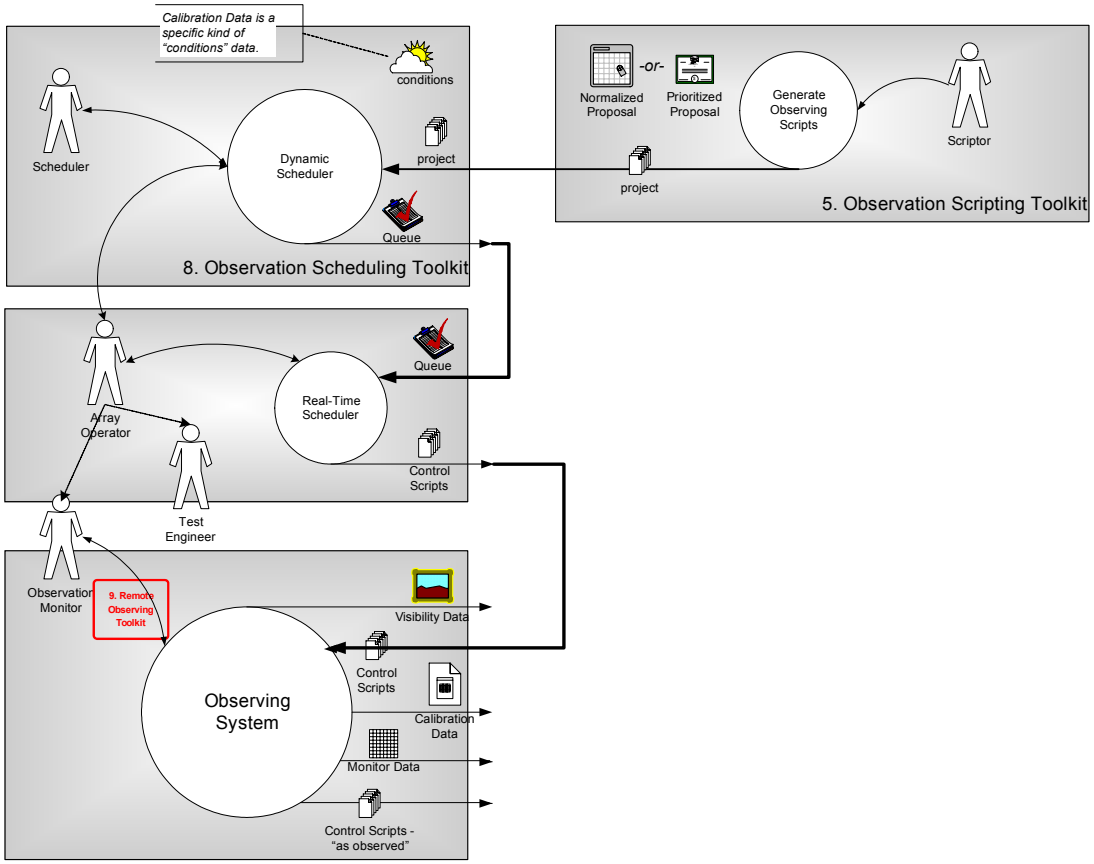



**Figure 14 - Phases of an Observation Schedule**

Figure 13 is taken from the architecture diagram, and makes explicit the flow of an Observing Schedule from creation (by the Observation Scripting Toolkit) through *Dynamic Scheduling* (discussed below), and ultimately execution on the telescope Observing System.

The Scheduling Toolkit will generate the Observing Blocks.

It would be unbearably tedious for a human to do so, given the degrees of freedom involved.

The process is analogous to a two-pass complier: one pass is from the Observation Description into Observation Blocks, and another pass is from an Observing Block to a Control Script.

The Scheduling Toolkit generates blocks. The Telescope system generates Control Scripts.

We had to come up with some arbitrary terminology to represent the observation throughout the various phases of scheduling: "*project*", "*queue*", and "*control script*".

#### 8.5.3.1 "Project"

A Project is a complete description of an observation that is the output of the Observation Description Toolkit.

#### 8.5.3.2 "Queue"

The Dynamic Scheduler breaks the Project into Observing Blocks, and then evaluates a number of Observing Blocks to produce an ordered list of them: the Queue.

#### 8.5.3.3 "Control Script"

A Control Script is the thing that gets evaluated by the telescope Observing System; it tells the telescope what to do for a specific period of run time. It is the responsibility of the Telescope system (the "real-time scheduler") to generate Control Scripts from Observing Blocks.

### 8.5.4 Telescope Interaction: General

#### 8.5.4.1 Queue ↔ Telescope Interaction

##### 8.5.4.1.1 ONE BLOCK AT A TIME

As shown in Figure 14, the Real-Time Scheduler waits for a "block ready" request from the Telescope Observing System, and replies to such requests by issuing the next block in the queue.

- Simplifies the telescope state data
- Telescope reports block execution status back to the block queue
- All "observing logic" is maintained by the Scheduling Toolkit

This last point – the maintenance of the observing logic by the Scheduling Toolkit – emphasizes the decoupling of the telescope logic and the scheduling logic: The Scheduling Toolkit knows about blocks, and how to order the blocks given certain heuristics about observing conditions. The telescope knows how to execute a block.

In this way we attempt to reuse the Scheduling Toolkit across multiple instruments: we will develop a detailed design API for the expression of these observing heuristics (in order to implement scheduling), and a detailed design API for the queue-to-telescope protocol.

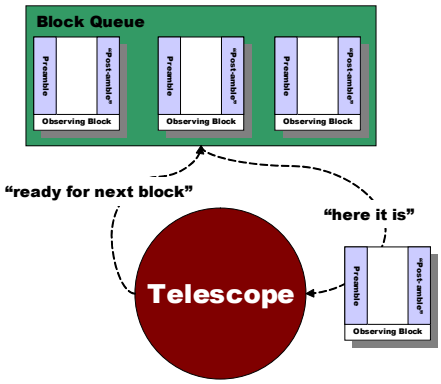



**Figure 15 One Block at a Time.**

##### 8.5.4.1.2 BLOCK PRE-EMPTION

The Dynamic Scheduler may terminate a block that is in-progress.

There may be at least two forms of pre-emption: terminate-request and terminate-immediate.

Terminate-request lets the Telescope know that it should finish the current block within a time constraint. The post-amble of the block should be executed.

Terminate-immediate will cancel execution of the currently-running block.

##### 8.5.4.1.3 BLOCK EXIT STATUS

A block may run successfully, may run unsuccessfully, or may be pre-empted.

An "unsuccessful" run of a block is one which fails to meet the block constraints. For example, a block may "give up" after a certain period of time if a calibration solution fails to converge.

Given that blocks may be pre-empted, it may be that we don't need the "unsuccessful" status – we could let blocks run forever, and then pre-empt them when they have run for too long. But we suspect that there is more information *inside* the block concerning what is "too long"; it should be the responsibility of the block to terminate itself in certain cases.

A block will report its exit status back to the Scheduler, as well as writing its status to the archive in the block's MeasurementSet.

##### 8.5.4.1.4 FAILED-BLOCK RE-SCHEDULING

It may be that we wish to re-run blocks that have been pre-empted or have been unsuccessful at some later time.

If a block is unsuccessful or is pre-empted, it may request that it be put back in the queue for re-submission.

### 8.5.4.2 Telescope ↔ Archive Interaction

There is a one-to-one correspondence between an observing block and an AIPS++ MeasurementSet: for each observing block executed on the telescope, the telescope will write a MeasurementSet to the archive.

In all cases, a MeasurementSet is written when a block ceases termination, whether it was terminated or ran to completion.

### 8.5.5 Telescope Interaction Example: EVLA

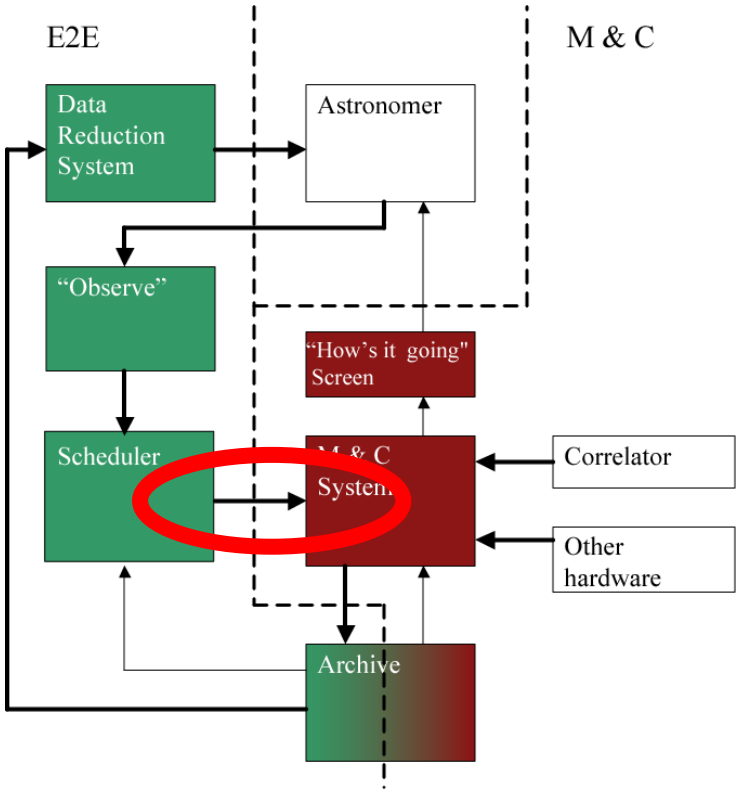



**Figure 16 - EVLA Scheduling Interaction (from B. Clark)**

#### 8.5.5.1 One Scheduler per Sub-array

Barry Clark writes:

The definition of a subarray should be "a collection of antennas that it makes sense to correlate together (but not excluding a single antenna subarray)." (This gets around the

ambiguity in the "group of antennas doing the 'same' thing" definition, about whether, say, the reference antenna for holography is doing the 'same thing' as the others.) I'm in favor of extending this to mean that it makes no sense to correlate antennas not in the same subarray. (Others have argued otherwise, but I haven't heard an argument that I consider justifies the complexity.) In the EVLA, unlike the VLA, we want two layers of subarraying. The top layer is like the current system - the array operator assigns antennas to the subarray, and subarrays run completely independent scripts. The second layer is to make subarraying available to the user (an example would be the multi-subarray sort of thing that Jim Lovell was doing in January and May this year). For this, I think you want all the user's subarrays bundled together in one observing script, with a facility for swapping antennas amongst them. This way the scheduler program need not know about this sort of subarraying. (I presume the observing script interpreter process would fork itself at each subarray creation, but this is an implementation detail.)

### 8.5.6 Design Concepts – Summary

- The Scheduling Toolkit implements Dynamic Scheduling.
- Observations are split into Observing Blocks by the Scheduling Toolkit.
- Observing Blocks do not know when they will be executed on the telescope, (but may express constraints that effectively determine the execution time).
- Observing Blocks may be pre-empted by other blocks.
- Observing Blocks may run for an unspecified period of time.
- The Scheduler maintains a Queue of Blocks.
- The Scheduler waits for a block request from the Telescope, decides which block is the best next block, and gives it to the Telescope for execution.
- The Telescope reports the block execution status back to the Scheduler.
- The Telescope writes a MeasuermentSet to the Archive upon block completion.
- “Astronomy logic” is maintained in the Scheduling Toolkit, rather than in the Telescope.

### 8.5.7 Next Steps

We have enough of a design to generate a detailed description of the Telescope ↔ Scheduler interoperability protocol.

We need to formalize requirements for the Observing Block constraints.

More work needs to be done on the details of the transformation of an Observing Block into executable Telescope commands. This is strictly outside the scope of e2e, but is required for e2e EVLA implementation.

## 8.6 Implementation

## 8.7 References

# 9. REMOTE OBSERVING TOOLKIT

***Author: Tim Cornwell***

***Revision date: 2002/07/12***

***Status: early draft***

## 9.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The Remote Observing Toolkit allows an authorized observer to interact with observations on NRAO telescopes and associated pipelines and archives in real time from any location on the net.

Thus, for example, an astronomer would access a given web page, supply authorization and project information, and thence be allowed access to web-based dynamic information on the progress of a given project or observing session, along with the associated pipeline and archive activities. It is envisaged that this route would be the primary way that external astronomers would interact with observations. However, telescope support staff would almost certainly directly use those capabilities provided by the telescope Monitor and Control systems.

## 9.2 Background

Currently no web-based real-time observing capabilities exist for any NRAO telescope. The 12m telescope had a set of X-windows tools that could be run from any net location to monitor observations. Other telescopes (*e.g.* OVRO) have similar capabilities implemented in various ways (*e.g.* OVRO uses a Java client-server model).

Both the GBT and EVLA have plans to provide real-time access to telescope operations but these do not necessarily include access to information about the pipeline and archive activities.

## 9.3 Scientific requirements

The goal is to provide a single way for astronomers to access visual displays of both the telescope activities and the pipeline and archive system. Thus this toolkit integrates two separate views: that of telescope monitor and control system, and that of the pipeline and archive.

- The telescope monitor and control (M&C) system will, of necessity, be implemented by the telescope builders. Wherever possible, e2e will re-use the existing telescope systems for the display of telescope data (real-time monitor

data, measured data, logging) and for telescope control. To increase the likelihood of successful re-use, e2e will work with telescope builders and encourage the deployment of industry-standard protocols (*e.g.,* CORBA for inter-object communication, IRC for operator-observer chat).

- The provision of displays for the pipeline and archive will be the responsibility of e2e.

It is vital that close cooperation with the M&C groups be maintained to avoid any duplication of effort in these areas.

**Table 21 Remote Observing: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 8.1.1 | 0 | View dynamic summary of observing on any NRAO telescope from any web location |
| 8.1.2 | 0 | Maintain necessary security and privacy |
| 8.1.3 | 0 | Provide integration of telescope-specific remote displays |
| 8.2.1 | 0 | Display current setup of telescope |
| 8.2.2 | 0 | Display recent measured data |
| 8.2.3 | 0 | Display all real-time monitoring devices |
| 8.2.4 | 0 | Display recent calibration results from pipeline |
| 8.2.5 | 0 | Display recent Quick-Look results from pipeline |
| 8.3.1 | 1 | Allow user configuration of displayed items, refresh rates, precisions, *etc.* |
| 8.3.2 | 2 | Allow optional logging of results |
| 8.4.1 | 1 | Allow chat between operator and observer |

## 9.4 System requirements

**Table 22 Remote Observing: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |

| From Package | Derived system requirements |
|---|---|
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | None |
| Remote Observing Toolkit | N/A |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | Archive queryable when Observing |
| Pipeline Toolkit | Results from pipeline available when Observing |
| Pipeline heuristics | None |
| Calibration source toolkit | None |

## 9.5 Design concepts

## 9.6 Implementation

## 9.7 References

# 10. ARCHIVE TOOLKIT

***Authors: John Benson, Tim Cornwell***

***Revision date: 2002/07/11***

***Status: Sufficient for cycle 2***

## 10.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

An archive is a software and hardware system that is:

- A catalog of existing observations, each with associated meta-data, plus pointers to the physical location of the relevant data.
- A place for radio telescopes to store observations.
- A place for a processing pipeline to retrieve from and store to.
- A resource for users to query and from which to request data

The archive toolkit allows interactions with the archive, including submission, querying, distribution requests, *etc.*

An archive query accesses the meta data in the archive catalog. Once the query has returned an answer, the astronomy and/or ancillary data may be requested. Both query and the request may come from a web interface (a button push) or from a program (*e.g.* the telescope monitor and control system, or the pipeline).

## 10.2 Background

For the VLA archive, Barry Clark writes:

> *The current recommendation on use of the archive is that the requestor contact the original observer. Some people do this religiously; others do not. In either case, the request for archive data is sent to me or Miller Goss. I use the Dbase database to see if the request is in compliance with the rules, and if so, forward the request to the analysts, who send the data along by ftp or tape, as they would for a current observer. For observations less than ten years old, in those cases that the requestor does not explicitly state that he has contacted the original observer, I send an E-mail to the last address I have for the original observer. For very large requests (more than about a dozen files), we may decline to provide the analyst time to copy the data for the requestor, and instead ask him to come to Socorro and provide the copying labor himself.*

A large number of organizations have considerable expertise in archiving. For example, STScI and CADC have systems that could be adapted to our use. We are therefore pursuing the possibility of outsourcing the development and installation of the archive. The physical archives will be located at NRAO sites, but the software would be that developed by another organization. The acquisition of the software would be subject to the standard NRAO RFP process.

## 10.3 Scientific requirements

The overall scientific priorities in archiving are (in order of importance):

1. Enabling easier access to the raw data
2. Supporting pipeline processing on exit from archive
3. Providing a more powerful web interface.

The telescope data formats to be supported are:

- VLA: Export format: a specialized format developed specifically for the VLA. This must be augmented by monitor data written separately, and by other data formats (*e.g.* Atmospheric Phase Interferometer).
- VLBA: FITS-IDI standard.
- GBT FITS binary table format. All data are written in this format.
- EVLA: TBD
- Generic: MeasurementSet V2

**Table 23 Archive: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 10.1.1 | 0 | Stage data in various formats - FITS binary tables, FITS images, VLA export, ASCII, calibration tables, *etc.* |
| 10.1.2 | 0 | Ingest from tapes, data-streams, ftp servers, *etc.* |
| 10.1.3 | 1 | Archive all ancillary data in original format |
| 10.2.1 | 0 | Extract specified meta-data |
| 10.3.1 | 0 | Add ingested original data to a catalog of all projects. |
| 10.3.2 | 0 | Associate meta-data with project in catalog. |
| 10.3.3 | 2 | Database interface should be generic to allow migration to different engine. |

| Ident. | Pri. | Description |
|---|---|---|
| 10.4.1 | 0 | Accept distribution requests from web or other interfaces |
| 10.4.2 | 0 | Enforce proprietary access rights to projects. Metadata is open. |
| 10.4.3.1 | 0 | Make requested data available via web |
| 10.4.3.2 | 0 | Make requested data available via web within 2 hours |
| 10.4.3.3 | 1 | Make requested data available via web within 30 minutes |
| 10.4.4.1 | 0 | Write requested data to a physical medium for delivery |
| 10.4.4.2 | 0 | Write requested data to a physical medium within 24 hours |
| 10.4.4.3 | 1 | Write requested data to a physical medium within 6 hours |
| 10.4.5 | 1 | Support pipeline processing on exit from archive |
| 10.5.1 | 0 | Back up catalog and all data to chosen storage medium |
| 10.5.2 | 0 | Allow migration of storage media |
| 10.5.3 | 1 | Allow straightforward reconstitution of archive from backup |
| 10.6.1 | 0 | Provide access to the catalog from a web-based interface |
| 10.6.2 | 1 | Support queries and sub-queries from web-based interface |
| 10.6.3 | 1 | Search by astronomical name, translated using with users choice of standard query machines |
| 10.6.4 | 1 | Provide compact previews of images and other data |
| 10.7.1 | 0 | Provide API equivalent to major functionality of web interface |
| 10.7.2 | 0 | Allow API-based project or subset retrieval |
| 10.7.3 | 0 | Allow API-based queries and sub-queries |
| 10.8.1 | 0 | Allow operators to manage various distribution queues, including query, reorder, resubmit, delete, *etc.* |
| 10.8.2 | 1 | Allow and provide tools for migration of storage media |
| 10.8.3 | 1 | Allow mirroring to one or more other locations |
| 10.8.4 | 2 | Keep physical copies at two or more geographically distinct locations |

## 10.4 System requirements

**Table 24 Archive: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | Access to calibrator information |
| Remote Observing Toolkit | Access speed adequate for Observing |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | N/A |
| Pipeline Toolkit | Access and store data |
| Pipeline heuristics | None |
| Calibration source toolkit | Access calibrator data |

## 10.5 Design concepts

The design has the following elements:

- Conversion of raw data into AIPS++ MeasurementSets
- Extraction of meta-data from AIPS++ MeasurementSets
- Storage of meta-data in catalogs
- Querying of meta-data from within AIPS++ and from web pages
- On-request delivery of raw data files

## 10.6 Implementation

AIPS++ stores data in a specialized Table system. The data format is called the AIPS++ MeasurementSet. A formal description is given in AIPS++ Note 229. An equivalent of the MS in FITS binary tables is used in AIPS++ as an informal archive format.

Conversion programs for the telescope data formats are available in AIPS++ for VLA, VLBA, and GBT. A substantial problem is that there exist other sources of data, such as monitor data or information from specialized instruments such as the VLA Atmospheric Phase Interferometer.

A project may consist of a collection of related files, including original data, derived MeasurementSets, derived calibration tables, and derived images. Additions to a project may be made at any time.

One of the main clients of the archive will be a calibration and imaging pipeline. The pipeline will read project data from the archive (convert to MeasurementSet format if necessary), process according to various heuristics, and insert the resulting calibration information, images, and processing logs back into the archive.

### 10.6.1 Implementation Completed – Development Cycle I

The first E2E development cycle ends on 15 July 2002. In this section, we describe what parts of the archive system have been constructed and the current status of the archiving operation.

The major goals set for the Archive System during Development Cycle I are:

- Acquire a high capacity disk array and begin copying VLA archive tapes onto disk,
- Design and build a catalog table schema that describes the contents of the archive,
- Support AIPS++ pipeline access to the catalog tables,
- Build a web-tool interface that allows users to browse the catalog tables by submitting queries,
- Support downloading data files selected by the web-based query tools.
- Respond to cone-search URL queries submitted by NVO users, reply with XML VO tables.

The current status of the archive system meets the Cycle I objectives.

#### 10.6.1.1 New Archive System Hardware

We have purchased and installed a fiber channel disk array (SAN-Storage Area Network) that is attached to the local NRAO network. The disk array currently has a 2 Tbyte capacity, and will be expanded to 4 Tbytes in the near future. The SAN is located in the server room in the AOC building.

A four processor IBM Model 370 (really) provides communication to the disk array and serves as the host for the AIPS++ pipeline processes.

#### 10.6.1.2 Archive Data Being Loaded

The NRAO-AOC computer division has begun copying VLA archive tapes onto the disk array. All newly observed data on archive tapes are copied to the disk as they arrive in Socorro from the VLA site. In addition, archive tapes are being retrieved from the tape library and copied to disk in reverse time order from the current date. We estimate that

nearly 1200 ModComp archive files will be loaded by end of cycle 1. We intend to hire a student who will copy VLA tapes to the disk at the greatest rate possible.

In the same manner, newly written VLBA archive tapes and tapes from the VLBA library are being copied onto the SAN disk array. This work is being done by the VLBA Correlator operators as time permits.

Three NRAO surveys are currently on the archive disk array: NVSS, FIRST and the VLBA Calibrator Survey. Images from these surveys are accessible through the network connection.

#### 10.6.1.3 The Archive Catalog Tables

The various forms of data loaded in the archive are described in a series of catalog tables. Users send queries to the catalog tables in order to browse the contents of the archive and to select data for downloading directly. The catalog tables provide a more detailed description of the archive than those of previous NRAO archives.

The catalog system contains meta-data describing all raw data files from the VLA and VLBA, and is configured to catalog raw data from the GBT. Image files produced by the AIPS++ pipeline and survey results are currently cataloged. We also have the capability to catalog the calibrated measurement sets that will be produced by the AIPS++ pipeline.

The design of the catalog tables for astronomy data is basically complete and the table schema is implemented. Catalogs of ancillary and monitor data do not currently exist.

We have chosen to write the catalog tables as AIPS++/Glish tables rather than use a commercial database system. The Glish table environment has a rich tool-set that allows us to build and load tables easily, and data can be simply retrieved by means of a query language quite similar to SQL. A major advantage to using AIPS++/Glish tables is that the AIPS++ pipeline will have a direct connection to the catalog. This is important enough that it is worth a serious effort on out part to support the archive catalog in Glish tables. In addition, the archive system will then be almost totally based in AIPS++ and thus be applicable to any observatory running AIPS++ (a goal supported by our NSF/ITR grant). Thus far the Glish tables have worked fine.

##### 10.6.1.3.1 Filling the Catalog Tables

Since the E2E Archive will contain data from many different sources and in different formats, providing software to retrieve meta-data from many file types would be a major undertaking. Instead we have chosen to convert the archived data files to AIPS++ measurement sets and AIPS++ images. Fillers already exist in AIPS++ for loading VLA Export files, VLBA FITS files, GBT FITS files and AIPS FITS image files. Thus the meta-data is mined from standard AIPS++ file types using Glish scripts.

The Glish scripts (really programs) that retrieve meta-data from measurement sets and AIPS++ image files exist and are currently in use building the catalog tables as archive files are copied onto the disk array. These programs are now in a fairly mature state and

operate in a reasonably stable manner. The loading of the catalog tables is pretty much keeping up with rate at which new archive files are copied to the disk array.

#### 10.6.1.3.2 CATALOG QUERIES IN THE GLISH ENVIRONMENT

Queries directed to the catalog tables come from a number of outside (non-Glish) sources, such as the archive web-based search tool and URL queries coming in from National Virtual Observatory users. There are two basic components that support queries: 1. a socket-pipe that connects to any provided CGI program and 2. a Glish server program that receives and processes the piped queries. Currently responses are formatted in html streams and piped back to the CGI program. The pipe process (referred to as the 'chromepipe' on its better days) was written by Boyd Waters, the Glish query server (e2equery.g) by John Benson.

### 10.6.1.4 The Archive System User Interface

In order to meet the requirements of the Cycle I Development Cycle we have constructed a web-based user interface that consists of relatively simple web-pages which allow users to query the catalog system by filling out forms. The web-pages are written in HTML for ease of modification and experimentation.

The home page is http://bernoulli.aoc.nrao.edu/E2E/catalogs. At this time there are four separate query form pages that return different kinds of information. In response to a query, one may receive a tabular list of observing projects, one project per row, or a tabular list of observing scans (similar to the current VLA DB query tool), or a tabular list of the raw data archive files, or a list of image files in the loaded surveys or reference images created by the AIPS++ pipeline.

One of the mains goals of the E2E Archive System is to allow authorized users and NRAO support people to directly download archived data. For Cycle I we only allow the system to copy files to a local disk that is accessible to the local ftp-server. Users will ftp their data manually. At this time, the downloading feature is restricted to the NRAO data analysts and a few other local staff members. There is currently no user authentication procedure, so proprietary data is protected by denying download privileges. Early on in cycle 2, we will build a user login/account system that will allow people to retrieve their own data.

# 11. PIPELINE TOOLKIT

***Author: Tim Cornwell***

***Revision date: 2002/07/12***

***Status: early draft***

## 11.1 Summary

The Pipeline Toolkit allows submission, monitoring, and interaction with a pipeline reduction system in AIPS++.

A pipeline may run in a variety of contexts:

- In a user's desktop machine,
- In a user's personal network,
- In a dedicated telescope-based network,
- In a Grid.

The pipeline will run the same software, AIPS++, as that used for interactive data reduction. The pipeline is split conceptually into *mechanism* and *heuristics*. This task describes just the mechanism. The definition of heuristics is split into a separate task.

## 11.2 Background

Pipeline processing has been one of the key requirements driving the design of AIPS++. The tool-based approach and scripting capabilities provide a suitable environment in which to develop pipelines. Two pipeline systems have been completed using AIPS++:

- The Parkes Multibeam Project (http://www.atnf.csiro.au/research/multibeam/) used tools written in AIPS++ to pipeline process data from a 13-beam focal plane array receiver on the Parkes 64m telescope. This resulted in the very productive HI Parkes All-Sky Survey (http://www.atnf.csiro.au/research/multibeam/release).
- The Herzberg Institute of Astronomy Auto-Correlation Spectrometer and Imaging System (http://www.drao.nrc.ca/science/jcmt_correlator/acsis.html) project has developed a Beowulf-type system to edit, calibrate, and image data from a focal plane array in real time. Again, tools written in AIPS++ and Glish were used for this development.

These pipelines were designed for specific types of processing, and are therefore limited in flexibility. One missing ingredient is the use of meta-data describing the contents of data sets and the relationship between different data sets. A prototype *meta-data driven* radio astronomical pipeline based on AIPS++ has been under development at NCSA for

the last year. This pipeline has been designed to connect to the BIMA millimeter wavelength synthesis radio telescope, and to provide pipelined processing for that telescope.

Another missing part is a well-defined executive. Pipelines are generally decomposable into three parts:

- A resource manager used to find and use resources
- An executive controlling the execution of parts of the pipeline
- The commands to be processed in the pipeline.

Experience with other pipelines has shown the value of work dedicated to the executive. Although it is possible to work with monolithic end-to-end scripts for processing from raw data to final image, there are numerous advantages to splitting scripts into separate steps (*e.g.* data conversion, data editing, initial calibration, editing refinements, initial imaging, first self-calibration, *etc.*) The executive should therefore be able to execute a sequence of steps, and keep track of the conclusion of each, and so on. The STScI OPUS system has such capabilities, and is based upon many years experience of running pipelines for HST and other missions.

A major part of the scientific work will consist in the development of recipes for specific types of processing.

## 11.3 Scientific requirements

*Requirements are to be inserted into a table. A unique number must identify all items. If an item is deleted after the draft, it should be struck out rather than removed from the list. The priority must be chosen from: 0: essential, 1: desirable, 2: if possible. The description should be concise.*

**Table 25 Pipeline: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 11.1.1 | 0 | Process observations from an archive to a calibrated reference image that is subsequently stored back in the archive |
| 11.1.2 | 0 | Provide standard log of processing, including diagnostic plots and statistics such as comparisons to previous results |
| 11.1.3 | 0 | Be installation-flexible: can be installed on non-specialized hardware by end user |
| 11.2.1 | 1 | Provide real-time feedback via standard compact displays and plots |
| 11.2.2 | 1 | Be operable automatically or manually |

| Ident. | Pri. | Description |
|---|---|---|
| 11.2.3 | 1 | Allow preemption, termination, resubmission, etc. |

## 11.4 System requirements

**Table 26 Pipeline : System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | None |
| Remote Observing Toolkit | See processing queue in real-time |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |
| Pipeline Toolkit | N/A |
| Pipeline heuristics | Pipeline must execute heuristics |
| Calibration source toolkit | None |

## 11.5 Design concepts

Processing will be using AIPS++ tools such as imager and calibrater. The pipeline scripts will be expressed in Glish. The high level control will be in Glish. The pipeline will retrieve data from the archive catalogs using specialized Glish functions.

Here we give a brief overview of a possible design.

- Meta-data for a given observation is obtained directly from a query of the archive using Table Query Language constructs.
- Processing heuristics are captured in Glish scripts. The Glish scripts are assembled from Glish fragments using a make system to capture dependency information.
- The selection of the relevant makefile is dependent upon meta-data about the observation.

- The processing within the Glish script is determined from meta-data, from other information within the data set, and from a standards master (for conventions as to image cell size, for example).
- The Glish script is passed to an executive for processing. The prototype executive is quite simple.
- Results from the pipeline processing are published to a web page.

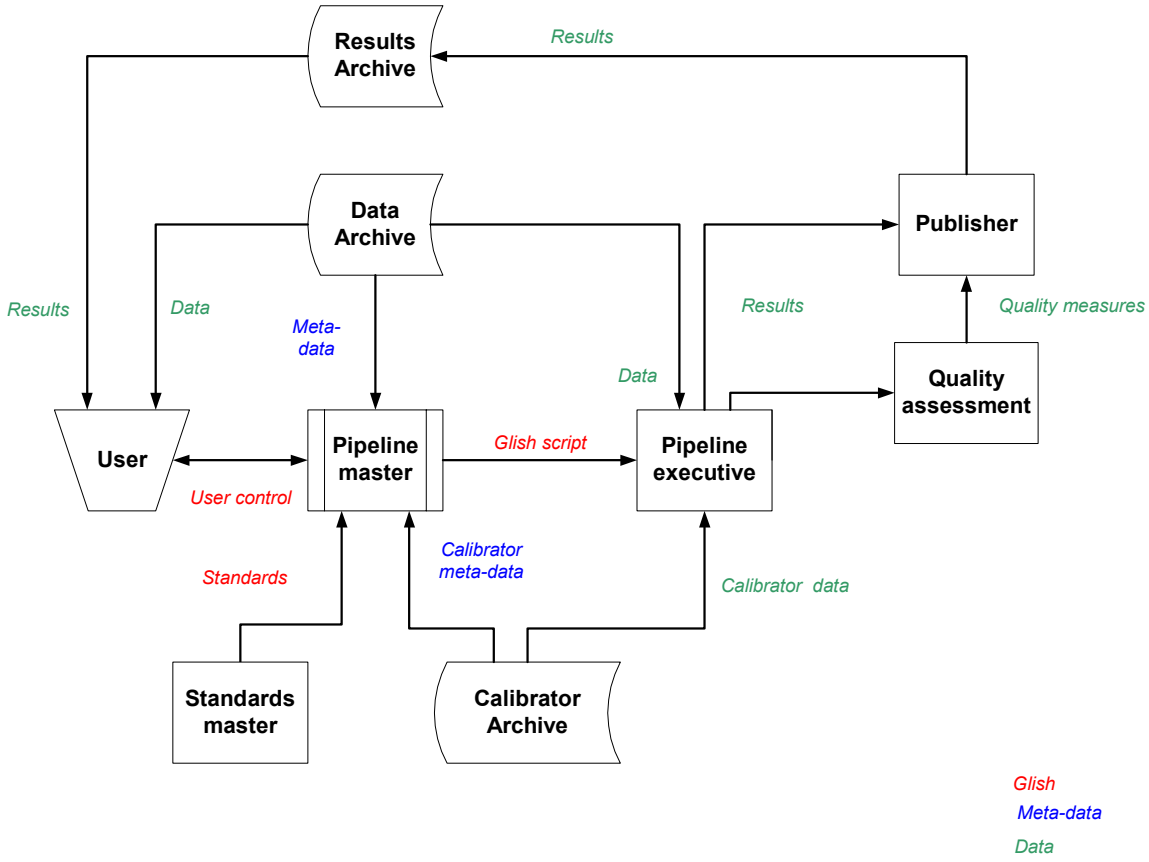



Figure 17 Outline of design of prototype pipeline

## 11.6 Implementation

### 11.6.1 Implementation

The implementation in cycle 1 is of the prototype design described above based entirely in AIPS++. Details are available in **e2e memo #5**. The work described here was

conducted between February and June 2002, consuming about 6-8 FTE-weeks of Tim Cornwell's time. The prototyping work was concluded at the end of the e2e development cycle. As of July 3, 2002, the status of the prototype was:

- All parts of the design diagram were filled out *except* for the calibrator archive access *and* the quality assessment.
- About 1,000 lines of new Glish code were needed. The focus has been entirely on VLA pipeline reduction.
- Little effort has gone into developing heuristics for data reduction. Instead, the focus has been on the structure of the pipeline design.
- The pipeline could process a simple continuum project, AB973, to through to HTML pages.
- Most of the processing of a spectral line project, AM649, could be processed but some peculiarities in the observation setup require a more sophisticated approach to determining and annotating observer intention.
- Not all data products have been produced. The MS as binary FITS cannot be written by the current archive2ms tool. Source and calibrator catalogs are on hold pending a decision about formats. There has been no attempt to publish calibration tables.

Our assessment is:

- The basic design is adequate for current needs. It could be re-factored but there is no urgent need to do so.
- The use of makefiles to contain rules looks promising. A complicated hierarchy of makefiles has yet to be assembled. The make system is stable and needs little extra work.
- The weak point is in the whole area of observer intention. Much more work is needed here, first in recording intention, and second in guessing it from the existing MeasurementSet and OBSERVE script. The concept of threads help is disentangling multiple activities within one observation but the threads themselves need to be more capable.

Our recommendation is that in the next development cycle we should continue with this prototyping work, initially concentrating on the question of observer intention. In parallel, some work on heuristics for simple data reduction cases should start.

## 11.7 References

# 12. PIPELINE HEURISTICS

***Author: Tim Cornwell***

***Revision date: 2002/07/12***

***Status: early draft***

## 12.1 Summary

The Pipeline heuristics are rules to help in the processing of observations on the pipeline. They are driven by the data itself, pipeline processing parameters, and meta-data.

## 12.2 Background

## 12.3 Scientific requirements

*Requirements are to be inserted into a table. A unique number must identify all items. If an item is deleted after the draft, it should be struck out rather than removed from the list. The priority must be chosen from: 0: essential, 1: desirable, 2: if possible. The description should be concise.*

**Table 27 Pipeline Heuristics: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 12.1.1 | 0 | Execute in standard pipeline |
| 12.1.2 | 1 | Use standard AIPS++ facilities only |
| 12.1.3 | 1 | Support processing of most major observational modes on NRAO telescopes |
| 12.2.1 | 1 | Calibrate, edit, and image observations without manual intervention |
| 12.2.2 | 0 | Provide diagnostics in the form of standard log, plots, and displays |
| 12.2.3 | 0 | Assign a separate quality measure to each of derived calibration, editing commands, and reference image |
| 12.2.4 | 0 | Exactly repeatable in all situations |
| 12.2.5 | 0 | Decomposed into separate phases each with well-defined input and output requirements. |
| 12.3.1 | 0 | Subject to change control by relevant change control board |
| 12.3.2 | 1 | Documented in standard format |

| Ident. | Pri. | Description |
|---|---|---|
| 12.3.3 | 0 | Accompanied by test script and data for regression testing |

## 12.4 System requirements

**Table 28 Pipeline Heuristics: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | None |
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |
| Pipeline Toolkit | Scripts must be in Glish/AIPS++ |
| Pipeline heuristics | N/A |
| Calibration source toolkit | None |

## 12.5 Design concepts

Processing is expressed in terms of Glish scripts calling AIPS++ functions. A script is to be assembled from script fragments (*e.g.* fill, edit, calibrate, image, *etc.*).

## 12.6 Implementation

### 12.6.1 Implementation in Cycle 1

The implementation in cycle 1 is of a prototype based entirely in AIPS++. Details are available in e2e memo #5. Here we provide a brief overview of the expression of heuristics.

Processing scripts are assembled from makefiles containing fragments of Glish code. For clarity, we show here the hierarchy of makefiles invoked for processing of a VLA continuum observation.

## The root makefile root.gm:

```
#
# Root make file
#
PIPESTATE      = pipelinestate
VERSION        += root.0.01
#
# This target must always be invoked
#
root:
        glish global environ;
        glish global e2edir := environ.E2EROOT;
        glish shell('mkdir -p pipelinestate');
        glish include "logger.g";
        glish if(is_record(dl)) dl.screen();
        glish global system;
        glish system.output.pager := F;
        glish include "servers.g";
        glish whenever defaultservers.alerter()->["fail error"] do {
        glish   print "Script exited because a server failed ", \$$value.value;
        glish   exit(1);
        glish };
        glish include 'sysinfo.g';
        glish for (f in field_names(sysinfo())) note(f, ":", sysinfo()[f]());
        glish note('Pipeline makefile versions ${VERSION}');
#
# Simplest make rule
#
all: root
```

## Makefile for filling data: fill.gm

```
#
# Make file for a pipeline to process VLA data: filling only
#
PIPESTATE      = pipelinestate
VERSION        += fill.0.01
#
```

```
# Assumes definition of variables by e2epipeline
#
# archive   : location of archive
# project   : project name
# archfiles : archive files to load
# tbeg      : beginning time (can be unset)
# tend      : end time (can be unset)
# msname    : MeasurementSet name
#
# Fill the MS from the data tapes
#
${PIPESTATE}/fill:
        glish include "vlafiller.g";
        glish result := T;
        glish v:=vlafiller();
        glish for (tape in archfiles) {
        glish   filename := tape ~ s!_!/file_!g ~ s!//!/!g;
        glish   note('Filling ', msname, ' from tape ', filename);
        glish   result := result && v.diskinput(filename);
        glish   result := result && v.output(msname);
        glish   result := result && v.selectproject(project);
        glish  result := result && v.selecttime(start=tbeg, stop=tend);
        glish   result := result && v.fill();
        glish }
        glish v.done();
        glish include 'table.g';
        glish if(!tableexists(msname)) {return throw('Failed to create MS ', msname)};
        glish t:=table(msname);
        glish if(is_fail(t)) fail;
        glish if(t.nrows()==0) {return throw('MS ', msname, ' has no rows')};
        glish t.done();
        glish include "ms.g";
        glish m:=ms(msname);
        glish if(is_fail(m)) fail;
        glish m.summary(verbose=T);
        glish m.done();
        glish if(result) shell('touch $@');
#
# Threads
#
threads.g: ${PIPESTATE}/fill
```

```
        glish include "e2emsutils.g";
        glish msutils := e2emsutils(msname);
        glish msutils.summary();
        glish msutils.writethreads();
        glish msutils.done();
#
#
# This target definition overrides those in previous makefiles
#
all:    root ${PIPESTATE}/fill threads.g
#
# Root file
#
.include "root.gm"
```

## Makefile for processing VLA continuum data: continuum.gm

```
#
# Make file for a pipeline to process continuum data
#
VERSION        += continuum.0.01
PIPESTATE      = pipelinestate
#
threads: threads.g
        glish include "threads.g";
        glish if(is_fail(threads)) fail;
#
# First edit of data: flag autocorrelations, and quack.
#
${PIPESTATE}/firstedit: ${PIPESTATE}/fill
        glish include 'autoflag.g';
        glish result := T;
        glish af := autoflag(msname);
        glish if(is_fail(af)) fail;
        glish result := result && af.setselect(quack=[30,10], autocorr=T);
        glish result := result && af.run(plotscr=F);
        glish af.done();
        glish shell(spaste('mv flagreport.ps ', archive, '/firstedit.ps'));
        glish if(result) shell('touch $@');
#
```

```
# Initialize fluxes for calibrators only
#
${PIPESTATE}/setjy: ${PIPESTATE}/firstedit
        glish include "imager.g";
        glish result := T;
        glish img:=imager(msname);
        glish if(is_fail(img)) fail;
        glish for (thread in threads) {
        glish   if(thread.valid()) {
        glish     result := result && img.setdata(msselect=thread.query());
        glish     for (fieldid in thread.fields().Gcal) {
        glish       for (spwid in thread.spwid()) {
        glish         result := result && img.setjy(fieldid=fieldid,
        glish           spwid=spwid);
        glish       }
        glish     }
        glish   }
        glish }
        glish img.done();
        glish if(result) shell('touch $@')
#
# Solve for G terms
#
${PIPESTATE}/Gcal: ${PIPESTATE}/setjy
        glish include "calibrater.g";
        glish result := T;
        glish cal:=calibrater(msname);
        glish if(is_fail(cal)) fail;
        glish for (thread in threads) {
        glish   if(thread.valid()) {
        glish     fields:=thread.fields().Gcal;
        glish     if(is_fail(fields)) fail;
        glish     msselect:=spaste('(FIELD_ID in ', as_evalstr(fields), ')');
        glish          if(thread.query()!='')  msselect:=spaste(thread.query(),  '  &&  ',
msselect);
        glish     result := result && cal.setdata(msselect=msselect);
        glish     Gcaltable := thread.caltable('G');
        glish     result := result && cal.setsolve('G', table=Gcaltable, t=600);
        glish     result := result && cal.solve();
        glish    thread.addhistory('Calibrated   G   Jones   (antenna-IF   gain)   on   ',
thread.sources().Gcal);
```

```
        glish   }
        glish }
        glish cal.done();
        glish if(result) shell('touch $@')
#
# Get calibrator fluxes
#
${PIPESTATE}/scaledGcal: ${PIPESTATE}/Gcal
        glish include "calibrater.g";
        glish result := T;
        glish cal:=calibrater(msname);
        glish if(is_fail(cal)) fail;
        glish for (thread in threads) {
        glish   if(thread.valid()) {
        glish     Gcaltable := thread.caltable('G');
        glish     scaledGcaltable := thread.caltable('scaledG');
        glish                result   :=   result   &&   cal.fluxscale(tablein=Gcaltable,
tableout=scaledGcaltable,                              reference=thread.sources().Fluxcal,
transfer=thread.sources().Gcal);
        glish    thread.addhistory('Flux       scale       determined       from       ',
thread.sources().Fluxcal);
        glish   }
        glish }
        glish cal.done();
        glish if(result) shell('touch $@')
#
# Apply scaled calibration table
#
${PIPESTATE}/Gcalibrate:      ${PIPESTATE}/scaledGcal
        glish include "calibrater.g";
        glish result := T;
        glish cal:=calibrater(msname);
        glish if(is_fail(cal)) fail;
        glish for (thread in threads) {
        glish   if(thread.valid()) {
        glish     fields:=thread.fields().all;
        glish     msselect:=spaste('(FIELD_ID in ', as_evalstr(fields), ')');
        glish          if(thread.query()!='')  msselect:=spaste(thread.query(),  '  &&  ',
msselect);
        glish     scaledGcaltable := thread.caltable('scaledG');
        glish     result := result && cal.setdata(msselect=msselect);
        glish     result := result && cal.setapply('G', table=scaledGcaltable);
```

```
        glish     result := result && cal.correct();
        glish   }
        glish }
        glish cal.done();
        glish if(!is_fail(result)) shell('touch $@')
#
# Second edit of data: thread-independent for the moment
#
${PIPESTATE}/secondedit: ${PIPESTATE}/Gcalibrate
        glish include 'autoflag.g';
        glish result := T;
        glish af := autoflag(msname);
        glish if(is_fail(af)) fail;
        glish result := result && af.setuvbin(thr=0.01, econoplot=T, column="CORR");
        glish result := result && af.run(plotscr=F);
        glish af.done();
        glish shell(spaste('mv flagreport.ps ', archive, '/secondedit.ps'));
        glish if(result) shell('touch $@')
#
# Make cleaned images
#
${PIPESTATE}/image: ${PIPESTATE}/secondedit
        glish include "e2eimagingutils.g";
        glish result := T;
        glish for (thread in threads) {
        glish   result := result && e2eimagingutils().image(thread);
        glish   result := result && e2eimagingutils().selfcal(thread);
        glish }
        glish if(result) shell('touch $@')
#
# Publish the answers
#
${PIPESTATE}/publish: ${PIPESTATE}/image
        glish include "threads.g";
        glish include 'e2epublish.g';
        glish publish := e2epublish(project, '${VERSION}');
        glish result := T;
        glish note('Publishing results to web pages');
        glish result := result && publish.publishproject(threads);
        glish publish.done();
        glish if(result) shell('touch $@')
```

```
#
# Print log
#
log:
        glish include 'logger.g';
        glish dl.printtofile(filename=spaste(archive, '/', project, '.log'));
#
# This target definition overrides those in previous makefiles
#
all:    root threads.g threads ${PIPESTATE}/publish log
#
# Include makefile for filling
#
.include "fill.gm"
```

Different strands of intention within the observation are identified by a thread. A thread knows how to select the data relevant for it only. It also knows which sources are targets, G (antenna gain) and/or B (bandpass) and/or T (atmospheric phase) calibrators, flux calibrators, *etc.* The functions called for each thread (*e.g.* e2eimagingutils().image()) know how to process the data for that data, either from looking at the data in detail or from a standards master that knows, for example, to use wide-field imaging for B array observations at 5GHz.

This thread-based approach works well. Deficiencies come from the limited expressiveness allow in the current implementation of threads and the difficulty of decomposing an observation into threads. Both of these can be addressed in future prototypes without discarding the basic concept of threads of intention.

## 12.7 References

# 13. CALIBRATION SOURCE TOOLKIT

***Author: Greg Taylor, Tim Cornwell***

***Revision date: 2002/07/12***

***Status: good version, sufficient to be implemented as prototype***

## 13.1 Summary

*Include here a summary what the component does (and doesn't do). Include an example or two of the use of the component.*

The Calibration source Toolkit handles information about calibration sources for all NRAO telescopes. The primary goal is to consolidate all information on calibration sources, flux and position reference systems, in one logical location, from which other catalogs may be derived as necessary.

## 13.2 Background

The following description is from a memo written by Greg Taylor and others (2001/04/27). Some changes have been made to reflect the overall e2e approach of integrated tools.

A fundamental need of VLA and VLBA users is to search out the best calibration sources for a given observing program. At the present date there are two major databases (with about 800 sources in common) and a large number of tools in various states of disrepair. No common interface is presented to the user, and even worse is the fact that most of the tools are disjoint. For example there is a tool to search the VLA calibration source manual, and another tool to look at the flux history. Even after manually passing information from one to another the user frequently needs to consult the VLA calibration source manual to find out positional accuracies and to look at images and visibility plots. The situation for VLBA calibration is perhaps worse. Information about calibration sources is even more widely dispersed and the search engine is incomplete, inadequate and irreparable (the code has been lost).

To remedy the current situation we would like to establish a single database for both VLA and VLBA calibration sources, and a suite of tools to access it. Below we attempt to describe the desirable specifications for such a system from a users point of view.

Conceptually, we'll consider three steps for the process: (1) Search; (2) Evaluate; (3) Include. Lastly we provide a few guiding remarks about (4) Infrastructure.

### 13.2.1 Search

In this first stage the user has in mind a *specific telescope* and *frequency* and a *target RA, Dec* or *source name* and desires to find suitable calibration sources in the vicinity. Generally the angular separation between calibration source and target is an important consideration that should be available both graphically and numerically. The current tools in use for the VLA and VLBA are both clumsy. A much better interface has been developed by James Morgan for BIMA (see http://bima2.astro.uiuc.edu/java/Calfind/ ). This interface allows the user to input:

- Field Size in degrees
- Planet Epoch + update button
- Center RA, DEC + update button

What it provides is a RA vs Dec skyplot with the target location and that of calibration sources. The sun and planet are also plotted for the planet epoch specified. The calibration sources are color and size coded for brightness. A simple uptime display is also shown in a panel to the right of the skyplot. If a calibration source is "clicked on" then it returns one line of information about it in a text window at the bottom, including the most recent flux density measurement, and the source-calibration source angular separation.

*This is an excellent illustration of the functionality we would like.* [1]

### 13.2.2 Evaluation

The BIMA Java applet provides some information about the calibration sources, but not as much as VLA and VLBA users need to make an informed decision. There are still a number of features we would like.

A) A frequency, and/or flux density, and/or angular size filter such that only calibration sources meeting certain criteria are shown. This pre-selection might be necessary to avoid users being swamped with a large number of nearby but faint/resolved choices. The best way to implement this would be an interface on the main plotting widow allowing the user to select such items as: (1) desired coordinate frame (J2000 or B1950); (2) Frequencies of interest (perhaps by using buttons to select bands); (3) the instrument of interest (VLA/VLBA/ALMA—each of which could select what

[1] Greg Taylor writes "I sent e-mail to James Morgan on March 21st asking him if he would be willing to provide us with his JAVA applet and he has delivered it to me."

information is returned about each calibration source like the default visibility plot, and would add some features like latitude lines on the plot for the selected instrument; and (4) a flux density cutoff.

B) Links to images and visibility plots. Ideally these would appear in a compact one-line form when one clicked on a calibration source.

C) A link to a light curve from the VLA flux history database (http://aips2.nrao.edu/vla/calflux.html).

D) Link to a spectrum (total flux vs. frequency) using most current measurements but also with historical ranges plotted.

E) Additional textual information about the calibration source like positional accuracy, (u,v) restrictions. This may require a couple lines like:

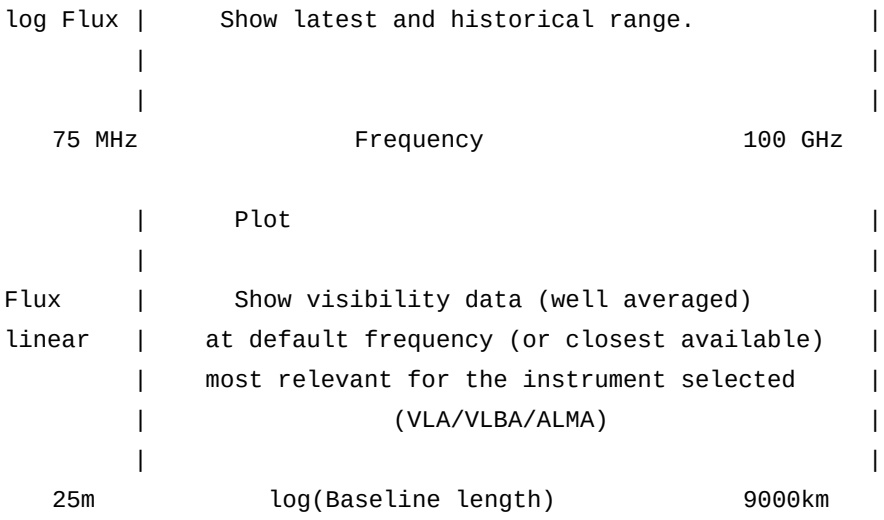
```
0319+415    code=A  dist=2.23 deg  aliases  J0319+4130  0316+413  3C84  J0319+41
              J2000           Error     gal lat/long    Date (4 Mar1997)
   RA       03 19 48.160500    0.0054    -13.26115       03 19 49.123456
   Dec      41 30 42.10300     0.00032   150.57583       41 30 04.90514
       Position from JVAS - Patnaik et al. 1992, mnras, 254, 655.
       Comment:  Good VLA calibration source, heavily resolved with VLBI.
       Polaration Calibration source (link to polarization monitoring data)
```

| BAND | VLBA | A+ | A | B | C | D | S_T(Jy) | S_short | S_long | Date | MIN(kL) | MAX(kL) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 90cm | P | X | S | S | X | X | 8.0 | 3.0 | 0.8 | 1996Feb01 | 13 | |
| 50cm | | X | S | - | - | - | 15.0 | 5.0 | 1.2 | 1996Feb01 | 13 | |
| 20cm | L | X | P | P | P | X | 23.9 | 6.0 | 1.3 | 1996Feb01 | 13 | |
| 13cm | | X | P | - | - | - | 23.5 | 7.0 | 1.4 | 1996Feb01 | | |
| 6cm | C | X | P | P | P | P | 23.3 | 8.0 | 1.5 | 1996Feb01 | | |
| 3.7cm | X | X | P | P | P | P | 21.7 | 9.0 | 1.6 | 1997Mar12 | | |
| 2cm | U | X | P | P | P | P | 20.7 | 10.0 | 1.7 | 1996Feb01 | | |
| 1.3cm | K | X | P | X | S | S | 16.4 | 12.0 | 1.8 | 1996Feb01 | | |
| 7mm | | X | P | X | X | S | 13.0 | 10.0 | 1.9 | 1996Feb01 | | |
| 3mm | | X | P | - | - | - | 7.0 | 5.0 | 1.2 | 1996Feb01 | | |
| 1mm | | X | | | | | ?? | | | | | |

The band codes indicate default continuum frequencies commonly used. If other, intermediate frequencies become commonly used by the EVLA then they could warrant additional lines.

Default plots obtained by clicking on a calibration source:

```
|     Plot                                                    |
|                                                             |
```




Click on quality code for links to visibility curve, image, flux history, spectra, and other information. Clicking on one of the links should pop-up a postscript or gif image in a separate window. This textual information (and links) could appear in a special box that updates each time a calibration source is selected in the main RA-DEC plot window. Some of the links could also appear in a drop-down style menu so as to provide the maximum amount of information with the minimum amount of clutter.

### 13.2.3 Include

The tools described here should be tightly integrated with the observation scripting toolkit.

### 13.2.4 Infrastructure

Most of this document up to this point is comprised of discussion about a suite of web-based tools to access the unified calibration source database. In this section some demands on the database itself are discussed.

A single calibration source database should be established. This would help to keep the VLA and VLBA in the same reference frame. It would also reduce our overhead costs of maintaining separate databases and tools. Authorized people could check in additions and corrections to the database. Epochs should be attached to all additions and corrections to provide a history for those that need it. Only J2000 positions would actually be stored and other coordinates would be derived on the fly.

Although most people will query the database through the web-based tools described in section 2, it may be that other programs may want to dip into the database and retrieve information. JObserve and Sched, to name the two most popular programs for creating VLA and VLBA schedules, may want to build in some functions to either interact with the web-based tools, or to retrieve information from the database directly. Other

programs like cjobgen will also make use of the database. The ALMA observing tool may require the ability to query the calibration database (through some TaQL string or special function) by the observation description software. Or people may want to write their own special-purpose software that uses the calibration source database.

## 13.3 Scientific requirements

*Requirements are to be inserted into a table. A unique number must identify all items. If an item is deleted after the draft, it should be struck out rather than removed from the list. The priority must be chosen from: 0: essential, 1: desirable, 2: if possible. The description should be concise.*

**Table 29 Calibration Source Toolkit: Scientific Requirements**

| Ident. | Pri. | Description |
|---|---|---|
| 13.1.1 | 0 | Store time-tagged position, flux, polarization, structural information for calibration source sources |
| 13.1.2 | 0 | Integrated database for VLA/VLBA |
| 13.1.3 | 0 | Database for GBT |
| 13.1.4 | 1 | Single database for all NRAO telescopes |
| 13.2.1 | 0 | Tools for maintaining, updating, revising databases |
| 13.3.1 | 0 | Web-based tools for accessing calibrator information |
| 13.3.2 | 0 | Web-based tools for finding calibrator candidates |
| 13.4.1 | 2 | API for accessing database |

## 13.4 System requirements

**Table 30 Calibration Source Toolkit: System Requirements**

| From Package | Derived system requirements |
|---|---|
| Proposal Submission Toolkit | None |
| Proposal Management Toolkit | None |
| Telescope Simulation Toolkit | None |
| Observation Evaluation Toolkit | None |
| Observation Scripting Toolkit | None |

| From Package | Derived system requirements |
|---|---|
| Remote Observing Toolkit | None |
| Observation Scheduling Toolkit | None |
| Archive Toolkit | None |
| Pipeline Toolkit | None |
| Pipeline heuristics | None |
| Calibration source toolkit | N/A |

## 13.5 Design concepts

## 13.6 Implementation

### 13.6.1 Implementation prior to first development cycle

A prototype version of the web-based tool that searches the VLA-VLBA unified calibrator source database has been constructed and is currently available to all NRAO users. A link to the tool is located on the VLBA Calibrators List web-page (http://magnolia.nrao.edu/vlba_calib).

The prototype tool allows users access to more physical information about prospective calibrator sources than earlier search tools. The calibrator source database has been greatly expanded by the addition of more sources, much better position determinations and better descriptions of source flux densities at various frequencies.

The prototype tool is, however, an HTML/cgi entity. The final version will be written in Java and will exhibit a much higher level of interactive sophistication.

### 13.6.2 Implementation in first development cycle

A java applet version of the calibrator search tool has been built and currently under a work group testing at NRAO. The tool searches the combined VLA-VLBA calibrator source database. The tool can be accessed at (http://bernoulli.aoc.nrao.edu/e2e/calib).

The calibrator search tool includes an applet and two data access proxies. One proxy is used to access the flux history database on aips2.aoc.nrao.edu and another proxy is used to access the calibrator database on yoda.aoc.nrao.edu. The tool accesses the source image and source visibility gif plots on hernoulli.aoc.nrao.edu (the web server that host the applet). The main GUI interface is shown below.

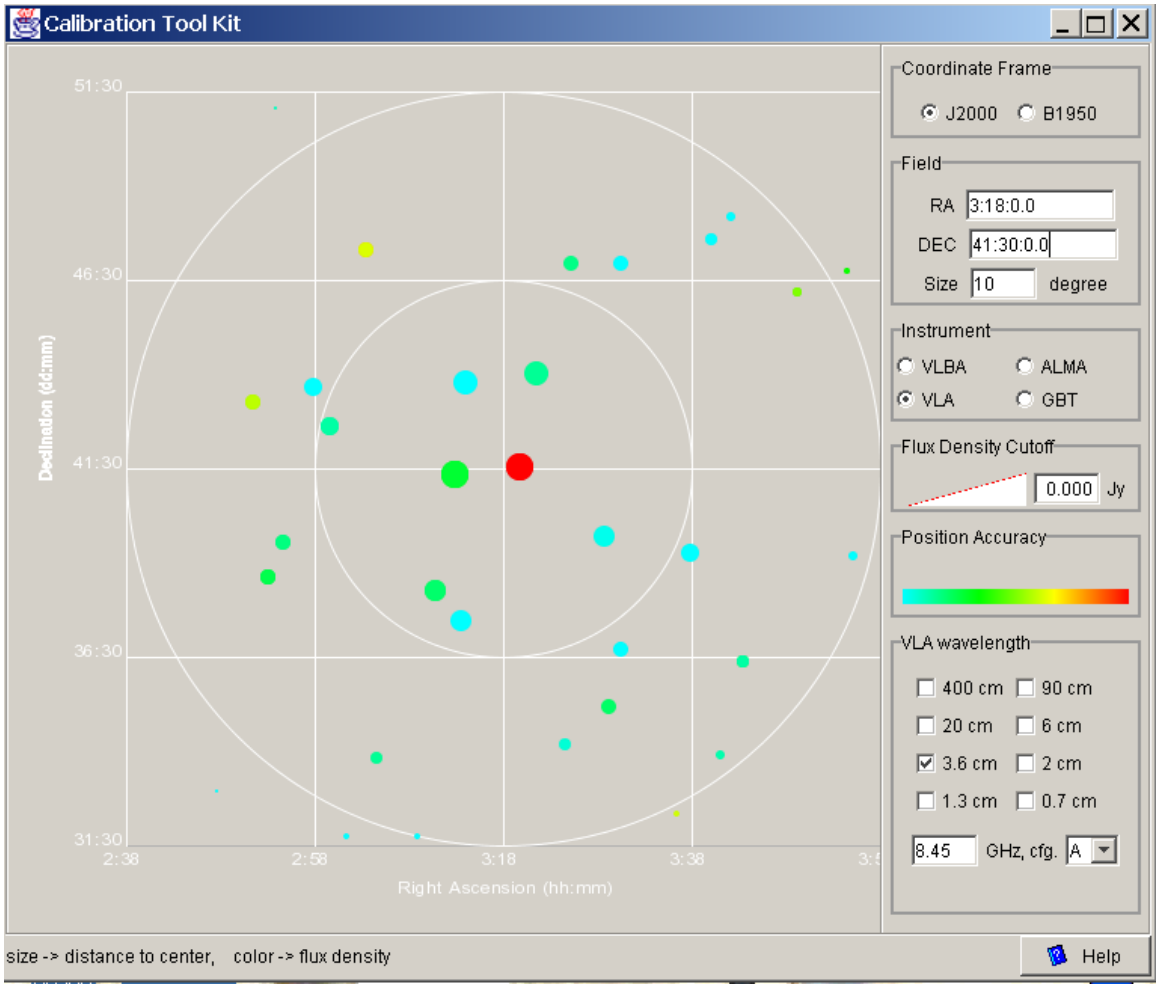



**Figure 128 GUI for Calibration Source Toolkit**

## 13.7 References

# 14. CYCLE 1: NOVEMBER 2001 – JULY 2002

*Author: Tim Cornwell*

*Revision date: 2002/07/12*

*Status: final version*

## 14.1 Goals

The goals of cycle 1 are to investigate and prototype in the areas of archiving, pipelines, and scripting.

1. To get expertise in dealing with large volumes of data, we will move the VLA archive onto disk. Highest priority will be given to a contiguous block of tapes from the late nineties.
2. We will develop a meta-data extractor in AIPS++ and test it on various forms of MeasurementSet.
3. A catalog of original data and derived results will be constructed from the meta-data.
4. Interfaces to the catalog will be provided from AIPS++ and the web.
5. It will be possible to download data sets via Internet protocols (e.g. ftp).
6. An initial pipeline will be constructed to read data from the archive and put derived results back into the catalog.
7. All data of a simple class (*e.g.* A configuration, 8 GHz, continuum) will be processed into images, and the results inserted back into the catalog.
8. We will investigate whether the GBT observe design can be adapted to allow specification of a simple class of observations (*e.g.* A configuration, 8 GHz, continuum).
9. In addition, the preliminary work done on the Calibration Source Toolkit will be continued.

## 14.2 Agreement with AOC computing

As agreed with AOC computing:

We plan to acquire and deploy an interim archive for the VLA. The purpose is to (a) improve the user access to the existing archive, and (b) to provide a test bed for e2e development of archives and pipelines. This interim archive may grow into the final VLA archive but that is not necessarily so.

This will be a joint project between DM/e2e and the AOC computing division. This plan covers the installation, filling, and deployment of the interim archive.

The existing VLA archive is stored on ~1500 tapes of up to 2 GB etc totaling 2TB altogether. The format of each tape is a number of files, usually one per project. An existing online database catalogs the archive so that one can find data by e.g. project code, source name, time range, etc. Monitor data are present on separate, non-cataloged tapes. Other ancillary data are present on various computers.

The steps are:

1. To acquire an interim archive server for the AOC. The archive server will store VLA archive tapes online on disks. The costs of server will be borne by Observatory Wide Computing. The initial storage purchased will be 2TB, giving sufficient room for a few years of VLA data plus copies, results, etc. Further storage will be purchased incrementally to minimize costs.
2. To copy the existing VLA archive tapes to the server. Tape numbers will be mapped to directories, and tape files to disk files. The exact strategy of which years to copy will be determined later.
3. To revise the existing VLA archive interface to access data on the server. Since the mapping between tape and disk files is one-to-one, this step will be quite straightforward.
4. To fix archive catalog errors known to us at this time by rebuilding the catalog.
5. To allow limited ftp access to non-proprietary data in the archive. The ftp access will be limited as necessary to avoid overloading the AOC bandwidth. Some software and operational support will be needed for this step.
6. To investigate the issues involved in moving the VLBA archive to disk. This would be a pilot project aimed at moving only a very small part of the VLBA archive.

The tentative timeline is:

```
      December 1:             Install server.
      January 1, 2002:        Start copying tapes.
      January 1:              Deploy  modified  interface  to  archive
server internally.
      January 1, 2002:        Start internal tests.
      March 15:                     First external tests.
      April 1:                      Announce   availability   in   NRAO
newsletter.
      February 1, 2003:       All copying complete.
```

AOC computing is primarily responsible for steps 1, 2 and 3. DM/e2e is primarily responsible for steps 4, 5, and 6. Each side will aid the other as appropriate. VLA/VLBA Operations will be responsible for system maintenance, and for support of interim archive operations (i.e. controlling access to the archive). The DMSWG will provide assistance with testing and evaluating the archive. If DM funds permit, temporary staff will be hired to aid in the transfer.

## 14.3 Specific activities

The following specific activities are planned to run from November 2001 to July 2002.

### 14.3.1 Archive

We plan to develop a useful but limited interim archive for existing VLA observations. The main question here is whether the AIPS++ MeasurementSet can be used as a source for meta-data for the catalog. A secondary question is whether the other part of the archive (*i.e.* the physical handling of data archives) warrants out-sourcing.

- Physical deployment and setup
- Development of catalog construction software
    - Software to aid physical loading of tapes (what's needed?)
    - Develop a filler (probably into AIPS++ tables)
    - Develop a filler for derived results such as images, processing logs, and calibration tables
- Development of catalog inquiry software
    - AIPS++: tool to interface queries to the catalog
    - Web: web pages to query the catalog
- Development of software to allow FTP access to cataloged files, including derived results.

### 14.3.2 Pipeline

We plan to develop an interim pipeline to perform some processing of some data from the VLA archive. We hope to learn enough to be in a better position to consider the merits of further development versus out-sourcing.

- Physical deployment of machine
- Development of a resource manager to find resources (*e.g.* hosts, disk space, *etc.*)

- Development of a simple executive to control flow of execution of scripts (*e.g.* run this script with that data set on that machine, with that amount of disk space)
- Development of a number of simple processing scripts (*e.g.* fill, edit, calibrate, image and possibly self-calibrate A configuration, 8 GHz observations)

### 14.3.3 Scripting

The intention in scripting is to start investigation of the adaptation of GBT Observe to handle observation scripting for the array telescopes.

- Investigation of expansion of GBT Observe to (E)VLA to allow passage of information for VLA observations sufficient for VLA 8GHz A-configuration. This is only a conceptual demonstration: the script will not be executable.

## 14.4 Current plan

**Table 31 Strawman designs at start of cycle 1**

| From Package | Current design | Status |
|---|---|---|
| Operational Model | Document | Done |
| Proposal Submission Toolkit | Web form or Java-based tool | Deferred |
| Proposal Management Toolkit | Java-based tools plus database | Deferred |
| Telescope Simulation Toolkit | AIPS++ tools | Deferred |
| Observation Evaluation Toolkit | AIPS++ tools | Deferred |
| Observation Scripting Toolkit | GBT Observe, GUI editor | Investigation this cycle |
| Remote Observing Toolkit | AIPS++ tools | Deferred |
| Observation Scheduling Toolkit | OMS + local adaptations | Deferred |
| Archive Toolkit | AIPS++ plus rdbms? | Prototype this cycle |
| Pipeline Toolkit | AIPS++ tools | Prototype this cycle |
| Pipeline heuristics | Glish scripts | Prototype this cycle |
| Calibration source toolkit | OMS? | Prototype this cycle |

## 14.5 Acheivements

*To be filled in at the end of the cycle.*

In terms of the goals, the achievements are:

1. To get expertise in dealing with large volumes of data, we will move the VLA archive onto disk. Highest priority will be given to a contiguous block of tapes from the late nineties. *Done. We have loaded 67 VLA XH* tapes (equivalent to roughly 700 VLA 9 track tapes), and 4 VLBA distribution tapes.*
2. We will develop a meta-data extractor in AIPS++ and test it on various forms of MeasurementSet. *Done. We have tested it on VLA (very well) and GBT (a little) MeasurementSets.*
3. A catalog of original data and derived results will be constructed from the meta-data. *Done.*
4. Interfaces to the catalog will be provided from AIPS++ and the web. *Done. The web interface has been tested "by hand", and the AIPS++ interface by the pipeline.*
5. It will be possible to download data sets via Internet protocols (*e.g.* ftp). *Close but not yet complete. This is close to being tested by the data analysts.*
6. An initial pipeline will be constructed to read data from the archive and put derived results back into the catalog. *Not quite done – the final step of inserting results into the catalog has not been done but everything else has.*
7. All data of a simple class (*e.g.* A configuration, 8 GHz, continuum) will be processed into images, and the results inserted back into the catalog. *Done. In addition, we did some work on processing an HI synthesis observation.*
8. We will investigate whether the GBT observe design can be adapted to allow specification of a simple class of observations (*e.g.* A configuration, 8 GHz, continuum). *Not done. A lot of thought has gone into the Observing Scripting but with little concrete resulting.*
9. In addition, the preliminary work done on the Calibration Source Toolkit will be continued. *Done. A working version has been completed and is being tested internally with deployment outside the Observatory to follow shortly.*

In addition, we did the following unplanned work:

- We performed analysis and design of the proposal submission and proposal management toolkits. As a result, we can implement a prototype in the next development cycle.

Our major misestimate was in the amount of time needed to procure and deploy the archive server. Since this depended on heavily-overloaded AOC Computing Staff, it took

roughly 4 months longer than expected. Fortunately, much of the associated archive implementation could be done independently. The net result is that we are behind on tape copying.

# 15. CYCLE 2: JULY 2002 – MARCH 2003

*Author: Tim Cornwell*

*Revision date: 2002/07/11*

*Status: initial version, sufficient to initiate*

## 15.1 Goals

The main goal this cycle is to complete the design phases to a level of detail sufficient to allow planning, costing, and scheduling of the complete e2e system. In addition, we expect to deliver working versions of the archive for VLA and GBT.

## 15.2 Specific activities

## 15.3 Current plan

**Table 32 Strawman designs at start of cycle 2**

| From Package | Current design | Status |
|---|---|---|
| Operational Model | Document | Done |
| Proposal Submission Toolkit | Java servlets, Oracle | Prototype this cycle |
| Proposal Management Toolkit | Java servlets, Oracle | Prototype this cycle |
| Telescope Simulation Toolkit | AIPS++ tools | Deferred |
| Observation Evaluation Toolkit | AIPS++ tools | Deferred |
| Observation Scripting Toolkit | GBT Observe, GUI editor | Investigation this cycle |
| Remote Observing Toolkit | Java and AIPS++ tools | Deferred |
| Observation Scheduling Toolkit | OMS + local adaptations | Deferred |
| Archive Toolkit | AIPS++ | Refinement this cycle |
| Pipeline Toolkit | AIPS++ tools | Second prototype this cycle |

| From Package | Current design | Status |
|---|---|---|
| Pipeline heuristics | Glish scripts | Second prototype this cycle |
| Calibration source toolkit | Java tools | First generation complete |

## 15.4 Acheivements

*To be filled in at the end of the cycle.*