NOTE 214 – Design for User Interaction with Data (Green Bank)

J. McMullin

Oct 23, 1997

Contents

1 Motivation

These are the compiled notes from meetings discussing MeasurementSets in Green Bank.

These meetings were setup to help resolve a number of issues relating to user interaction with data in Green Bank and elsewhere.

Many thanks to Bob Garwood, Dana Balser, Rick Fisher, Ron Maddalena, and Mark McKinnon for their substantial input.

2 Introduction

A MeasurementSet (MS) is an AIPS++ table containing data from a telescope. Each telescope is expected to record data in a unique fashion; MSs will be the common description of how telescope measurements will be stored within AIPS++. Note 191 describes the complete definition of a MS including the predefined keywords, subtables and columns of a MS.

3 Design Items:

The following items detail specific points about the structure and management of the data from telescope acquisition to reduction.

3.1 Data Structure:

1. The consensus is that the backend is a natural division for MSs; each backend should have a separate MS.

2. A single MS per backend per observing run is deemed the most convenient result if subentities based on source, time, etc. can easily be obtained (this is the case). However, there are concerns/questions regarding how large an MS can be before it becomes overloaded (compromises performance). There will be projects with many thousands of spectra (e.g. 100 MBytes of data or in the case of pulsar data, a run can be GBytes-this was awkward in the original tables).

   Comments: Memory and storage: The storage mechanism for each column in a MS is not the same. The DATA column is stored as a tiled array (each spectra is a single tile); the data is not all in memory at any one time; still need to investigate cases of large MSs.

3. Mixing complex and float data in a single measurementset seems to be the most simple from a user stand point (having complex rows columns along side of float columns). However, if the bloat factor from making everything a complex number isn't too high (what is the factor; is it just a factor of 2 bigger to make all floats complex?), and if it simplifies the programming, then having all MSs be complex is a backup solution. No one seemed enthusiastic about separating complex MSs and float MSs.

   Comments: This should be possible without any data bloat, though it requires a complex storage manager. It may work like something like this(B.Garwood):

   (a) *For a given row, the data is either complex or real, there is NO mixture on a single row.*

(b) *The storage manager makes use of the standard storage managers underneath to store the rows with real data as real values and the complex data as complex values. The type of data being stored determines what underlying storage manager is used.*

(c) *The storage manager "serves" the data to the user in the type requested: bloating the data on demand if float data is requested to appear as complex. I'm not certain what the storage manager should do if complex data is requested to appear as float - either it should refuse and give an error (probably the sanest solution) or it should give a warning and return the magnitude (I don't like this at all).*

(d) *We would have to allow the "type" of data in a row to change based on what new data the user placed in that row. e.g. if you did a "put" using float data where you once had complex data the SM will need to remove the previous complex data and store the new float data as a float.*

4. It would be useful to add a subtable which keeps a history of all of the operations performed on a MS.

   Comments: There is current work on specifying a HISTORY table within a MS.

5. Ron suggested a look through the UNIPOPS keywords to see if anything was noticeably absent. There were several items missing but they fell under the catagory of being telescope specific parameters or derivable from existing quantities.

6. It would be useful to develop a header command which would print a range of parameters for one or several scans in an MS; this could take a parameter such as full, brief, etc.

3.2 Data Management:

1. For the user interaction with online data, the MS should be added to automatically with no action required from the user. This will require a filler daemon running and updating the MS along with a quick indexing technique for large numbers of files; this has already been discussed and is in development. There should be a summary file of filled scans which is updated to screen to let the user know the

current status. The summary information should include things like date of observation, object name, backend, scan number range, etc.

In addition, although the table browser is sufficient for a specific MS, there is a need for a simple tool for extracting subsets of data from an MS based on standard selection criteria (object name, time, frequency, etc). This tools should be accessible to the User.

Comments: See Figure 1 for design sketch.

2. There is a question as to how data will be archived. Will data be archived as MSs or some other format?

   Comments: The original FITS files should be archived; FITS is trusted as a long term storage format. MSs may also be archived though conversion programs may fade into obsolesence; ultimately, a program which converts between FITS and MSs will be required (tricky to do for a general case and still maintain the complexity of an AIPS++ table). SDFITS should be the primary product of an observing run.

   We would also like to see an automatic archive of the SDFITS data with an archive summary. An archive tool should also be constructed which will allow selection of archive data to be filled into an MS.

3. It was suggested that it is important to provide avenues from MSs to other software packages (e.g. CLASS).

   Comments: Actually, it only really needs to provide a route to SDFITS; all viable packages should be able to read this format.

4. Security. Some data security needs to be present but not much. This will be developed by Mark McKinnon and will probably fall along the lines of restricting users access to data outside their project while giving operators full access to all data.

4 Unresolved Issues

1. How are calibration data handled and applied?

   Comments: Not clear yet.

2. Will process from SDFITS to MS be reversible?

   Comments: SDFITS->MS preserves all the information while MS->SDFITS may lose information in some cases. The ms2sdfits program *will eventually attempt to identify when information is being lost. Ultimately, AIPS++ will provide a lossless MS->SDFITS conversion (this will be complicated involving lots of FITS binary tables and their relationships specified in some other table, etc.).*

5 User Design Sketch

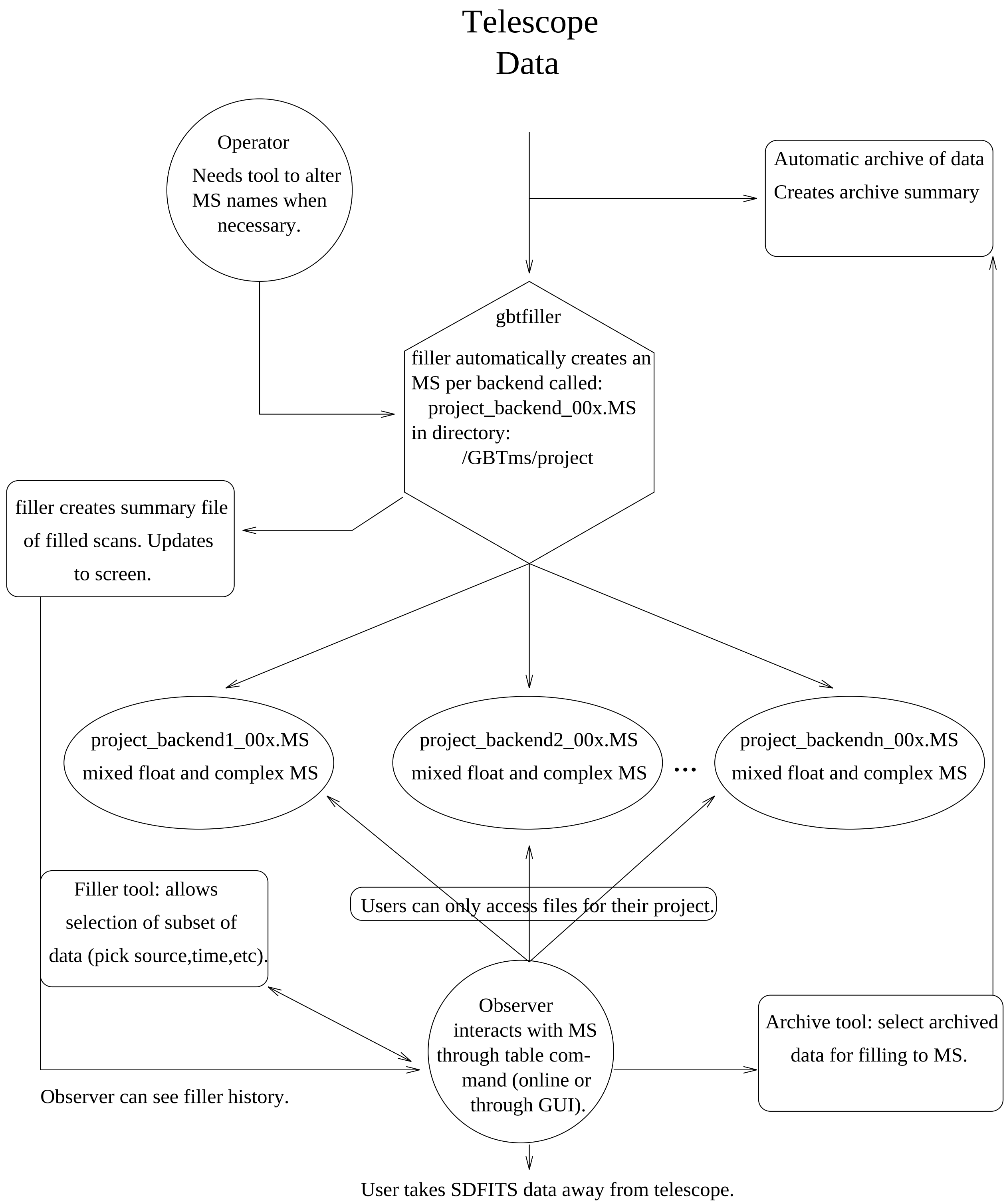

Figure 1: Sketch of User Interaction