# ALMA Memo 465 & IRAM Memo 2003-4

# Case for interoperability as an ALMA off-line model


J. Pety[1,2], F. Gueth[1], S. Guilloteau[1,3],
P. Teuben[4] and M. Wright[5]

1. IRAM (Grenoble)
2. LERMA, Observatoire de Paris
3. European Southern Observatory
4. Astronomy Department, U. Maryland
5. Radio Astronomy Laboratory, U.C. Berkeley


2003-06-26

*Change Record*

| Revision | Date | Author | Section/<br>Page affected | Remarks |
|---|---|---|---|---|
| 1 | 2003-05-20 | Jérôme Pety | All | Initial version |
| 2 | 2003-06-26 | Jérôme Pety | Last | Included referee's comments |

$Id: interoperability-for-offline.tex,v 1.33 2003/09/15 09:13:52 pety Exp $

# Contents

**Abstract**

In this memo we argue that **interoperability** between the existing radio-interferometry packages provides a fast, robust, flexible, complete, and user-friendly data reduction package for ALMA. We discuss the meaning of these terms from the point of view of the end-user, the data reduction specialist, and the package maintainer, and show that an open software environment provides a cost effective, flexible, and scientifically more valuable data processing environment than a monolithic data reduction package. We discuss our first steps toward interoperation with `MIRIAD` and `GILDAS` using `python` as a command line interface.

For a few years, `python` has been a natural candidate for the command line interface both inside and outside ALMA. This is the first step toward interoperability. We now suggest that ALMA should think about its data formats (both native and exchange) with the community to promote interoperability.

# 1 Introduction

In a companion memo, we discussed three state-of-art data reduction packages (DRP) for $\lambda$ mm radio interferometry. To do this, we presented the results of evaluations of the `GILDAS` and `MIRIAD` software packages following the same template that was used for the `AIPS++` audit for compliance with the ALMA Offline Data Processing Requirements (SSR). Those evaluations show that about 2/3 of the SSR requirements are fulfilled by each data reduction package, and almost 90% are fulfilled if we use existing software from all three packages. Moreover, `GILDAS` and `MIRIAD` summarize almost 15 years of experience in $\lambda$ mm radio interferometry and will continue to benefit from daily confrontation with real $\lambda$ mm data over the next 10 years of ALMA construction. A natural conclusion is that ALMA would greatly benefit by using software from the existing packages which were designed for $\lambda$ mm arrays.

The ideal off-line package for an instrument should at least be fast, robust, flexible, complete and user-friendly. This mixture of properties is difficult to achieve because they can be contradictory. Indeed, compromises between these properties are often made to build a real off-line package. We argue in this memo that **interoperability** between the existing radio-interferometry packages is a cost efficient way to reduce the magnitude of the trade-offs. By interoperability, we mean the possibility to do the required data reduction steps (*i.e.* data editing, calibration, imaging, data analysis and publication plots or any part of them) coded into different packages from the same Command Line Interface (CLI) and/or Graphical User Interface (GUI).

In section 2, we describe *i)* how the needs of an off-line package can be contradictory and *ii)* how interoperability can help. In section 3, we then analyze what is required to achieve interoperability. In section 4, we describe the steps currently undertaken to interoperate two data reduction packages adapted to current $\lambda$ mm arrays, namely `GILDAS` and `MIRIAD`. We finally suggest that the data format (both content and implementation) should be widely agreed upon by the community as this greatly facilitates interoperability.

# 2 Why interoperate?

There are two models for off-line data processing: a monolithic DRP able to handle many instruments, or an open software environment which uses parts of software dedicated to solve particular problems.

## 2.1 The needs of an off-line package

Three kinds of actors must interact around a data reduction package:

**End-users,** astronomers who just want to use the DRP to make science from their data;

**Reduction specialists,** usually astronomers who tune existing algorithms for use in the data processing for a specific instrument, or design new data reduction algorithms for new observing modes;

**Package maintainers,** usually software engineers.

The frontier between these three kinds of actors is blurred and it may be that the same person is in turn end-user, data reduction specialist and maintainer. However this classification is useful as it allows us to illustrate the different meanings that fast, robust, flexible, complete and user-friendly may have depending on the point of view. This is important as it enables the reader to see how different compromises can affect each kind of actor.

**Fast** As we are dealing with the off-line, we will look at the speed problem only from the point of view of interactive use. The speed of data reduction steps may be traded-off against ease of maintenance or generality. The value of speed depends on *i)* the actual time that the data reduction step takes and *ii)* the number of times this step will be repeated. Indeed speed is never a problem when data reduction steps take only seconds. However, as an order of magnitude, for small (*i.e.* continuum) data sets of ALMA+ACA complexity, deconvolution typically takes 15 minutes inside `GILDAS` on *today's* standard PCs (cf. ALMA memo 398). In such an example, an increase of a factor 2 in speed would enable us to make 8 tests (or spectral channels) in an hour compared to only 4 in the current situation. Such a factor 2 is an essential gain for the reduction specialist who will have to repeat this step a large number of times to tune an algorithm. This is also an important gain for end-users who will probably reduce their data at the Regional Support Centers where performance and throughput are important factors in optimizing the science. With a faster DRP, users can reduce their data more quickly, and are able to analyze their data more thoroughly. At some point, computing speed always limits the science. A project that takes 2 months to analyze is less likely to get done than if it took only one month. A user would only be able to analyze 2048 spectral channels, even though 4096 were recorded and the unanalyzed data had weak spectral lines. Etc. We also argue that benchmarking on standard PCs is the most relevant for reduction specialists as they do not always have a powerful, expensive computer.

**Robust** End-users expect predictable behavior from the DRP, *i.e.* it should always correctly filter out the mistakes an end-user might make. Reduction specialists want to use the software in unexpected ways. They thus expect that the robust software is not making unnecessary assumptions (on its input for instance) which will lead to bugs if violated. Finally, maintainers are responsible for the package robustness. This may lead them to impose a limiting number of programming languages to ease maintenance.

**Complete/Flexible** The completeness of a DRP is linked to its flexibility and generality. End-users want all the needed steps already to exist and be easily reachable. Reduction specialists want flexibility in the available tools to be able to create missing functionalities, verify results, and implement new ideas. Maintainers want to provide very general tools as every operation will then be possible from a small number of tools. This last scheme has two drawbacks: *i)* general tools are often also very abstract, which make them more difficult to learn for the end-users and *ii)* generalities prohibit intelligent approximations which can simplify and greatly speed up reduction algorithms without lost of correctness in the results. For example, general tools may prohibit approximations that lead to errors much smaller than the noise level.

**User-friendly** For the end-users, a friendly DRP must have good cookbooks and good heuristics. Intelligent scripts should know enough about the instrument and the current standard observing mode to guess correct input parameters for most of the algorithms used in the reduction. This would allow an end-user to push only one button per conceptual step of the data reduction. There should be simple ways to change automated choices, a good visualization of data and probably an intuitive GUI to guide users through all the possibilities. The reduction specialist will want good documentation of the basic tools, an easy interaction with the data through an efficient CLI and the possibility to quickly include new algorithms written in the language he/she is comfortable with into the overall architecture.

### 2.2 How interoperability helps

As defined in the introduction, interoperability will help the end-user, the reduction specialist and the package maintainer.

**End-users** The usual approach for end-users is to calibrate and edit the data in the package which was designed for each specific instrument, image and deconvolve in a package which has the needed functionality (mosaicing, deconvolution algorithms, and speed), and scientifically analyze the data in a package which is comfortable and familiar. The end-users increase the yield of their data analysis when they are able to make the best use of each package and avoid loosing time by using software they know.

**Reduction specialists** Interoperability can help reduction specialists in three ways. First, it allows reduction specialists with different backgrounds to progressively discover other packages and functionalities. Second, interoperability considerably facilitates comparison between different methods of data reduction as it avoids recoding everything in a given environment. Indeed, comparison is an important step in selecting the most suitable methods. Phase I of the `AIPS++` reuse test showed the difficulty of recoding working algorithms. The ease of comparing data processing will also lead to improved algorithms as the reduction specialists will tinker with the code again, which they would normally not bother with. Third, interoperability allows ALMA outsiders who are familiar with one of the interoperating DRP to contribute new algorithms. This is particularly critical for ALMA where resources devoted to data processing are scarce.

**Package maintainers** Maintainers also gain from interoperability as their work-load is distributed among the different package maintainers. For instance, coming from $\lambda$ cm interferometry, `AIPS++` has good calibration algorithms for high Signal-to-Noise Ratio observations while the packages currently used for $\lambda$ mm interferometry have calibration algorithms adapted to low SNR observations. Although, those calibration algorithms are coded with very different technology, interoperability allows the maintainers to offer both possibilities to the ALMA users, without extra work and software maintenance.

Finally, the custom packages needed to reduce special observing modes (*e.g.* VLBI and pulsar observation) could also fit in this picture.

## 3 Key points for interoperability success

Interoperability may be thought at different levels depending on the size of reduction steps that may be interchanged from one DRP to another. The two extreme levels are:

**Weak interoperability** A typical example here is to do the whole calibration in one package, imaging in a second one, and data analysis and publication plots in a third one. In this case of limited interoperability, only an **exchange** data format is needed.

**Strong interoperability** For example, gain and bandpass calibration could be done in `GILDAS`, polarization calibration and imaging in `MIRIAD`, and the deconvolution methods of each package could then be tried in turn.

Between those two extremes, a continuum of interoperability modes are possible. However, strong interoperability is the one where most of the benefits of interoperability are obtained. In particular, this brings the maximum transparency to the end-user. This is also the easiest way to compare different algorithms for the same step of data reduction. Three basics ingredients are needed to enable strong interoperability.

**Same command line interface** All DRP should use the same CLI. Indeed, over the ALMA life-time and more particularly during early science, the standard observing modes and the associated reduction scripts will evolve. It is thus advisable to have a CLI as primary user interface and to build simple GUIs around the scripts instead of directly hardwiring the GUI to the underlying code.

**Same basic steps of reduction** The smaller the size of the basic reduction steps that can be interchanged with other DRPs, the stronger the interoperability. All packages must ideally have the same understanding of the basic steps of the data reduction: *i.e.* they must take the same input and give the same output information. This is the only way to be able to interchange reduction steps from one package to another.

**Same native data format** Finally, the most important point is to share the same native data format, first in content (as this is a pre-requisite of the previous point) and, if possible, also in implementation. Different implementations can be accomodated provided adequate mechanisms are implemented for cross-access between the packages. This has nevertheless a cost, at least in maintenance. If the implementations are too different, one will need to resort to dedicated fillers which will have a cost in efficiency and will decrease the flexibility of interoperability.

A thorough examination of the data formats (both native and exchange) used by ALMA should be engaged with interoperability in mind. Here are a few more thoughts on this point. Two data models are in competition:

- A "universal" data format from calibration, to data analysis through imaging. This model is conceptually appealing as all the information about the data is stored in the same structure.
- "Dedicated" formats for each step (e.g. raw data, calibration, imaging). In this scheme, much better optimization in input/output access are possible. The information available at each reduction step is limited to what is generally required to perform it. This could be a problem, but it is not serious because all the information is stored in the archive and can be obtained if needed.

The data structures should be flexible. Throughout the lifetime of ALMA, and especially during the development phase, as each new frequency band, correlator system, or observing mode is developed, we will need to modify the data structures. It should also be easy for the user to attach their own information to the data, such as notes about the data reduction logistics, reasons for choices of methods, notes about the source structure, etc.

# 4 Current efforts made to interoperate `GILDAS` and `MIRIAD`

`MIRIAD` and `GILDAS` compare very favorably in the tasks they do well, which is the core processing required for ALMA. These packages have good documentation, data selection, imaging, mosaicing and deconvolution algorithms. They are user friendly. They had to be as they were developed in an open market. `GILDAS` does not yet handle polarization because Plateau de Bure Interferometer is still single polarization. `MIRIAD` is weak in data analysis because the original BIMA decision was to concentrate on the core functionalities and to interoperate (although the expression was not coined in 1988), using other packages for data analysis. It seemed thus natural to make the needed efforts to start interoperation of both packages. This could serve as a proof-of-concept of interoperability benefits.

The first steps toward interoperation is to be able to interact with both packages from the same CLI. `python` has been chosen as this is the current baseline CLI for ALMA software. Interaction with `python` brings different challenges in both package. We have nevertheless started to gain experience: a few standard examples of `MIRIAD` scripts have been converted to `python`, and `GILDAS` is able to make a full calibration of Plateau de Bure data inside `python`. In both cases, the current solution is not satisfactory as *i)* it does not give atomic access to the underlying libraries and *ii)* it gives only indirect access to the visibilities. Studies continue and we are confident that we will find a user-friendly solution for both packages.

The second step is to use different algorithms from both packages inside the same `python` interface to reduce the same data from calibration to data analysis. As the `GILDAS` and `MIRIAD` internal data formats are different, we decided to first try weak interoperability, *i.e.* calibrate PdBI data in `GILDAS`, image in `MIRIAD` and then analyse the result in `GILDAS`. Data exchange is made through FITS. In the longer term, we envisage that we will be able to read `MIRIAD` data formats directly into `GILDAS` and vice versa.

Our goal is to make available to the community a fully working but limited example by end of summer 2003.

## 5 Conclusion and suggestion

The way the off-line *audit* has been designed implicitely assumes a single global solution for all ALMA data reduction needs. In practice astronomers today use a variety of packages for their private data analysis. It is quite usual for astronomers to calibrate data in one package, image in another and analyse their data in a software package which they find comfortable.

It is vital for the success of the ALMA project that other software developers, particularly astronomers with innovative experiments and software, have easy access to ALMA data processing. Particle physicists can bring special back-end experiments to international accelerators, but it is increasingly difficult to incorporate user-hardware with arrays of antennas. Software is one of the few things astronomers can contribute, and should be helped to contribute, to ALMA. Indeed, in radio interferometry, cutting-edge discoveries are made using observing modes to the limit of the instrument. The analysis of such observations often requires special reduction algorithms that depends on the scientific goals. Interoperable software should facilitate integrating user software into the data reduction pipelines and offline processing.

The ALMA project is implementing today the mechanisms that will help the users to make best use of the instrument. We believe that software interoperability will maximise the science that the telescope produces because *i)* it maximises the user efficiency in daily usage and *ii)* it helps innovative experiments to happen. However, to be a success, interoperability must be implemented early on and use simple techniques. For a few years, `python` has been a natural candidate for a CLI both inside and outside ALMA. This is the first step toward interoperability. We now suggest that ALMA should think about its data formats (both native and exchange) with the community to promote interoperability.

## References


[1] Gueth, F., Guilloteau, S., Lucas, R., Pety, J., & Wright, M.C.H., 2003, "Evaluation of the `GILDAS` package for ALMA Off-line Data Processing", BIMA Memo 96 & IRAM Memo 2003-2.

[2] Kemball, A., Golap, K., Moellenbrock, G., Lucas, R. & Broguière, D., 2003, "`AIPS++` Reuse Analysis Test: Report on Phase I".

[3] Myers, S., Viallefond, F. & Morita, K.-I., 2002, "Audit of the `AIPS++` package for ALMA Off-line Data Processing", ALMA-SW memo 19.

[4] Pety, J., Gueth, F., Guilloteau, S., Teuben, P. & Wright, M.C.H., 2003, "Complementarity of the `AIPS++`, `GILDAS` and `MIRIAD` packages as seen from evaluations for ALMA off-line data processing", ALMA Memo 464 & IRAM Memo 2003-3.

[5] Wright, M.C.H., Tauben, P.J., & Pety, J., 2003, "Evaluation of the `MIRIAD` package for ALMA Off-line Data Processing", BIMA Memo 95 & IRAM Memo 2003-1.