ALMA Development Study: Unleashing ALMA for Big Data

Lee G. Mundy, Peter Teuben, Marc Pound, Amitabh Varshney, Horace Ip
(University of Maryland),
Leslie Looney, Douglas Friedel (University of Illinois),
Jeff Kern, Adam Leroy (NRAO)

1 Summary

This is the final report for the ALMA Development Study on enhancing user access to the science data in large ALMA data cubes and creating an environment for scientific data mining. The top goal of this study was to create a well-defined plan for development of the software infrastructure and prototype key applications for enabling science with large image cubes. The second goal was to scope the manpower and cost for achieving a specific level of functionality which brings strong scientific value. Based on the work done in this study, this team wrote an ALMA Development Project Proposal to implement an XML infrastructure and tool set. The costs and man-power requirements are presented in the proposal and are not reported here.

The main body of this report presents study results in the five proposed areas. First area: in access performance, we found that speed of access to data in CASA image format was efficient if the data were being accessed in the optimal sequence for the tiling; it can be inefficient if accessed against the "grain" of the tiling. This feature of tiling can be minimized by taking advantage of larger memory capacities in new generations of machines and allowing for a few different tiling choices where appropriate.

Second area: we developed a set of goals for our proposed tool set and a technical approach for our package which we call ADMIT: ALMA Data MIning Toolkit. We present a conceptual overview of the design and operation and how ADMIT fits into the ALMA and CASA environment.

Third area: we prototyped some tools for ADMIT as examples of what can be done and as mechanisms for defining the way that ADMIT infrastructure and tools can interact.

Fourth area: we explored a number of new algorithms that would enhance user ability to understand the science content of large data cubes. We specifically discuss the overlap integral and descriptor vectors.

Fifth area: the interface between modeling and simulation data and observational data de-emphasized as a study area so that more time could be spend in areas two, three, and four, and the Conceptual Design Document

The first accompanying memo summaries the results of timing measurements of access to data by a number different programs for a number of different data configurations.

The second accompanying memo is a Conceptual Design Document for ADMIT.

2 Introduction

ALMA will enable groundbreaking science over a wide range of fields. Exciting discoveries will be made with simple images of continuum emission and/or individual molecular lines. For many areas of science, however, such images are the tip of the iceberg; large multi-channel data cubes and large area maps will exploit the greater capability of ALMA. ALMA's high sensitivity and large correlator enable the collection of large spectral data cubes which contain a wealth of information about the kinematic, physical, and chemical state of the galaxies, molecular clouds, circumstellar disks, and evolved stars under study. ALMA's speed can enable wide field mappings and large sample surveys. The overall goal of this study is to outline the tools for effectively deriving science from large ALMA data cubes. We seek to facilitate standard scientific analysis and to enable new and creative ways to derive science from the cubes.

The original focus areas in the study as stated in the first section of the proposal were:

- "interfaces to CASA image cubes and infrastructure for efficient, versatile access to these cubes in an open-source/application environment;
- design of metadata structures that would accompany the astronomical and image data to enhance the utility of applications;
- options for quantitative visualization and the speed/capability trade-offs;
- applicability of and development paths for existing algorithms in the computer science community to identify and quantify structure of astronomical interest in large data cubes.
- interfaces for intercomparison of computational models with observational data "

These focus areas were refined over the course of the study. For example, the visualization area was de-emphasized in our study because a selected 2012 ALMA Development Study lead by Eric Rosolowsky focused on visualization of ALMA datasets. One of our team members (Peter Teuben) became an outside collaborator on that team; this facilitated information exchange and reduced overlapping efforts. Second, as a part of the study effort, we came to realize that the file size, Internet transfer time, and complexity of the data cubes (multiple windows with many channels) were themselves impediments for typical users trying to access the science in their data. These complexities represent an even larger barrier for scientists new to the radio field. Hence we expanded our study in this direction.

The following sections summarize the activities and outcomes of the study organized into sections following the above five itemized study areas. The two appended documents provide: (A) discussion of the problem of optimizing speed of computer access to the data and (B) an conceptual design document for an "ALMA Data Mining Tool (ADMIT)".

This study established the groundwork for a proposal for ADMIT that was submitted for the 2013 ALMA Call for Development Projects.

3 Access to CASA Image Cubes

We started our study by looking at approaches for accessing large ALMA datasets. Since image cubes, not *u,v* data, are the focus of this study, we looked at the access speeds for various operations

to data cubes in a number of different formats including the CASA tiled data format. Not too surprisingly, tiled data access works very well for balanced cubes, but slower access in some of the directions for stick-like cubes (large in the z-dimension compared to x and y). This is detailed in the attached memo.

In the course of this work, and from discussions with CASA programmers, several points were clear. First, CASA made the decision to follow a tiled storage approach rather than a memory intensive model. CASA has been effective in implementing this strategy. With the expanding capabilities of machines and the decrease in memory cost, it is attractive to consider tuning CASA to be more memory based. The tiled data storage approach of CASA is efficient when the data are being accessed in a sequence which follows the tiling; it can be significantly less efficient for access which is against the grain of the tiling scheme. While the tile definition can be tuned to the expected access pattern to give better performance, this is not currently utilized in CASA as an active option.

Recommendations: CASA should periodically review the memory model for the prototype user machine and tune CASA image storage as possible to optimize memory usage. CASA should review the usage patterns of programs to optimized the tiling for the most commonly used programs. There may be circumstances where it is sufficiently compute efficient to encourage users to store a given dataset in multiple tiling formats.

Given the evolving capability of machines and CASA, it was clear to us that there was no strong advantage to another data storage format to increase general compute speed at the cost additional disk usage. This conclusion might have been different if we had been focused on a specific visualization tool or application.

4 Science Metadata and the User

It became very clear to us while working with public access ALMA Cycle 0 data that efficient access was about more than just compute efficiency. Current ALMA datasets are contained in multiple large tar files which can be very costly in time to download. The cubes for each band have a large number of channels without information about what lines are in what channels. It is not a simple and efficient interface for the user of the science content in the data.

A large fraction of our study focused on the design of an XML-based scientific meta-data system and the design of tools that can be build on that system.

The concept is to create a value-added software package that integrates with the ALMA archive and CASA to provide scientists with immediate access to traditional science data products such as moment maps, and provide innovative tools for exploring data cubes. The proposed package is called ADMIT, for ALMA Data Mining Toolkit, a software suite that creates value-added data products from ALMA data image cubes directly from the ALMA pipeline for ingestion into the archive and provides a standalone infrastructure and applications for mining user-downloaded cubes.

The top-line goals of the ADMIT are to:

(1) make the scientific value of ALMA data more immediate to all users,

(2) create an analysis infrastructure that allows users to build new tools,

(3) provide new types of tools for mining the science in ALMA data, and

(4) increase the scientific value of the rich data archive that ALMA is creating.

ADMIT would provide capabilities, missing in the current ALMA baseline system, that are key to achieving ALMAs full science potential for both proposers and archival data users. The following are the top-line goals for the system and how they can be achieved:

Goal 1: To provide an immediate data access capability. This will be accomplished by creating a set of basic data products (BDP) which are included into the archive and available to the users for quick overviews of their data. The BDP are a combination of XML files, PNG images, and FITS files that will be produced by ADMIT software (a combination of Python scripts and CASA programs) from the ALMA approved pipeline image cubes (all correlator bands). The BDP consist of harvested header information about the source and observational setup, statistics of the data cube, spectra through emission peaks, identification of strong molecular lines present in the data, and integrated intensity, velocity centroid, and velocity dispersion maps of each identified line. The BDP images will be associated with XML metadata that contain the information about their creation. The BDP, in the range of 5-40 MB in size, will be ingested into the ALMA archive concurrent with ingestion of the ALMA pipeline image cubes. The ADMIT BDP will be available to the archive user as an independent item that can be selected and downloaded separately from the large image cubes. On the users local machine, ADMIT software (distributed as a package in CASA) is used to view and manipulate the BDP. Since the BDP are primarily XML and PNG, the information can be accessible to the ALMA archive database and CASA Viewer if desired by ALMA. The small size and summary scope of the BDP are ideal for:

- quickly exploring what is in the data set,
- deciding what full cubes should be downloaded, and
- cross comparing detections or emission properties across multiple sources.

An ALMA example is examination of results for a spectral line survey of 25 sources. Without downloading any of the data cubes, the scientist could quickly view the moment maps to classify sources based on line detections, line widths, and kinematics. Then, the full data cubes for sources of the most interest could be downloaded to further explore populations. The BDP assist in this process because they include the information about where the lines are located within each cube.

Goal 2: To provide a simple infrastructure that can be easily adopted by users to create new data mining tasks. We will support this capability by building the infrastructure from small re-usable unit tasks, documenting the pieces and creating example use cases. Since ADMIT is a combination of XML, Python, and CASA, the component languages are all familiar to moderately experienced CASA users. Users can add features directly to create their own custom ADMIT pipeline and add new tools to the toolkit that can be shared among collaborators and the community. A custom ADMIT pipeline is especially useful for large surveys where astronomers need to uniformly process the sample, perhaps repeating the analysis many times to iteratively adjust parameters after examining the results.

Goal 3: Use the BDP and ADMIT infrastructure as groundwork for building innovative tools for mining information from the data cubes. We propose to build several specific tools as examples of the potential of ADMIT. First, with the spectral lines identified, it is possible to calculate overlap integrals between velocity channels to see how the emission changes within an individual line, and/or to calculate the overlap integral between different lines in a dataset to see how the spatial distribution of the emission is correlated. We will write a CASA task to do this with the outputs of CASA image and XML/PNG data products. An example use case is an observation of a hot core source where the data cubes have scores of detected lines. The overlap integrals will show how the spatial distributions of the lines compare.

Past these familiar analysis tools, we will develop analysis based on a descriptor vector approach to characterizing the observed emission. The idea here is taken from contemporary computer science image analysis techniques where the image is divided into a set of sub-regions (for example Nyquist-sampled beams), each characterized by an N-dimensional vector according to a set of prescriptions. The distance between vectors can then be calculated to identify which parts of the image are most similar or most different, or to search for which parts of the image are most like a selected region. For instance, in the image recognition world it is possible to define a vector that well-characterizes the image of a person within a multi-gigapixel data cube, and then find all of the people in the data cube by identifying all vectors with similar characteristics. An ALMA example would be to analyze spatial molecular line distributions in images of a local galaxy. Starting with a dataset of N line images, a descriptor formulation can build vectors for each Nyquist-sampled region (or a chosen resolution) across the image with significant emission in any line. These vectors would characterize the emission in each molecular line in the project (all bands and multiple correlator setups if observed) by, for example, intensity and line width. ADMIT would then be able to identify regions throughout the image that have vector values close to a user selected reference position; for example, locations where both CO and SiO were present but not CH3OH. With current software, this comparison would require tedious visual inspection of the cubes. For this project, we have experimented with infrastructure for the descriptor vector task and create descriptor formulations for several standard ALMA applications. We will document the descriptor formulation procedure so that users have the option to build on this approach, share vector formulations, and build libraries of formulations.

This innovative tool aspect of ADMIT can also be a compute layer for creating new image products that can be fed into an independent visualization tool. With XML, PNG, and FITS/CASA images as the ADMIT products, it will be easy to design-in compatibility with CASA Viewer to enhance its use. We would also look forward to working with any new ALMA visualization tool that might be funded to define an XML/image interface.

Goal 4: To optimize the long term science value of the ALMA archive. The key here is to make the archival data easily accessible to science-based data-mining investigations. The BDP for each project is a first step in this direction. It allows the scientist to download the overview information to see if the project contains data of interest. The second step is that the ADMIT scripts to create BDP can be executed on the local machine to create a new and, if desired, custom version of the BDP. Thus, the scientist could download 10 or 50 data cubes from a single or many projects onto a directory on their local disk and then run the ADMIT script to generate new BDP that contain information for all of the image cubes in the directory. A future possibility would be

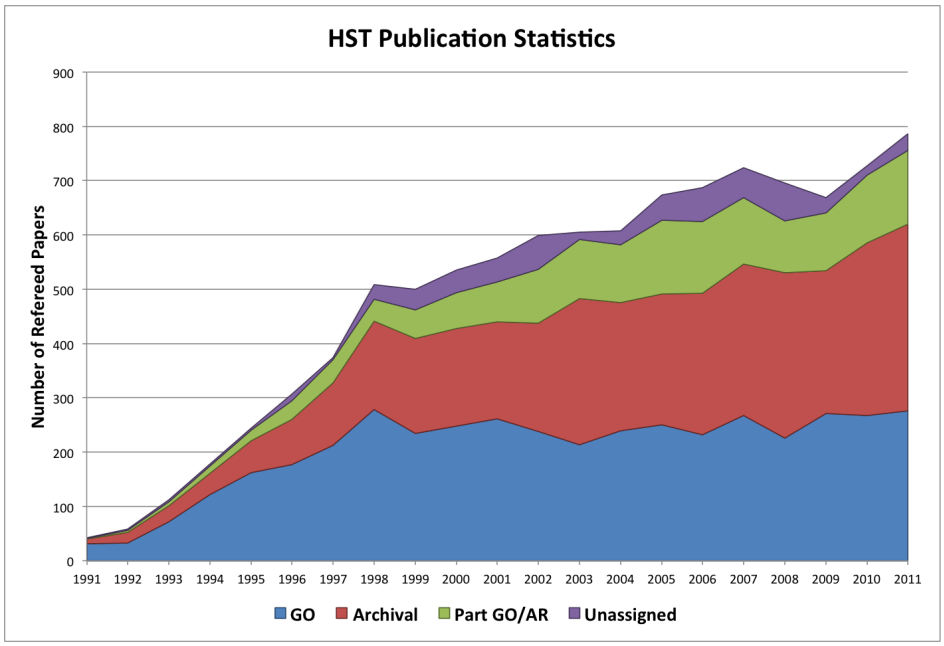

Figure 1: Number of HST GO and archival papers published as a function of year (figure from White et. al. 2010, "The High Impact of Astronomical Data Archives", ASTRO2010 position paper, http://archive.stsci.edu/hst/bibliography/pubstat.html

for the BDP XML metadata to be ingested into an archive database tool to enable searches based on user requests; this would be an optional outcome to be implemented in collaboration with the ALMA Archive team if they choose. An ALMA use case example is a scientist downloading the BDP of two or three large surveys of Class 0 protostellar sources. From examining the BDP, the scientist can decide what full images cubes to download from the archive. With all of the cubes on the local directory, the custom ADMIT scripts can be run to create a uniform set of BDP for the selected sources which sets the stage for comparisons and further analysis.

It is important to emphasize that the ALMA archive will be a tremendous resource for science after only a few years of operation. Tools that maximize the science from the ALMA archive amplify the impact of the instrument at little cost. As an example, Figure 1 shows that the Hubble archive is extremely productive, producing more archival-data based papers per year than papers from current observations. In addition, science papers with archival datasets have been found to have about as much impact in the community to original papers from the same datasets (White et. al. 2010, "The High Impact of Astronomical Data Archives", ASTRO2010 position paper, http://archive.stsci.edu/hst/bibliography/pubstat.html). It is crucial improve the end-to-end user access to the ALMA data even at this early stage because ALMA data, with so many spectral channels, is much richer scientifically than the early HST data.

4.1 Technical Approach Overview

ADMIT would be an add-on software package that interfaces to the CASA software system and ALMA archive, implemented both at the server side of the archive and as a desktop application with a full GUI. The ADMIT core is the infrastructure layer that defines XML structures and programmatic pipelines, extracts and organizes scientific metadata from image cubes, defines the ap-

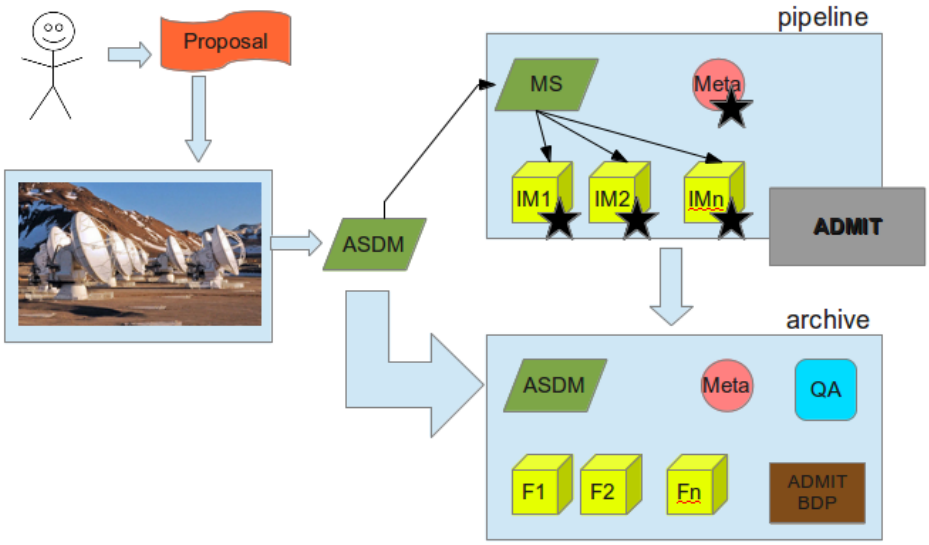

Figure 2: ADMIT and ALMA archive interface. Admit is an add-on suite that works on the data cubes output from the ALMA pipeline and their associated metadata (all marked with the black star symbol) ADMIT creates the Basic Data Products that are placed into the archive with the data.

plication programming interface (API) for higher-level tools, provides for I/O interaction with the XML, and computes scientifically relevant quantities. Built upon the infrastructure layer are specific pipelines to produce Basic Data Products (BDP) and the ADMIT Tools that provide advanced functionality for scientific analysis. ADMIT is targeted at both a novice user (via the convenient GUI), as well as an experienced user (via a Python toolkit within the CASA framework), and is designed to be extensible.

The interface to CASA is accomplished primarily through XML files and Python scripts. Where possible, ADMIT will call CASA tasks via the XML parameter files to accomplish computations. The new tools in the proposal will be Python or CASA tasks, depending on the required level of access to CASA data and whether the task requires speed optimization through more direct access to the CASA core routines. ADMIT will be designed and tested to work in the CASA environment.

4.1.1 ADMIT Interface to ALMA Archive

The interface to the ALMA Archive will be simply through the creation of an ADMIT data package: a tarball or other standard package format as requested by the Archive. The ADMIT data package will be ingested, archived, and served to the user as a single item. The user downloads the ADMIT data package by selecting it through the standard ALMA Archive interface. This approach has been discussed with members of the ALMA archive team (Felix Stoehr and Mark Lacy) and is straightforward. Since ADMIT outputs are standard XML and PNG images, it is a future option for the Archive Database to ingest the ADMIT information to enhance the archive search capability; it is not a baseline requirement for ADMIT but it is designed as a feature.

Figure 2 shows how ADMIT slots into the Reduction Pipeline and ALMA Archive. The ADMIT core processes (gray box) are executed after the observatory certifies that the image cubes meet proposal science requirements. Information for the BDP is gathered from the image cubes, relevant

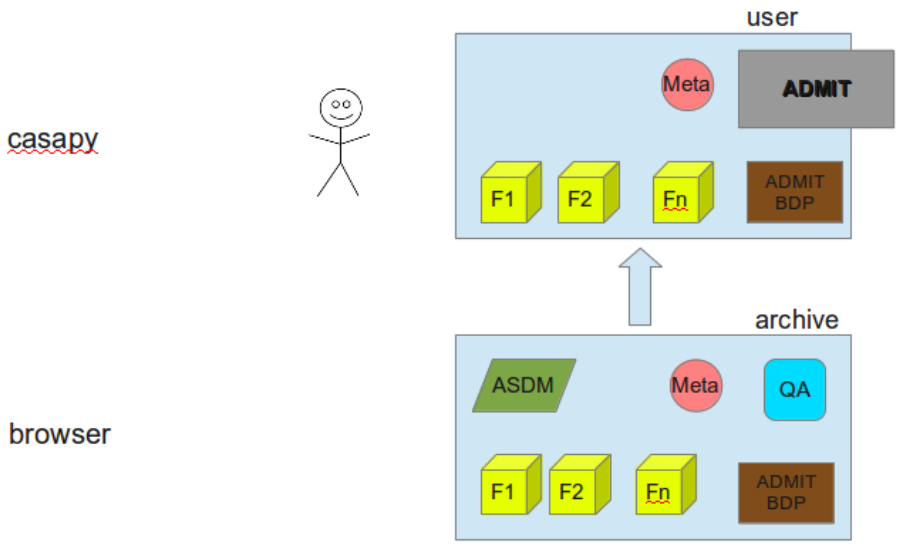

Figure 3: ADMIT Basic Data Products in the archive browser (bottom) and downloaded onto the user desktop. The downloading of the actual data cubes is not required to be able to view the BDP; it is required if the user wishes to recalculate BPP on the local machine.

ALMA pipeline metadata outputs, and calculated quantities. These are packaged into a file that is ingested (brown box).

Figure 3 shows the ADMIT BDP in the archive. Through the standard ALMA Archive browser interface, they can be downloaded to a target machine and viewed either through the ADMIT GUI or the Python toolkit interface.

4.1.2 Infrastructure Layer and Pipeline

The infrastructure of ADMIT consists of a CASA-compatible Python module of access routines and an XML-format metadata file. The XML metadata file, created in the BDP or by any re-running of the ADMIT script locally, contains information about the observational configuration, source, data statistics, and discovery information about the identifications of the lines in the observed bands, channels with detected lines, and data products. Simple routines access this metadata to retrieve information as needed for the current operation. For interfacing to CASA tasks, the metadata is used to construct XML parameter files which are used to run the task. Each ADMIT operation builds additional detail into the XML file creating a more comprehensive, scientifically useful description.

The ADMIT pipeline is a series of operations that create metadata then utilize the metadata to create higher level products. This approach is simple to expand to add further operations or to customize for specific applications. The default ADMIT pipeline consists of the summary information, noise levels, spectra, line identifications, moment maps, overlap intervals, and image saliency information. ADMIT can work with images in both FITS and CASA format. The outputs of the pipeline are wrapped into a single, self-describing tar file, which any ADMIT Tool can parse and manipulate. A typical tar file might contain a descriptive XML file and associated small FITS and PNG files.

Figure 4 shows a mock-up of the ADMIT initial screen as invoked on the local user computer. ADMIT next searches the local directory for ADMIT BDP files and displays a mock-up in Figure 5 indicating what it found and some instructions. Figure 6 shows a mock-up of how the summary data page could for a target source in the datasets that ADMIT found. The multiple tabs along the top are for different sources. The items down each column show types of data available, spectra, moment zero, one or two maps, etc, click on the data-cube image puts up a more detailed page for that item.

ADMIT runs at the ALMA pipeline-archive interface to create the initial BDP, but it can also be called by a user from within the CASA Python environment as individual commands or a script. This flexibility allows a user to tweak the ADMIT settings, add user functionality for a specific survey or project, or create more complete metadata. The infrastructure can encompass multiple projects and sources, and perform the same operations on each source. This can be done either in parallel or recursively. It can then extract information from each project and in essence mine a large suite of data, allowing linked data techniques to visualize the extracted information and provide new insight on what is common or different in the sources.

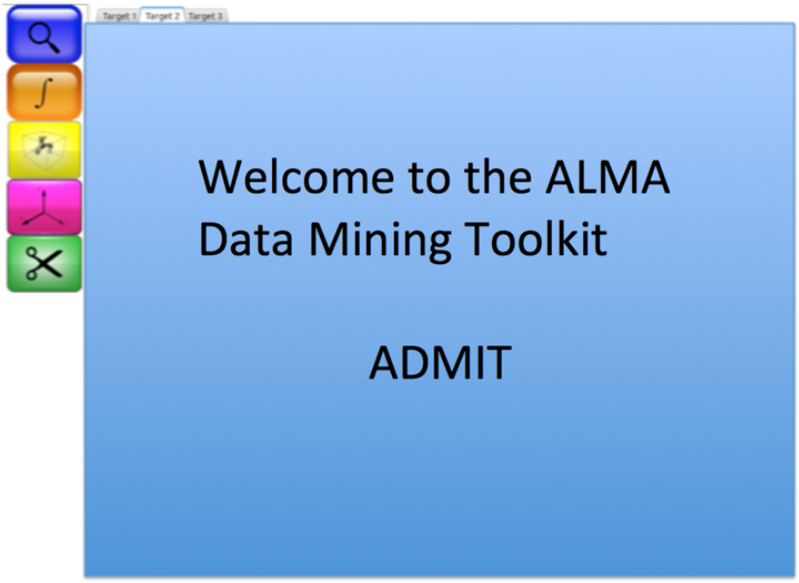

Figure 4: Mockup of the start screen of the ADMIT GUI (desktop application). On the left from top to bottom are buttons to invoke ADMIT Tools (Data Summary, Moment, Feature Extraction, Saliency, and Line Cube Cutout) for the target source selected in the middle panel. On the top row, tabs are visible for a number of collected sources.

4.1.3 The Toolkit

Once the core ADMIT components have created the XML metadata, ADMIT uses the toolkit component to explore the metadata and make it direct assessable the user (both novice and expert). This is possible because ADMIT defines a standard architecture that allows users to use existing analysis tools or to create and "plug in" their own analysis tools to ADMIT. ADMIT provides an API for these tools to call existing CASA commands, manipulate images or metadata, compute new quantities, return data structures to casapy, and store results in the tar file with new metadata

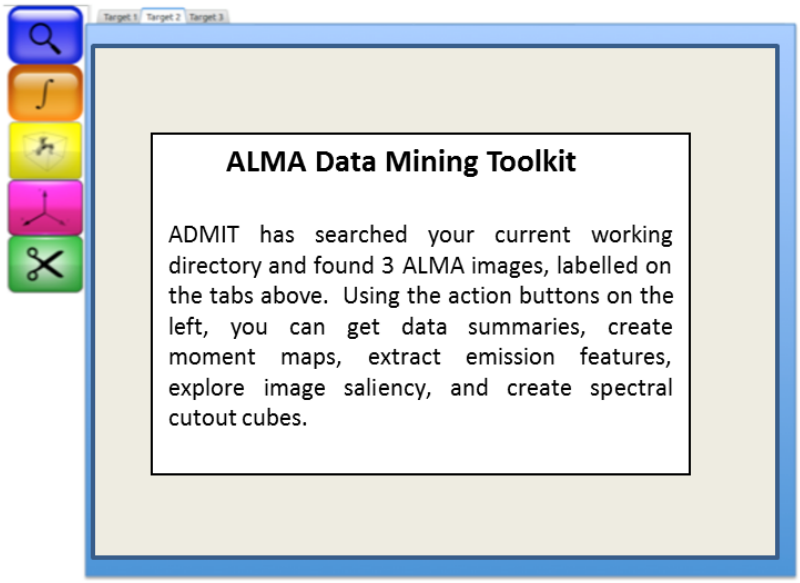

Figure 5: Mockup of screen of the ADMIT GUI (desktop application). ADMIT would automatically scan the local directory for ADMIT files and load them into the interface if found.

written to the XML. Below we described the initial set of tools accessed in the GUI that will typically be run as a desktop application (see Figure 4). These tools will deliver science mining applicable to a broad range of sources and encapsulate the best methodologies arrived at through the communitys many years of experience.

4.1.4 Data Summary

The Data Summary gives the scientist a quick, visual answer to the question: "Whats in my data?" ADMIT answers that question, but as a powerful addition, the Data Summary also provides the answer to that question at any point or step in the analysis process. ADMIT collects metadata and writes them to an XML file based on a set of ADMIT schema. The metadata are gathered from the image cube headers, any metadata produced by the ALMA pipeline, and computed quantities from the ADMIT pipeline. When run locally, the default is to gather the XML data from all ALMA image files in the current directory. Some components of these metadata contain project information (e.g., sources, sky positions, ALMA bands used), others may be computed values (e.g., image statistics such as min, max, mean, robust median, RMS per channel, etc.), still others will be determined by any previous workflow from ADMIT analyzes. Each ADMIT Tool operation will update the metadata so that an up-to-date summary of everything that has been gathered and computed is quickly presented (Figure 6). Selecting a particular band, line, or source would open up more detailed views of the BDP.

4.1.5 Example Use Case

In this use case scenario, the user has already downloaded a number of projects from the ALMA archive, and all have the basic ADMIT data included. The hypothesis to be checked is if the line width in CO correlates with the presence of a selected number of molecular lines, in the sense that for wider profiles some of the molecules would be destroyed, and thus the overlap integral would not achieve the maximum value (where all molecules are present). In this case the user selected

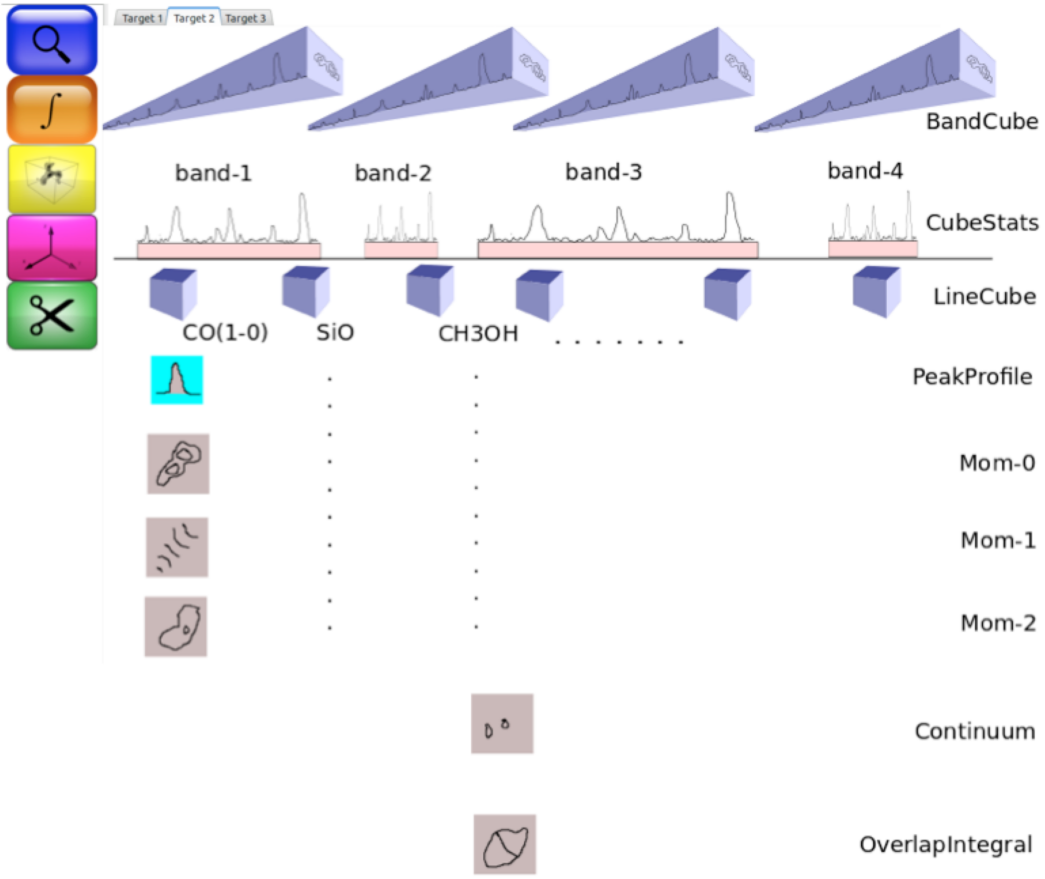

Figure 6: Mock-up of screen of the ADMIT GUI (desktop application). The data summary page give the user an overview of the data there were present for a given source. Each row show the output of an ADMIT tool operation listed on the right, giving an easy-to-understand visual summary of the science data and tool outputs for the target source selected in the middle panel. On the top row, tabs are visible for the collected sources.

CO as the reference line, and C17O, CS, and H13CN as probes. In this example the user is more interested in a global value per object, than a point by point comparison in the map.

The code loops over all projects, and first ensures that all lines are represented. It estimates the line width from the PeakProfile (precomputed by ADMIT) in the CO line cube as a representative value for the line width for the object. This particular user is conservative and wants to recompute the integrated flux (moment 0) maps with a 3-sigma clip. These new moment maps are then turned into an overlap map (one of the ADMIT tools) and via some simple NumPy math the fraction is computed from number of pixels with value 15 (where all lines are present) divided by the number of pixels with any signal in any of the lines (overlap map value more than 0). The line width is then plotted against this fraction, with the expectation that it starts near 1 and then suddenly drops at some line width. In the example code is shown below, one can see the power of ADMIT: in just a few lines of simple Python, a scientific hypothesis can be created and examined based on ALMA images.

```
import admin, math                                  # grab some needed python modules
import numpy as np                                  #

adm = admit.ADMIT()                                 # initialize ADMIT

projects = adm.query_dir('.')                       # look for ADMIT projects here

lines = ['co', 'c17o', 'cs', 'h13cn']               # first line is reference line
omax = math.pow(2,len(lines))-1                     # max in overlap map, 15 in this case

s_list = []                                         # accumulate line widths
f_list = []                                         # accumulate fractions

for p in projects:
  adm.setdir(p.dirname)                             # move into the proper project directory
  line_cubes = {}
  for c in p.linecubes:                             # loop over line cubes and grab the line
    if lines.count(c.line):                         # if we got one of the lines we wanted
       line_cubes[c.line] = c                       # store reference to that line cube

    if len(lines) != len(line_cubes):
       print "Skipping ",p.dirname
    continue

  c_ref   = line_cubes[lines[0]]                    # reference to the CO cube
  x       = c_ref.peakprofile('vlsr')               # spectrum X in KM/S
  y       = c_ref.peakprofile('value')              # spectrum Y in JY/BEAM
  x_mean  = (x*y).sum()/y.sum()
  x_width = (x*x*y).sum()/y.sum() - x_mean*x_mean

  s_list.append(x_width)                            # accumulate for later

  m = []                                            # accumulate maps for new overlap
  for l in lines:
    m0 = adm.moment(c,[0],'clip',rms=3*c.rms)       # compute new moment maps
    m.append(m0)

  o = adm.overlap(p,m)                              # get overlap image
  oval = o.array()                                  # get a numpy reference for work
  f_all = len(np.where(oval == omax)[0])
  f_sig = len(np.where(over > 0)[0])

  f_list.append(f_all/f_sig)
  #
```

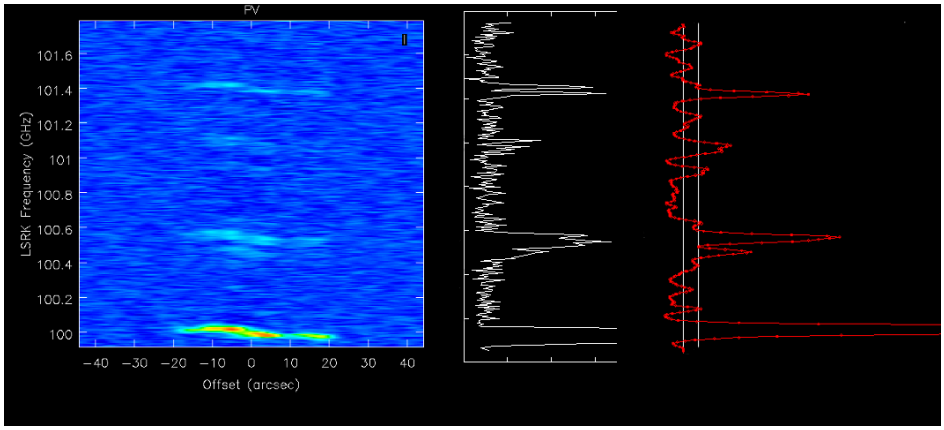

Figure 7: Line identification can be tremendously improved in a position-velocity diagram by cross-correlating the signal in the velocity direction. In Figure 7, compare the more traditional noisy peak/RMS plot in the middle (white) with the smooth cross correlation technique on the right (in red). The lines are better resolved using the cross-correlation technique. The two vertical lines denote zero (white line) and a 3-sigma clipping (gray line).

```
#
adm.plot2d(s_list,f_list)                    # scatterplot of accumulated data
```

5 ADMIT Tools

This section shows some of the tools that we have been prototyping for ADMIT. The tool are layered on top of the XML infrastructure of ADMIT and communicate through that sturcture for inputs and outputs. We envision that there will be a set of basic tools that create some of the Basic Data Products and there will be an evolving set of tool that add new capability. The tools will be compatable with the CASA environment and utilize the CASA core and existing CASA routines where applicable. The tools will operate on CASA image format data.

A second operational goal for tools is that they can be re-executed based on data in the ADMIT XML files and the XML files can be modified to allow the tools to be re-run in bulk. For example, if you decide to change the velocity range for a moment map, you could modify the XML information and then re-run the moments for all of the lines of interest with with a single command.

5.1 Line Identification and Line Cutout

The Line ID Tool will compare the rest frequency coverage of the image cubes with a database of common line retrieved from Splatalogue to identify the appropriate channel/cube locations of potential lines. A line-strength vs. frequency table is the essential input into a line identification procedure (here we will leverage Tony Remijans datasplat project). The input data for the procedure will be improved by the previously computed per-channel cube statistics and aided by cross-correlation techniques in a position-velocity diagram (Figure 7) or in the cube itself. The output will be a series of identified and unidentified lines across the different bands in a simple table of line, frequency, channel range, and detection probability.

Based on the line identification line tables, line cutout cubes can be extracted, with now the third axis in Doppler velocity space, to ensure that we can compare the different lines and molecules on

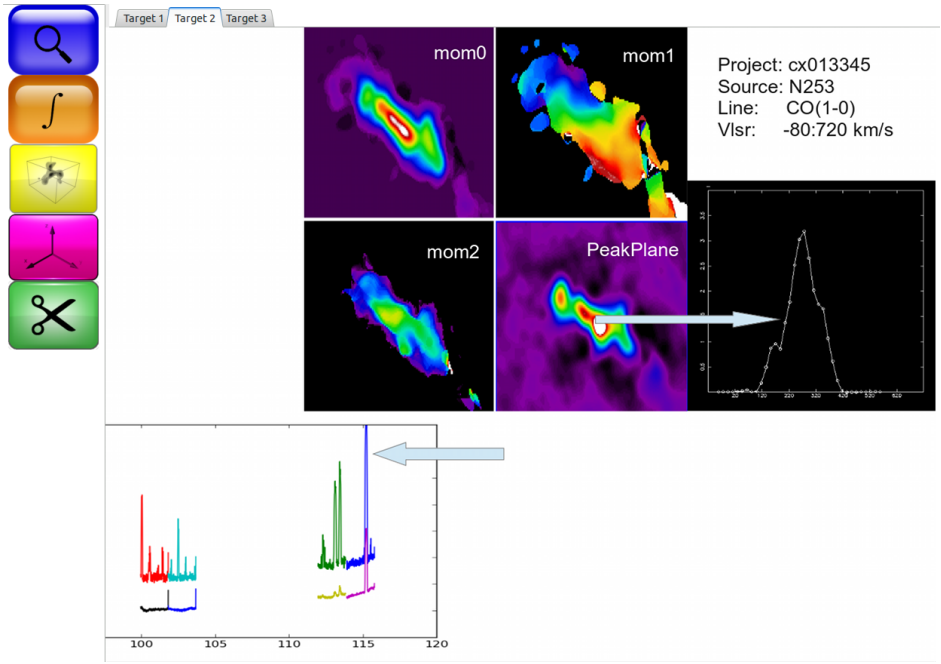

Figure 8: Mockup of the detailed view of some of the data products of a single source and spectral line, as produced by the Data Summary tool for the target source selected in the middle panel. In this scenario, the user has clicked on the CO data cube icon in Figure 6. On the top row, tabs are visible for a number of collected sources. The lower arrow indicates which line is the CO line in the full-band spectra. The upper arrow indicates the reference position for the spectrum shown.

the same basis. Depending on the number of lines in the bands, keeping only the line cubes can significantly cut down user disk usage.

5.2 Moment Maps

Moment maps are a simple, yet powerful tool for examining global properties of an objects emission. The most commonly used maps are integrated flux (moment 0), centroid velocity (moment 1), and velocity dispersion (moment 2). The Moment Map tool takes the line cubes produced by the Line ID Tool and creates the three moment maps for each spectral line (see Figure 8). The moment can be clipped based on simple RMS cutoff (already computed by the Data Summary tool!), but also could employ more sophisticated methods such as local smoothing to increase the signal to noise to create a more robust clipping mask. Moment maps can also be produced using parametric shapes, e.g. a Gaussian fit, which would then store the total emission, mean velocity and width of the spectrum at each location.

The advantage of the ADMIT approach is that the previous steps of line definition and rest frequency assignment allow automated calculation of the velocity scale for each line. This can be combined with simple spectra at peak position as shown in the mock-up figure.

6 Exploration of New Algorithms

To enable the creation of an interactive analysis, reasoning, and discovery environment, new methodologies are needed to detect and effectively presentation the scientifically interesting data buried in large ALMA data cubes. The fundamental fact is that most of the pixels in any data cube

are noise, and data cubes with significant signal density (many lines and/or many spatial features) are difficult to mine for science content.

Much of the science with large data cubes to date has been accomplished with simple information compression techniques: moment maps, velocity position diagram along selected axes, rotation curve fitting, and visual inspection for eye-catching correlations. Analysis such as principle component analysis and other orthogonal component decomposition analysis have been applied for analyzing structure in complex environments such as molecular cloud. In general, however, the generally available techniques in standard radio astronomy data reduction packages have not change in major ways in over 20 years.

As a part of this study, we have looked at the kinds of new analysis tools will be of high value to astronomy for: (1) comparing the characteristic of emission within a given line at different velocities, in different lines, for the same line in different sources. The subsections below discuss two specific examples that we explored at the level of creating prototype tools.

6.1 Overlap Integral

The overlap integral provides an overview of the emission properties across a number of spectral lines; it can also be generalized to apply to the emission as a function of velocity within a line or across source. In the multi-line application, for each detected line, one assigns a one or zero indicating the presence of emission at some selected level, and then logically ORs them across all lines, creating a bit-mask map of the emission present in each spatial-velocity pixel. One can do this in a moment map, or a cube, to determine in which spatial or velocity regions certain lines are present or absent in relation to other lines.

The final image or cube is a integer map with 0/1 in each bit representing the presence/lack of emission in one of the lines. This integer is then easily displayed as a color image. An example using NGC 253 spectral line data from ALMA Cycle 0 is shown in Figure 9. This technique can also be applied to individual channels of a single spectral line to examine the velocity pattern of emission, similar to a renzogram.

6.2 Descriptor Vectors

One technique from computer science with significant potential for implementation in astronomy is ”descriptor vectors”. The approach is to segment an image or cube and and calculate an n-dimensional description vector to characterize each segment. One of the current applications of this technique in visual image analysis is in automated feature identification. In this application, the image is segmented into M-regions and each region is characterized by an n-dimensional vector which might simply the intensity distribution of the pixels within the segment, or the pixel color distribution, or contrast level; it could be that the segment is characterized by the spatial shape distribution. Once the vectors are generated, then it is easy to calculate the ”distance” between the various vectors and to identify which vectors are most like, or unlike, each other. One can identify clusters of vectors. This gives you an ordered list of locations in the image or cube that share similar characteristics.

The descriptor vector approach provides an application-independent, purely mathematical measurement of information (M. Chen and H. Janicke. “An information-theoretic framework for visualization” IEEE Transactions on Visualization and Computer Graphics, 16(6):1206 1215, 2010;

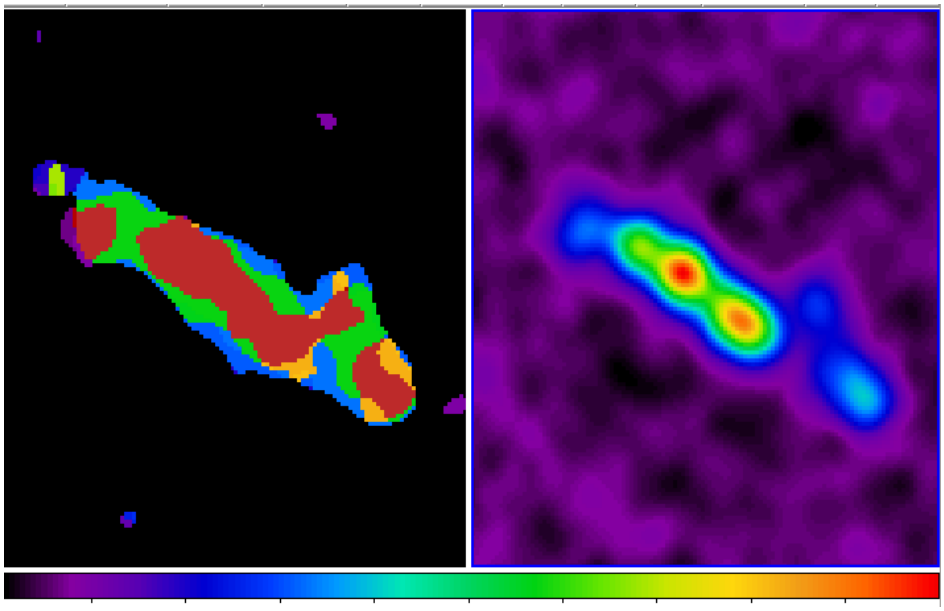

Figure 9: An example of an overlap integral using ALMA Cycle 0 data of NGC 253. Right: Integrated flux (moment zero) image of the CO J=1-0 emission. Left: The combined overlap integral – valued from 0 to 255 – of 8 spectral lines. Each line is represented by a single bit in an 8-bit integer which is then translated into a color scale. As can be seen, most of the peaks have all 8 lines present (red). The spectral lines used to create this overlap integral are CO, $C^{17}O$ J=1-0, CN J=1-0 F=1, CN J=1-0 F=2, HC_3N, CH_3OH, H_2CS, and CH_3C_2H.

H. Janicke and M. Chen. "A salience-based quality metric for visualization" Computer Graphics Forum, 29(3):11831192, 2010). For astronomical images, description vectors can be chosen from any number of properties of the emission. An example in vision research utilizes a color histogram, which enables a very fast cataloging of feature types in an image, as well as finding the features similar to a selected one, a type of machine learning (C. Y. Ip and A. Varshney, "Saliency-assisted navigation of very large landscape images", IEEE Transactions on Visualization and Computer Graphics, 17(12):1737 1746, 2011; C.Y. Ip, Ph.D. Thesis, University of Maryland, 2014). We have imported several astronomical data cubed into existing descriptor vector software associated with the work by Mr. Ip (UMD computer science graduate student) and Professor Amitabh Varshney (UMD Professor in computer science). One of the aspects of the descriptor vector method is that you have a set mechanism for how to proceed but you have wide latitude in how you segment the image and the mathematical prescription for calculating the n-dimensional vector. This makes the method very versatile.

It is important to note that although this may sound similar to Principal Component Analysis, the descriptive vectors do not need to be orthogonal or even have the same units. Because of the versatility and the specific focus possible with tuned vector formulations, data mining with this technique can be an extremely powerful, allowing a fast comparison of features that might be missed otherwise.

During our study, we evaluated this techniques potential for achieving astronomical science goals. The key to success is discovering the best description vector for the specific science case (M. Chen and H. Janicke. "An information-theoretic framework for visualization", IEEE Transactions

on Visualization and Computer Graphics, 16(6):1206 1215, 2010). For example, in one case the descriptive vector that best describes a specific science goal may be the properties of the certain molecular lines and the overlap interval, while another may be the location of the molecular peaks with respect to the continuum. ADMIT can provide the infrastructure to write various descriptor vectors, provide some of the vectors (based on BDPs) that will most likely to cover standard science goals, and provide a task that determines best vector choices based on user inputs. We will also document the descriptor formulation procedure so that users have the option to build on this approach in the future.

There are, of course, no significant barriers to generalizing this approach to comparing multiple lines or multiple sources. For example to be able to ask questions like: Where in my data cubes to HCN and CO have overlapping emission but there is no HNC emission? Give me a list of all of the position-position-velocity locations were SO and SiO emission are coming from the same locations. Or, give me a list of the places where the CO is falling off rapidly with velocity and there is SO emission.

6.3 Learning Algorithms

In computer science applications, the descriptor vector characterization of an image can be layered with learning algorithms which can power efficient data mining. The simplest application is to make a descriptor vector description of an image or cube then to manual look at the image or cube and identity a region of interest. The algorithm can then calculate which other segments have similar vectors to your selected regions, providing an ordered list of the 10 nearest vectors. One can then view those regions, select which ones are correctly of interest; feed the new set of regions if interest into the algorithm and get a new ordered list of most similar vectors/regions.

This combination of compact description approach and learning-selection is very well suited for combination with a visualization tool. The algorithms here provide a computing layer which provide an order list of interested regions in the cube for display in the visualization tool. The visualization give the user rapid feedback to refine the search; the list of preferred regions is feed back into the algorithm for an increasingly targeted search.

7 Intercomparison of Model and Observational Data

As the study progressed, we felt that this area was of less importance than continuing to development the XML structure design for ADMIT. By adopting the CASA and its python environment, we will be able to adopt python based program for reading output from modeling programs and simulations into a common format. During the course of our study, we also learned that CASA was in the process of evaluating its approach to the python environment. Changes there could make a wider range of existing programs easily available for use.

Benchmarking CASA Image Cube Access

Peter Teuben

September, 2013

We conducted a modest benchmark to gain insight in image I/O patterns and efficiency of CASA compared to MIRIAD, and raw FORtran The first order goal of this benchmarking is to understand if the current CASA data access model is reasonably fast for general access. MIRIAD is the standard analysis package for CARMA. A specialized benchmark program was written in Fortran to accomplish the same tasks as being preformed in CASA and MIRIAD. The purpose of the "raw FORtran" program is to make an effort to see the "best possible" speed for each task. In the case of the FITS task, the raw FORtran program reads and writes a MIRIAD format file to simulate the I/O of the FITS read. The raw FORtran program keeps all arrays in memory. For these tests, we used version 4.1.0 (r24668) of CASA. These test utilize 3-dimensional cubes because they are the most common in ALMA data.

CASA uses a tiled storage manager, which in principle can be tuned to optimize access speed for the specific cube dimensions. Currently, CASA does not actively tune the tiling so these tests are done with CASA's standard tiling (1/32 of an axis length). MIRIAD has a traditional row based access; it rarely uses memory to store full planes or cubes. The Raw FORtran program keeps the whole cube in memory.

Three datasets of 4GB (10^9 pixels)[1] in size were used: a 1024x1024x1024 true data "cube", a 4096x4096x64 3-D "slab" and a 128x128x65536 3-D "stick". Four operations were considered: reading from fits into the native package format, a simple rms/mean statistics that visits the whole cube in the most efficient way, adding two cubes, and finally a Hanning smooth in X, Y and Z to stress access. For MIRIAD the Hanning smooth test had to be skipped in X and Y, unless an additional `reorder` step would be allowed, which we did not consider in this version of the benchmark. It should be added that a true Hanning in MIRIAD is the slowest possible operation, since the data is stored row-wise.

The tests were performed on a 3.6GHz i7-3820 CPU (10MB cache) with 64 GB of memory utilizing 1 core. Raw disk I/O (using `hdparm -t`) was measured at 800 MB/sec. Times reported are the wall clock time on an idle system. We report wall clock times because that is most relevant to the user experience and efficiency. Wall clock time often differs from standard measures of cpu usage, even when the system is 100% active on your task because linux operation

[1] 4GB also happens to be a good test to confirm there are no 32bit code issues left

system can do some disk I/O as system time without reporting it directly as CPU usage.

On a typical Linux workstation, disk space is cached to memory. On machines where there is at least several times more memory available than the typical size of a datacube, this system feature can affect a pipelined benchmark such as the one given here by using the disk cache in memory and thus report shorter processing times. We therefore made sure for each operation had no disk cache left[2].

	Cube 1024x1024x1024			Slab 4096x4096x64			Stick 128x128x65536		
	CASA	MIRIAD	FOR	CASA	MIRIAD	FOR	CASA	MIRIAD	FOR
FITS	16.5	12.2	9.6	14.7	12.4	9.6	26.1	14.7	9.8
STATS	17.1	14.0	9.8	17.7	13.8	7.2	17.9	15.2	7.3
MATH	11.0	16.7	10.4	11.2	12.4	10.4	11.0	9.5	10.5
HAN-x	15.6	n/a	12.3	12.6	n/a	12.2	40.9	n/a	12.7
HAN-y	16.8	n/a	16.5	29.7	n/a	37.0	51.3	n/a	13.6
HAN-z	24.8	112.0	61.3	72.0	102.5	24.1	18.4	137.0	40.8
sum-1	118.9	(133.2)	116.0	157.9	(138.2)	100.0	165.6	(179.0)	106.4
sum-2	109.3	(124.0)	95.3	139.9	(138.2)	77.5	142.5	(187.3)	74.0

Table 1: Comparing I/O access in a "cube", "slab" and "stick" like dataset. Times reported are the wall clock elapsed time in seconds. NOTE that the MIRIAD column sums should not be compared to the CASA and Fortran sums because MIRIAD did not include HAN-x and HAN-y.

The individual tests listed above are: (1) a FITS (BITPIX = -32) read using the native program for each system; (2) computing the mean and rms of the cube; (3) a mathematical operation, we added two (the same) cubes and wrote out the sum; (4) Hanning smooth in the X direction; (5) Hanning smooth in the Y direction; (6) Hanning smooth in the Z direction; The *sum-1* line in the table is a sum of the individual processes in the table above that line; *sum-2* is the total time of these processes are run as a single pipeline step, i.e without clearing cache between processes.

CASA performed as well as, or better than, MIRIAD and FORtran on the majority of tests. The tiled access in CASA performs well for true cubes, fairly independent of the access direction. The "stick" cube presented the greatest challenge to CASA with the current fixed tiling. FORtran's memory based model showed advantages in some operations, but not all. MIRIAD show a serious flaw in the Hanning-Z test due to its underlying access pattern.

Overall, we conclude that CASA is efficient enough at general data access that it does not make sense for the purpose of our current study to pursue alternative data storage and access models. We actually conducted this test near the start and near the end of our study, reaching the same conclusion both

[2]the command `"echo 1 > /proc/sys/vm/drop_caches"`, from root, was used for this.

times. The numbers reported in the table are from benchmarking in September 2013.

The system behavior seen during our benchmarking implies that running a pipeline on a dedicated machine with plenty of memory can be advantageous, as cached data is more likely to be reused.

NRAO Development Study Proposal: ADMIT Preliminary Design

Peter Teuben, Marc Pound, Doug Friedel, Lee Mundy, Leslie Looney

September 20, 2013

1 Overview

The ALMA Data Mining Toolkit (ADMIT) is a value-added software package which integrates with the ALMA archive and CASA to provide scientists with quick access to traditional science data products such as moment maps, and with new innovative tools for exploring data cubes. The goals of the package are to (1) make the scientific value of ALMA data more immediate to all users, (2) create an analysis infrastructure that allows users to build new tools, (3) provide new types of tools for mining the science in ALMA data, and (4) increase the scientific value of the rich data archive that ALMA is creating.

For each ALMA science project a set of science quality image cubes exist. ADMIT runs a series of "ADMIT tasks," which are essentially beefed up CASA tools/tasks, and produces a set of Basic Data Products (BDP). ADMIT provides a wrapper around these tasks for use within the ALMA pipeline and to have a persistent state for re-execution later on by the end user in ADMIT's own pipeline. ADMIT products are contained in a compressed archive file *admit.zip*, in parallel with the existing Alma Data Products (ADP)[1].

Once users have downloaded the ADMIT files, they can preview the work the ALMA pipeline has created, without the immediate need to download the much larger ADP. They will also be able to re-run selected portions of the ADMIT pipeline from either the (casapy) commandline, or a (CASA) GUI, and compare and improve upon the pipeline-produced results. For this some of the ADP's may be needed.

ADMIT introduces the concept of a "Virtual Project," which is a framework for a collection of similar ALMA projects. ADMIT is used to manage tasks run across all ALMA projects in the Virtual Project, after which selected results can be data-mined and correlated.

Below, we first outline use cases driving the ADMIT design and the resulting required functionality and products. Then we lay out the core XML components of ADMIT, describe the structure of Basic Data Products and tools, and conceptualize the software layers and interfaces. Finally we outline the requirements of

[1]Examples of ADP in a project are the Raw Visibility Data (in ASDM format) and the Science Data Cubes (in FITS format) for each source and each band

the ADMIT GUI, design and development of which is staged after the ADMIT XML and Python are in place. In the Appendices, we give greater detail on proposed XML structures.

2 Use Cases

2.1 Archive

This case covers a principal goal for ADMIT: produce a small file in the ALMA Archive that users can quickly download to inspect their data without needing to download the large full data cubes. ADMIT should produce in the ALMA pipeline a single, compressed file, `admit.zip` for a given project. A project contains one or more sources, and for each source the ALMA Archive will create four image data cubes, in FITS format. From these cubes, the ADMIT pipeline, running on the ALMA Archive, will produce Basic Data Products summarizing the observations and a descriptive `admit.xml` file and add this as `admit.zip` to the archive. The BDPs created for data inspection will be

1. A summary of the observations (source information, band information)
2. A list of detected spectral lines in each band (name, rest frequency, V_{LSR}, detection probability)
3. A small datacube centered in velocity around each detected spectral line.
4. Zeroth, first, and second moment maps for each detected spectral line
5. For each detected spectral line, a spatially-averaged spectrum through the peak in the zeroth moment map.
6. For each plane in each band cube created by the ALMA archive, a simple table of statistics: minimum, maximum, mean, median.
7. An overlap integral map that shows a combined view of zeroth moment maps of each spectral line.
8. Where applicable, a continuum map made from emission-free channels.

2.2 First Look

The user that has gone to his/her project on the archive, and downloaded the `admit.zip` file. The ADMIT environment in casapy will then either print to screen, or use a GUI, outlining all BDPs present, and allow looping over and visualizing them via Python methods. There is no essential computing needed in this step.

2.3 Rerun (Sections of) Pipeline

After a first look, the user may wish to modify the BDPs, for instance, by adding spectral line to the line catalogue (LineList), recreating moment maps based on a lower clip level or by a different method of computing moments. The input methods and parameters to compute a BDP can be changed, through by or, in later ADMIT versions, through a GUI.

2.4 Morphological Analysis

The user wish to make use of advanced data analysis techniques to discover structure in a source or compare structure across maps of different spectral lines. This could be done by providing an interface to invoke standard algorithms such as *clumpfind* or dendrograms. A new algorithm we propose to develop, taken from computer science visualization techniques, is using *description vectors* to determine image saliency. This provides an application-independent, purely mathematical measurement of information. For astronomical images, description vectors can be chosen from any number of properties of the emission. Since these algorithms are resource intensive, and not all users may want them, the associated BDPs would not be computed in the ALMA archive pipeline.

2.5 Virtual Projects

In this more advanced use case, the user selects a number of sources from one or more projects in a virtual ADMIT container. Any of the basic ADMIT procedures can now be re-run and/or reviewed, but looped over the selected sources. Users can write their own procedures, and store persistent data back into `admit.zip`. ADMIT has interfaces using Python to extract numbers and Python arrays out of `admit.zip`, and thereby allow for flexible data mining, linked data plotting, etc.

3 Basic Data Products

The pipeline results and outputs of ADMIT tasks are described as Basic Data Products. A BDP may contain computed or harvested information as well as point to an external resource, such as an image file. We expect for all BDPs there will be a one to one correspondence with an ADMIT task. All BDP instances will have the following structure:

```
BDP
  name
  data element 1
  data element 2
  ...
  data element N
  task
```

`dependencies`

where *name* is the descriptive name ("summary," "linelist"), *data elements* are harvested information, computed values, tables, or pointers to images (with a True/False value if that image is exported to `admit.zip` as well), *task* is the ADMIT/CASA task used to create this BDP with all all task parameter values specified, and *dependencies* are other BDPs upon which this one depends. If a dependency BDP for this BDP changes, then this BDP should be recomputed. For instance, the overlap integral depends on moment maps, so if the moment maps are recomputed the overlap integral should be as well.

4 XML Structure and Types

The XML structure should be complete enough to capture proposed use cases, but not so restrictive that it precludes future use cases. To that end, a limited hierarchy is desirable, with basic "objects" that can be contained within one another but are not *defined* within one another. However, we do need to follow what we understand to be the basic hierarchical structure that the archive produces Project→ Source → Band Cube → Line Cube. We argue it is not necessary to expose the Project container to the user, e.g. in the case of a data mining operation that spans multiple projects. The science is in the sources not the project. To that end, the Project container is extremely thin: it contains only the identificiation code, everything else is inside source containers.

The following list describes the basic XML types from which `admit.xml` and the Basic Data Products may be composed. Note all tables will be stored in VOTable format. Details of the XML structure of each are given in Appendix B.

- **Project** – This is a thin structure giving the name of the project and containing one or more Sources.
- **Source** – This container describes the source parameters (name, coordinates, etc) and contains all BDPs for this source.
- **Image** – An image reference, typically FITS or PNG. The image data file itself is normally not contained inside the XML, but external on disk.
- **Spectrum** – A one-dimensional spectrum through a data cube at a given location.
- **Statistics** – A summary of statistics computed for an image or spectrum
- **Task** – A full description of the task used to create a particular BDP. This includes task name and all parameters with which the task was invoked.
- **Dependencies** – A list of BDPs upon which a particular BDP depends.
- **Date** – The date and time when a particular BDP was computed.

Using these types, the `admit.xml` and BDP XML definitions take shape:

```
<project name="c1234_abcde_hijkl">
<source name="abcde">
<BDPName type="bdp">
   <element1>
   <element2>
   ...
   <elementN>
   <task>
   <dependencies>
</BDPName>
...
<BDPName type="bdp">
   <element1>
   <element2>
   ...
   <elementN>
   <task>
   <dependencies>
</BDPName>
</source>
</project>
```

As a detailed BDP example, a first moment map BDP might look like:

```
<moment type="bdp">
   <integral>1</integral>
   <description>velocity centroid</description>
   <method>clip</method>
   <image>...</image>  % e.g. the FITS image
   <image>...</image>  % e.g. the PNG image
   <task type="function" name="moment" category="imagefu">
        <param type="float" name="clip">
           <value>0.1</value>
        </param>
        <param type="float" name="vmin">
           <value> -123.0</value>
        </param>
        <param type="float" name="vmax">
           <value> 234.0</value>
        </param>
   </task>
   <dependencies>linecube</dependencies>
   <date>YYYY-MM-DDTHH:MM:SS</date>
</moment>
```

4.1 Virtual Project

A virtual project is a user or ADMIT defined structure that can span multiple projects, sources, and bands. The members of the virtual project can be grouped into one or more subgroups that can be processed in similar manners or to a similar depth.

The virtual projects defined by the user will have their own XML structure, not saved in the `admit.zip` file(s), but in a separate XML file. This file will

have links to the members of the virtual project. The virtual project will contain nodes for dividing up the individual parts so that they can be grouped by the user or ADMIT, for further analysis. A sample XML structure follows:

```
<project type="virtual">
    <group name="name1">
        <common item: cutoff, molecule, etc>
        <member>
            <file name="file name"/>
            <id name=""/>
        </member>
        ...
    </group>
    ...
</project>
```

Each source/band should have a unique ID (e.g. project-source-band) so that the virtual projects can link to all associated data. The link would include both the absolute file name and path and the ID for each member of the virtual project. Each time the virtual project is opened these links should be followed and verified (possibly loading the data also). If a file is not where it is expected the user will be prompted to locate the file again. The links would be like the Unix hard link, i.e. it will point to the original data/images unless the analysis is redone, at which point there will be a local version (in the virtual project) of the products, leaving the originals intact for comparison. There will also be an option for the user to package up the associated parts into its own `admit.zip` file so that it is portable. The virtual projects will retain the same project-source-band structure as the typical project so that processing in ADMIT and other programs is straight forward.

4.1.1 Virtual Project Example

There are two types of `admit.zip` files: the ones belonging to a genuine ALMA archive project, e.g. `c0123_admit.zip` and the ones belonging to a virtual project, e.g. `test1_admit.zip`. Virtual Projects inherit their projects by virtue of a hard link. Let's say `c0001`, `c0012`, and `c0123` are three projects that are in a virtual project, called `test1`, the directory `test1` will contain hard links the directories `c0001`, `c0012`, and `c0123`, and the sub-directories needed by virtual of their selection. This way, if in the virtual project one of the BDPs has changed, they overwrite their original version. [2]

Assume the user has downloaded a few ADMIT files, for projects `p1` and `p2` and `p1` has two sources, `s1` and `s2`.

```
% ls
  p1_admit.zip
  p2_admit.zip
% admit --extract p*zip
```

[2] Linux command: `cp -al`, Mac command: `pax -rwl`. Or `rsync`

```
p1/admit.xml              (via admit)
   p1.s1.b1.fits          (only present from archive if downloaded)
   p1.s1.b2.fits          (unique names containing P, S and B)
   p1.s1.b3.fits
   p1.s1.b4.fits
   p1.s2.b1.fits
   p1.s2.b2.fits
   p1.s2.b3.fits
   p1.s2.b4.fits
   s1/l1/l1_cube.im       (a recomputed casa cube)
         mom0.fits        (fits and/or jpg, via admit)
         mom1.fits
         mom2.fits
      l2/l2_cube.im
      ...
      overlap1.jpg        (overlap intergral from all Lines)
      ...
   s2/
   ...
p2/admit.xml
   ...
...
```

5 Interfaces

Since CASA interfaces are all in Python, most – if not all – of ADMIT will be in Python. Some adaptations to CASA routines will likely be needed, and this may invoke some changes to the CASA C++ core code. A simple example we encountered is a robust median, which is not in CASA core but is quite useful in computing statistics. We envision the ADMIT software structure to be three layers: the base layer encapsulates all XML operations, the middle layer is the ADMIT pipeline infrastructure and task interface, and the top layer is the user interface.

5.1 XML I/O and Manipulation Layer

This layer is to standardize the interaction with the XML for all higher level routines and to enable and simplity the XML stucture that we decide upon in the XML definitions. A choice between SAX and DOM parsing of XML will have to be made. SAX appears to be the favorite, based on much smaller memory footprint and more flexible parsing. The end user will not have access to this layer.

5.2 Pipeline Infrastructure and Tasks

The basic workflow of ADMIT is to open a project, compute a set of tasks, in a serial fashion, and close the project. This workflow is is the same regardless if ADMIT is run in the ALMA archive or by an end-user. Especially in the latter case, decisions are made based on interactive use with the pipeline. In

particular it will be important to be able to compare Basic Data Products of different invocations of an ADMIT task, for example, comparing the velocity field of a line using two different methods of computing the moments.

The following is the required pipeline functionality.

1. Opening a (single) project that is not ADMIT-enabled yet. There is no `admit.zip` present, so everything needs to be initialized: find the sources, the number of bands per source, and the variables describing each source/band (e.g. RA,DEC,vlsr,freq ranges etc.). The ADMIT XML needs to be created. No other tasks will be run in the pipeline, although this summary could be seen as the default initialization task.

2. (Re)opening from an existing XML. All structures will need to be intialized. Status and Dependency list of the various data products need to be set. Missing ADP and BDP are to be identified.

3. Opening from an existing ZIP. This is very similar to the previous item, but must also create a directory structure hierarchy and populate basic data products.

4. (Re)compute a Basic Data Product. The dependencies need to be reviewed (multiple are possible), as well as allow to bypass recomputing these if not desired. The method needs to be selected, parameters for this method set, as well as the style of execution When all is set, the task can be executed, and the BDP is created. All of this is wrapped in the `task` XML tag, compatible with the CASA `task` XML tag for inclusion into CASA.

5. Save a project: either the ADMIT Python object is serialized into XML, or in addition all associated allowed basic data products are wrapped in a single `admit.zip` file. Not all data products are normally wrapped in `admit.zip`. For example spectral line cubes are not, given their size. Moment maps have both PNG and FITS versions, both could be exported.

Another part of the "middleware" are the base class definitions of an ADMIT task and a Basic Data Product. We expect the ADMIT task base class will follow the CASA task structure closely if not identically. Using this class, all ADMIT tasks can be constructed and serialized to XML. The ADMIT task base class will be flexible enough to allow user-defined tasks as a future enhancement. Similarly, the Basic Data Product base class is the foundation of all BDPs and is serializable to XML.

5.3 User Interface

This is the layer that enables the user interaction with CASA and the pipeline. The initial user interface will be purely through Python/CASA commands. In addition to Python interfaces for all ADMIT tasks, We will provide shortcut commands for common operations, e.g., displaying a summary page.

A graphical user interface is anticipated to guide users through using ADMIT in an intuitive way. As a project is opened, a tabbed browser is display with for each source presented in a separate tab. This will for instance give an overview of cube statistics, a PV diagram, moment maps, lines detected. Any of the BDP can be recomputed with a pop-up parameter editor. Selections on projects/sources/lines can be made and grouped into a virtual project for re-processing these data in a common way.

6 Specific ADMIT Tasks

Here we describe a number of core ADMIT tasks in more detail.

6.1 CubeStats

The primary idea of CubeStats is to provide initial guidance for line identification. For each channel it tabulates the min, max, RMS, mean and median. A Robust Median is preferred, but is computationally expensive. (A 256x256x2048 cube requires 5 minutes as opposed to 5 seconds for a standard median).

6.2 LineList

A LineList is a simple table, tabulating which lines are present in the cube, and a likelihood the line is present. The likelihood is computed as *Signal/(Signal+Noise)*, thus is always between between 0 (unlikely) and 1 (likely). Based on more advanced concepts (e.g. PVCorr), a revised line list could be derived, resulting in a better line list source detections algorithms get more advanced in the pipeline.

6.3 PVCorr

In this task a position-velocity diagram is used more accurately determine the frequencies of detected lines. To create a good PV diagram, a position angle is needed, which can be obtained by summing up all emission above some value (e.g., a few times the RMS noise) and using a moment of inertia in the resulting summed emission map. When the axis ratio of this emission is round enough, an XYVcorr might give better S/N. The strongest line defines an area that is then cross-correlated in the V (frequency) direction and this defines a signal and noise within which line identification can take place.

6.4 XYVcorr

This is the 3D extension of the 2D PVcorr method. In theory should be more sensitive/accurate, if the emission is really not along some major axis. This has not been well tested.

6.5 Moment

Computing the moments of a line cube (really: deriving the total emission, the mean velocity and velocity dispersion) can be done with a number of methods, ranging from simple moments along the velocity axis, perhaps aided by masks generated from convolved data, to fitting model spectra. All of these are encapsulated.

6.6 FeatureList

A method has to be selected that assigns features (blobs, clumps, dendograms) to a line cube. For non-overlapping features, a simple table will suffice, for embedded features (e.g. dendogram), a hierarchical table will be needed. This latter model is not one of the basic data types (tables, image/cubes) that we support, so a new table type might be needed here.

6.7 DescriptionVector

A "DV" can be operated in in a line cube (or even a band cube) by assigning such a vector belonging to the features in the cube. But it can also be spanning multiple lines, in each line cube computing a vector that is now spanning lines, instead of features. There can be a Description Vector per Line Cube, but also per source, or even per (virtual) project. For example, a Description Vector in a Line Cube could be Feature based (e.g., the moments of inertia) or across different Lines, much like an Overlap Integral (cf. PCA), where they represent a multi-dimensional "color" of a feature.

7 Summary and Future Work

: WE NEED SOME KIND OF WRAP-UP SUMMARY HERE

A ADMIT Tree Overview

Without the full, cumbersome to read XML syntax, here is an overview of the tree that `admit.zip` will contain. The `[N]` notation means this item will occur `N` times at this level, for example for ALMA data you would see `NB=4`.

```
Project(name)[NP]
   Summary
      <atask name=at_summary>
   Source(name)[NS]
      Summary
         ra,dec,vlsr,...
         <atask name=at_summary>
      Band(number)[NB]
         URI:im
         Summary
```

```
      FreqMin,FreqMax,FreqStep
   CubeStats
       VoTable:tab
       <atask name=at_cubestats>
       <dep>
          URI:im
   PosVelSlice
       URI:im
       file:jpg
       <atask name=at_pv>
       <dep>
          URI:im
   LineList
       VoTable:tab
       <atask name=at_band2line>
       <dep>
          CubeStats
LineList
    votable:tab
    <atask name=at_linemerge>
    <dep>
       band[NB].LineList
Continuum(name)
    URI:im
    file.jpg
    <atask name=at_continuum>
    <dep>
       Band(this)
Line(name)[NL]
    LineCube
       URI:im
       <atask name=at_reframe>
       <dep>
          LineList
    RMS (since cubestats can differ per channel)
    Mom0
       URI:im
       file:jpg
       <atask name=at_moment>
       <dep>
          LineCube
    Mom1
    Mom2
    PeakSpectrum
       VoTable:tab
       <summary>
          Peak, RMS, V0, FWHM, SdV
       <atask name=at_spectrum>
       <dep>
          LineCube
    IntegratedSpectrum
       VoTable:tab
       <summary>
          Peak, RMS, V0, FWHM, SdV
       <atask name=at_spectrum>
       <dep>
          LineCube
```

```
        FeatureList
           VoTable:tab
           <atask name=at_feature>
           <dep>
              LineCube
        DescriptionVector
           VoTable:tab
    DescriptionVector
        VoTable:tab
DescriptionVector
    VoTable:tab
```

B Basic XML Types

.

All XML tags will be lower case; no mixed case tags allowed. In general, we follow the principal that data go in elements and metadata about elements go in attributes. (However, CASA tasks follow the convention that the *name* is an attribute so we will also.) For instance, if an element has a fixed type, or should not be changed by the user, we would use an attribute:

```
<myelement type="float"> 123.4 ></myelement>
<myotherelement type="string" readonly="true">Can't touch this</myotherelement>
```

B.1 Project

```
<project name="(the ALMA specified project name)"
</project>
```

B.2 Source

```
<source>
    <name></name>
    <coordinate>
        <!-- map this from/to the official FITS WCS convention -->
        <crvalN>
        <ctypeN>
    </coordinate>
    <equinox>
    <type> [galactic, extragalactic, etc]
</source>
```

B.3 Image

```
<image>
   <URI>
   <type> [data or thumbnail]
   <exported> [ true if contained in admit.zip]
   <description> [moment zero, moment one, spectral cutout, full cube.]
   <naxisN>
   <crpixN>
   <crvalN>
   <ctypeN>
   <statistics></statistics>
</image>
```

B.4 Spectrum

```
<spectrum>
   <!--
     Watch out for gridding effects:  If frequency has uniform gridding,
     velocity will not unless a regrid in that axis was done.
     Need proper WCS for this.
   -->
```

```
   <start freq>
   <start vel>
   <dvel>
   <nchan>   <!-- perhaps VOtable encapsulates number of channels already -->
   <statistics> </statistics>
   <votable> <!-- The spectrum actual data -->
</spectrum>
```

B.5 Statistics

```
   <statistics>
<!-- One can have multiple areas over which statistics are measured.
     Does casa multiple boxes print two stats or one?
 -->
        <region>
            <number>
            <!-- CASA Region Text Format, including channel or velocity range.
                 See
                 http://casaguides.nrao.edu/index.php?title=CASA_Region_Format
              -->
            <crtf type="string" >
            <mean>
            <max>
            <rms>
            <clip>
        </region>
   </statistics>
```

B.6 Task

We expect the ADMIT task will defined exactly as a CASA task, for which the XML representation is already defined as follows:

```
<task type="function" name="foobar" category="fumbar">
    <shortdescription>
    <description>
    <input>
    <param>
        <description>
        <any>
        <value>
    <returns>
    <example>
</task>
```

The only difference is that `casapy` keeps a local `task.last`, and a read-only version in `$CASA/share/xml/task.xml`, in ADMIT the current values are stored in the current `atask` of `admit.xml`.

B.7 Dependencies

This is a simple comma-separated list of BDP names. Note these are *not* chained dependencies. If BDP1 depends on BDP2 and BDP2 depends on BDP3, then BDP1 lists only BDP2 in its dependency list.

```
<dependencies>Name1,Name2,...,NameN</dependencies>
```

B.8 Date

This is the date and time following the FITS specification:

```
<date>YYYY-MMM-DDTHH:MM:SS.SSS</date>
```

C BDP XML Definitions

Here we detail XML definitions of specific Basic Data Products. Those marked "TBD" are still undergoing design.

C.1 Band

```
<band type="bdp">
  <number>
  <image>

  <summary type="bdp>
  <linelist type="bdp">
  <cubestats type="bdp">
  <posvelslice type="bdp">

  <!-- An "image" dependency means this BDP depends on
       its enclosed image URI or, if it has none, that of its parent.
   -->
  <dependencies> image </dependencies>

  <date>YYYY-MM-DD HH:MM:SS</date>
</band>
```

C.2 Line List

```
<linelist type="bdp">
   <votable>
   <!-- columns: name, restfreq, peakchannel, probability, where -->
   <!--     name  : fully qualified name. e.g. 12C17O(1-0)       -->
   <!--  restfreq : rest frequency in GHz                        -->
   <!--  peakchannel : channel of peak probability               -->
   <!--  probability : number between 0 and 1 indicating probability
                       that peak channel is this line.
                       0 = 0% probability, 1 = 100% probability.
     -->
   </votable>
   <task type="function" name="linelist" category="admit"> </task>
   <dependencies>cubestats</dependencies>
</linelist>
```

C.3 Cube Statistics

```
<cubestats type="bdp">
<!-- There will be one of <statistics> for each plane of the cube.
     The default region will be all of xy space, with the region
     keyword defining the boundaries and channel.
-->
  <statistics>  <!-- first channel -->
  ...
  <statistics>  <!-- last channel -->
```

```
  <!-- Depends on its parent's image -->
  <dependencies>image</dependencies>

  <task type="function" name="cubestats" category="admit"> </task>

</cubestats>
```

C.4 Moment

```
<moment type="bdp">
   <!-- moment value: 0,1,2 -->
   <integral>

   <description>
   <method>

   <!-- The FITS or CASA image -->
   <image>
       <type>data</type>
       <exported>false</exported>
       ...
   </image>

   <!-- The PNG image -->
   <image>
       <type>thumbnail</type>
       <exported>true</exported>
       ...
   </image>
   <task type="function" name="moment" category="admit">
        <param type="float" name="clip">
           <value></value>
        </param>
        <param type="float" name="vmin">
           <value></value>
        </param>
        <param type="float" name="vmax">
           <value></value>
        </param>
   </task>
   <dependencies>linecube</dependencies>
   <date>YYYY-MM-DD HH:MM:SS</date>
</moment>
```

C.5 Position Velocity Slice

```
<pvslice type="bdp">
<param type="float" name="pa"> Position Angle </param>
<image></image> <!-- the position velocity diagram -->
<image></image> <!-- Reference to the line cube from which the PV diagram was computed -->
<dependencies>linecube</dependencies>
<date>YYYY-MM-DD HH:MM:SS</date>
</pvslice>
```

C.6 Continuum Map

```
<continuum type="bdp">
<param type="int" name="nchan"> number of channels used</param>
<param type="float" name="bw"> total bandwidth used</param>
<image></image> <!-- the continuum image -->
</continuum>
```

C.7 Summary

TBD

C.8 Line Cube

TBD

C.9 Overlap Integral

TBD

C.10 Feature List

TBD – Depends on morphological analysis algorithms.

C.11 Description Vector

TBD – Depends on analysis algorithm.

D VOTable

All ADMIT tables will be VOTables. THe VOTable XML format is as follows:

```
<!DOCTYPE VOTABLE SYSTEM "http://us-vo.org/xml/VOTable.dtd">
<votable>
   <description>...</description>
   <resource  type="results">
      <description>...</description>
      <info name="QUERY_STATUS"  value="OK"/>
      <info name="distinct_dataset"  value="70"/>
      <table id="t1">
         <description>...</description>
         <field id="Ra"  unit="deg" datatype="char" name="ra_targ"  ucd="POS_EQ_RA_MAIN">
            <description>...</description>
         </field>
         <field id="Dec" unit="deg" datatype="char" name="dec_targ" ucd="POS_EQ_DEC_MAIN">
            <description>...</description>
         </field>
         <data>
             <tabledata>
             <tr>
                <td> 161.657 </td>
                <td>  17.152 </td>
             </tr>
             </tabledata>
         </data>
      </table>
   </resource>
</votable>
```