Astronomical Data Analysis Software and Systems XVI
ASP Conference Series, Vol. 376, 2007
R. A. Shaw, F. Hill and D. J. Bell, eds.

CASA Architecture and Applications

J. P. McMullin, B. Waters

National Radio Astronomy Observatory, Socorro, NM, USA

D. Schiebel

National Radio Astronomy Observatory, Charlottesville, VA, USA

W. Young, K. Golap

National Radio Astronomy Observatory, Socorro, NM, USA

Abstract. We describe the CASA (Common Astronomy Software Applications) package, its design and capabilities. CASA is a suite of applications for the reduction and analysis of radio astronomical data with a Python interface.

1. Introduction

Common Astronomy Software Applications (CASA[1]) are a suite of functions for the reduction and analysis of radio-astronomical data with a Python (IPython) user interface. CASA builds upon the libraries developed by the AIPS++ consortium, and is currently under development by NRAO, NAOJ, Penticton, and U. Calgary for the ALMA and EVLA observatories. CASA is currently in Alpha release within NRAO. Incremental releases to the community will occur in 2007.

2. Architecture

The CASA architectural components are:

- A suite of libraries for processing data (including the internal data model representation) principally implemented in C++, with some FORTRAN for efficient compilation of the innermost calculation loops.
- A command line interface providing access to the libraries.
- User reference documentation.

Previously (McMullin et al. 2006) we described the effort to update the system software and to provide a means of generating a common interface (C++/command line) to the developing libraries along with the required documentation. This system has been realized as illustrated in Figure 1 using Python as the command line interface/scripting language.

[1]http://casa.nrao.edu

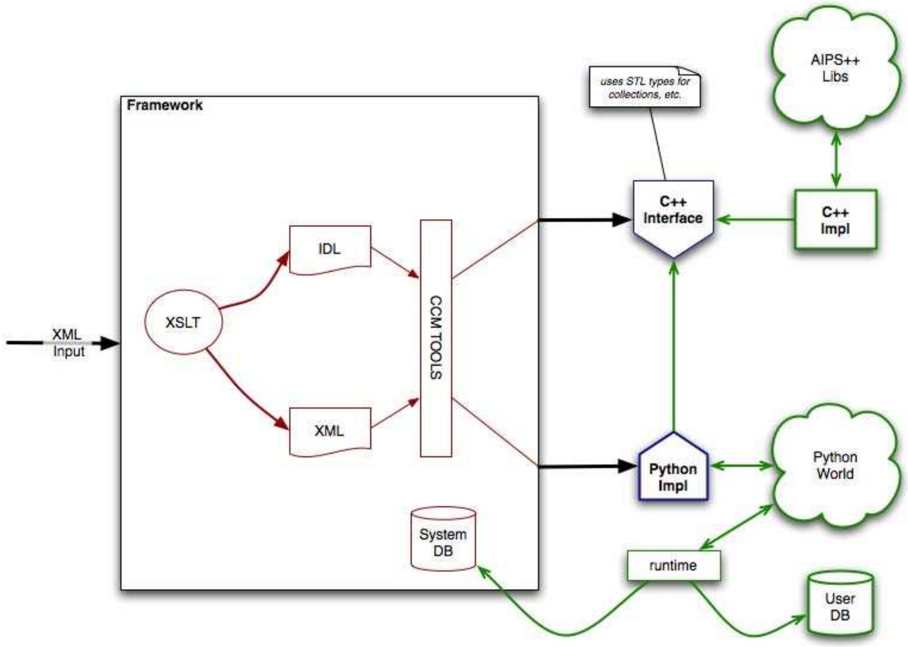

Figure 1. The CASA code generation process.

The CASA component system expands upon an open-source CORBA Component Model (CCM) implementation, CCMTools[2]. Component interfaces are specified in XML, processed with an XSLT transformation, which emits CORBA IDL3 and some XML meta information for our CCMTools processor. Our C++ interface stub permits the developer to implement new functionality in C++, or to expose features of an existing C++ library to the CASA system. The Python binding can use Python modules from the CASA code base or from other common modules (e.g., SciPy). Python is the scripting interface for the end-user. Distributed components may be implemented via the CCMTools MICO implementation of CORBA. Figure 2 summarizes the elements of the code generation system.

3. Applications

The key application elements are:

- A table-based data model capable of handling all of the observational data content.
- Facilities for loading data into the database from external formats, and the equivalent for storing into external formats.
- Python as a powerful, programmable Command Line Interface with scripting.

[2]http://ccmtools.sourceforge.net

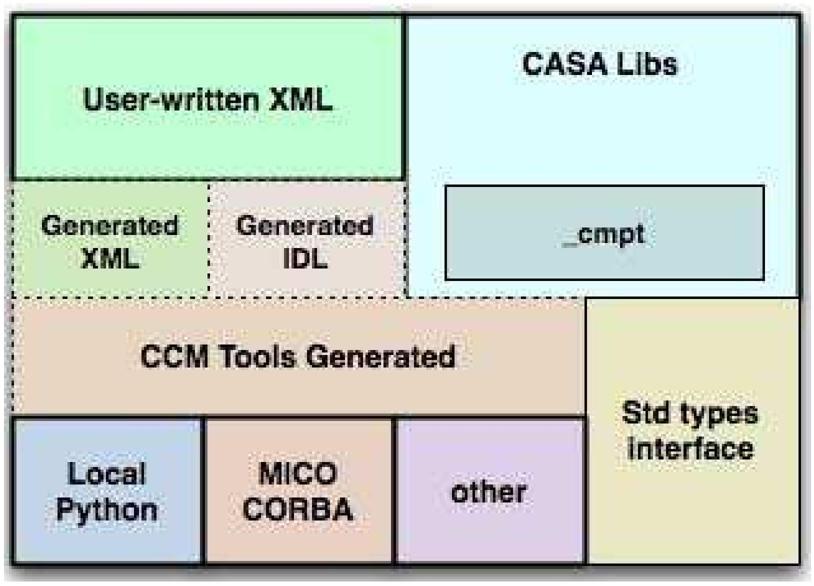

Figure 2. Summary of the elements of the code generation system.

- Low-level utilities that access, display, edit data. These tools need to know relatively little astronomy.
- An algebraic formalism for calibration and imaging (the Measurement Equation (e.g., Hamaker, Bregman & Sault 1996)
- Higher-level functions that encapsulate various astronomically important algorithms such as image formation, image deconvolution, astronomical analysis, etc.
- Facilities for the highly specific processing of data from radio telescopes into astronomical results.
- Documentation must be available for all aspects of the package.

The functionality is currently organized into two coarse categories or classifications: tools and tasks.

3.1. Tools

Tools provide access to both the lower level utilities (e.g., table browsing) and the fine-grained applications for astronomical processing (e.g., measures which enables coordinate transformations and manipulation of quantities with frame (time, position, direction) information). Tools have a somewhat object-oriented interface in that data is loaded into each tool and then manipulated by the associated functionality. The available toolset is currently:

- ms - Measurement Set utilities for manipulating the data (e.g., split out subsets, export, etc)
- mp - MSPlot - utilities for plotting quantities in a MeasurementSet
- cb - Calibration - utilities for performing calibration on visibility data
- cp - CalPlot - utilities for plotting calibration information
- im - Imaging - functions for creating images from visibility data
- ia - Image Analysis - utilities for manipulating images

- af - Auto Flag - functions for setting flags to data based on statistical measures
- tb - Table - utilities for examining the columned data in an MS or image
- tp - Table Plot - functions for plotting table data
- me - Measures - utilities for manipulating quantities with frame (time, position, etc) information
- pl - PyLab - the PyLab library interface

3.2. Tasks

Tasks provide a higher level interface to frequently used applications (assembled from the underlying tool functions); this is very similar to the interface provided by other radio analysis packages like AIPS or Miriad). The interface also supports the ability to poll the current inputs to a task, save the current inputs to a file and restore the inputs from a file enabling good versatility in scripting and pipelining of data reduction. The in-line help system is tied into the IPython interface allowing the use of: *task?*, *help task* or *pdoc task* to obtain help information. Currently the suite of tasks is small, representing only the most commonly used applications but is expanding during the current Alpha release. Status and updates on the package can be found at: http://casa.nrao.edu/info.html

References

Hamaker, J. P., Bregman, J. D., & Sault, R. J. 1996, A&A, 117, 137

McMullin, J. P., Schiebel, D. F., Young, W., & DeBonis, D. 2006, in ASP Conf. Ser. 351, ADASS XV, ed. C. Gabriel, C. Arviset, D. Ponz, & E. Solano (San Francisco: ASP), 319