Astronomical Data Analysis Software and Systems VII
ASP Conference Series, Vol. 145, 1998
R. Albrecht, R. N. Hook and H. A. Bushouse, eds.

Building Software from Heterogeneous Environments

M. Conroy, E. Mandel and J. Roll

Smithsonian Astrophysical Observatory, Cambridge MA 01801

Abstract. The past decade has witnessed a movement within the astronomical software community towards *Open Systems*. This trend has allowed projects and users to build customized processing systems from existing components. We present examples of user-customizable systems that can be built from existing tools, based on commonly-used infrastructure: a parameter interface library, FITS file format, Unix, and the X Windows environment. With these common tools, it is possible to produce customized analysis systems and automated reduction pipelines.

1. Introduction

Users and developers are confronted everyday with the challenge of cobbling together existing software to solve problems. However, much of the available software has features, limitations, and architectural assumptions that make it useless for new applications. It always is worth reminding ourselves that the primary purpose of software is to solve the users' problem: *Subject Oriented Software* (Coggins 1996) and not just to use trendy technology.

Currently, there is no set of uniform interfaces for the large body of existing astronomical software. Re-using a piece of software is complicated by the architectural buy-in of large systems which require mutually exclusive environments. (Mandel & Murray 1998).

Controlling complexity is the major problem facing software projects. The best watchword for developers is: *keep it simple.* Developers must keep tasks, their interfaces and execution environment as simple as possible because the lifetime of user software may be *one* execution. Therefore the most important software design features are ease of modification and adaptability.

2. System Components and Architecture

Software developers must address issues such as portability, platform independence, and freedom from licensing restrictions if they wish to free the users from these concerns so that the latter can solve analysis problems on their desktop. Users automatically gain the benefit of easily exchanging both software and data with collaborators independent of local environments. Toward this aim, the MMT Instrumentation project[1] surveyed the existing options and selected

[1]http://cfa-www.harvard.edu/mmti/

open-systems components to prototype several of our important applications. The critical components we have identified are:

- **Environment** A command line interface with a scripting language is essential for rapid prototyping and automated systems: a GUI is nice but not sufficient. POSIX components provide the architecture and platform independence for scripting languages, tools and software libraries. e.g., Korn shell, ANSI C and C++ libraries, FORTRAN90. POSIX also provides communication mechanisms such as pipes, shared memory and mapped files which can be used to efficiently pass data between processes.

- **Analysis Tools and Libraries** A tool-box of parameter driven tasks is needed to supply the necessary science algorithms. We have developed a system that we call UnixIRAF[2], that allows non-interactive IRAF tasks to be wrapped with generic Korn shell wrappers to emulate Open-IRAF tools. Starbase[3] provides another toolkit consisting of a full-featured relational data base (RDB) system. This POSIX-compliant toolkit can be used to construct complex data selection, extraction and re-formatting operations by piping together sequences of tools. These tasks are linked with the SAO Parameter Interface.

  Open-IRAF will provide C and C++ bindings for the IRAF libraries. SLALIB[4] will provide the world coordinate system libraries.

- **Visualization Tools/GUI** GUI-driven imaging and graphing applications are essential to aid users in understanding both the raw data and the analysis results. Tcl/Tk provides an architecture and machine independent widget set and a layered GUI written in a scripting language. We use SAOtng[5] for imaging and plan to re-use the Starcat catalog widget to add catalog position overlays.

- **Data Files** The tool box needs to run directly on standard machine independent data files, to free the users from the additional concerns of data conversion, archival formats and transportability. FITS *bintable* and *image* extension formats and ASCII tables are a necessary (but not sufficient) condition for providing machine independent formats. Additional conventions need to be added to FITS for metadata, world coordinate systems and other related information. This issue is complex and is covered in more detail in the Data Model work by several groups.

  FITS files are supported as native format both in IRAF for images and the TABLES layered package for (bin)tables. Starbase is being extended to support FITS bintable.

[2]http://cfa-www.harvard.edu/mmti/mmti/

[3]http://cfa-www.harvard.edu/~john/starbase

[4]http://http://star-www.rl.ac.uk/libs/slalib/mainindex.html

[5]http://hea-www.harvard.edu/RD/

- **Glue** There needs to be a mechanism by which these components can communicate with one another. The *SAO Parameter Interface* is such a mechanism, providing an API, interactive parameters, dynamic parameters and automatic parameter set configuration. When combined with XPA, it can connect analysis tools to visualization tools to build GUI-driven analysis applications (Mandel & Tody, 1995).

3. SAO Parameter Interface

Most of the items cited above exist in a variety of freely available implementations. However, the currently available parameter file systems have serious limitations when inserted into a heterogeneous environment. We therefore developed backward-compatible extensions to the traditional IRAF interface to create an *SAO Parameter Interface*[6] that allows multi-layered options for configuring applications and automating test scripts and pipelines:

- **Default Parameter File Override** We have added the ability to override the default parameter file specification. This allows multiple default parameter files to exist and be selected at runtime. E.g. radial_profile @@hst_prf or radial_profile @@rosat_prf
- **Common Data Sets** We have added a *Common Data Set* database to define dynamically configurable sets of parameter files. This provides the capability of automatically switching parameter files between pre-defined configurations based on the current environment: e.g., different time-dependent calibrations, several filters for the same instrument, or dozens of observations (and filenames).
- **Dynamic Parameter Values** There are many situations where the best parameter value is a function of other parameters or data. The parameter interface provides a mechanism to invoke an external tool to dynamically calculate a parameter value, returning the result to a program when it accesses this parameter at run-time.

3.1. Pipeline Applications

These parameter enhancements allow the developer to write generic pipelines that can be re-configured for different instruments and configurations. Meanwhile the user sees only a simple, explicit, reproducible batch script with no configuration dependencies because a complete, explicit record of the *as-run* parameters is saved.

The default parameter specification permits all the as-run parameters to be preserved even when the same program is run more than once. The common data set specification allows the pipeline to to be reconfigured when settings change, such as: filter in use, CCD-binning or instrument calibration. The dynamic parameters allow quantities such as bias and gain to be calculated from the current dataset and used as parameters by the calibration tools. These features

[6]http://cfa-www.harvard.edu/mmti/mmti

also allow pipelines to be reconfigured for different instruments so they can be re-used for new projects.

3.2. Interactive Analysis Applications

Dynamic parameters are very useful for coupling interactive tasks, allowing analysis to be driven easily the from image display. A simple scenario might be: image the data with SAOtng, draw regions of interest with the mouse, invoke analysis tools on the selected file and region. In this case *Dynamic Parameters* are used by the analysis tool at runtime to determine both the current file and the selected region to analyze. The dynamic values are determined by small scripts that invoke XPA to query the image display for the current file and the current region.

3.3. User Applications

Users often need to sequence several tools. The difficulties in making these user-scripts generic and re-usable stem from the fact that the filename changes at each step of the script and often the same parameter quantity has different names and/or units in each of the independent tools. *Common Data Sets* allow users to define an ASCII table to alias different tool-name:parameter-name pairs. Dynamic parameters can be used to automatically perform unit conversions.

4. Conclusions

The *MMT Instrumentation* group has used these components in all phases of the project, from instrument control and data acquisition to automated reduction pipelines and visualization. The toolbox consists primarily of existing IRAF analysis tools, special purpose instrument control tools and ICE tools. UnixIRAF enables the ICE data acquisition software to be controlled by simple POSIX-compliant Unix shell scripts in the same way as the instrument control software and the pipelines. Pipelines have been developed for CCD data reductions, spectral extractions and wavelength calibrations. Multi-chip CCD data are reduced efficiently by running multiple parallel pipelines for each chip. SAOtng and XPA are used to visualize mosaiced CCD data.

This approach has been highly successful. But it presents some challenges to the astronomical community: Who will contribute tools and components? Are developers rewarded for producing adaptable software?

References

Mandel, E. & Murray S. S. 1998, this volume

Mandel, E. & Tody, D. 1995, in ASP Conf. Ser., Vol. 77, Astronomical Data Analysis Software and Systems IV, ed. R. A. Shaw, H. E. Payne & J. J. E. Hayes (San Francisco: ASP), 125

Coggins, J. M. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. George H. Jacoby & Jeannette Barnes (San Francisco: ASP), 261

Part 4. Data Analysis Applications