Astronomical Data Analysis Software and Systems IV
ASP Conference Series, Vol. 77, 1995
R. A. Shaw, H. E. Payne, and J. J. E. Hayes, eds.

The PROS Big Picture: A High-Level Representation of a Software System

J. DePonte, J. Chen, K. R. Manning, D. Schmidt, and D. Van Stone

Smithsonian Astrophysical Observatory, 60 Garden St., Cambridge MA 02138

Abstract. We present a high-level representation of the packages and tasks within IRAF/PROS. The development of this overview is part of a comprehensive plan to document, and to improve consistency throughout, the IRAF/PROS analysis software. We anticipate that both current and future programmers will benefit from this document, and that the guidelines we have established will be applicable to other software projects.

1. Introduction

PROS is a multi-mission X-ray analysis software system designed to run under the Image Reduction and Analysis Facility (IRAF) (Tody 1986). The PROS software includes spatial, spectral, timing, data I/O, and plotting applications, and algorithms for performing general arithmetic operations with image data (Worrall et al. 1992). We have produced a high-level overview of PROS. The present paper describes this picture and the project that produced it.

2. Approach

A project to document a system requires a good definition for success. It can easily become unwieldy and misguided. The approach we took was based on four objectives. The first three were: a statement of mission, or what the project was intended to accomplish; a strategic plan to meet the objectives; and a set of short-term goals, to be met by means of scheduled task assignments. We also defined our long-term goals, a broader context for the mission.

The mission statement for the PROS overview specified four objectives: to communicate the content and organization of PROS to current and future programmers and projects; to make it easier to maintain the system and to evaluate enhancements; to identify code of high algorithmic content, for possible inclusion in libraries; and to identify temporary or prototype code for replacement or elimination.

The strategy was to do the project in stages, focusing at each stage on low-effort, high-payoff work that would produce useful deliverables. The incremental stages were the short-term goals: a project overview diagram (i.e., an organizational diagram of software modules); high-level diagram (i.e., a bubble diagram of each module in the system); and unifying element tables (i.e., tables of task vs. software attributes).

3. Implementation

To meet the long term goals within short term objectives, a hierarchical chart showing all packages and tasks was developed. The diagram was the basis for task assignments and progress evaluation (see Figure 1).

Next, we developed a high-level representation of the system. Each task was shown along with file I/O in a bubble diagram. Connectors were placed to show paths to other tasks within the same package. Standard analysis and design diagramming were applied, but standards were customized when appropriate for the application. For example, we used our own notation to represent inclusive/exclusive ORs and defined an icon to represent graphical output (see Figure 2).

Once diagrams were generated for all of the PROS tasks, the unifying elements of the system were identified. The elements (e.g., file I/O, code/data dependencies, maintenance of header keywords) were derived by analyzing the libraries and key software modules. We represented them as table columns and categorized the elements into four groups: I/O attributes (i.e., activities such as file I/O for each type of data file in the system); data attributes (e.g., the dependence of reference data on time); form attributes (e.g., whether the task is compiled code or a script, or whether it utilizes lookup tables); and derived attributes (e.g., whether the task maintains WCS or file header values, or performs error computation).

Next, a unifying element table was built for all tasks in the system. In the table, elements were represented as columns and tasks as rows. The parameter files, help files, and code, were used to analyze the modules and complete the tables. Table 1 shows an example of a table of I/O attributes of tasks in the PROS **xtiming** package.

The last phase focused on each identified attribute, documenting its usage in the code. Different projects could use different views for this phase of documenting their systems. For example, the structure charts of each module could be derived, or one aspect of the larger infrastructure could be identified and examined. In our preliminary work on this phase, we have taken the latter course, investigating the interrelations among library routines involved in implementing regions and masks.

4. Project Status

We have completed the project overview diagram, the bubble diagrams, and the attribute vs. task tables for all modules in PROS. We are currently analyzing code with identified attributes and documenting the usage throughout the system.

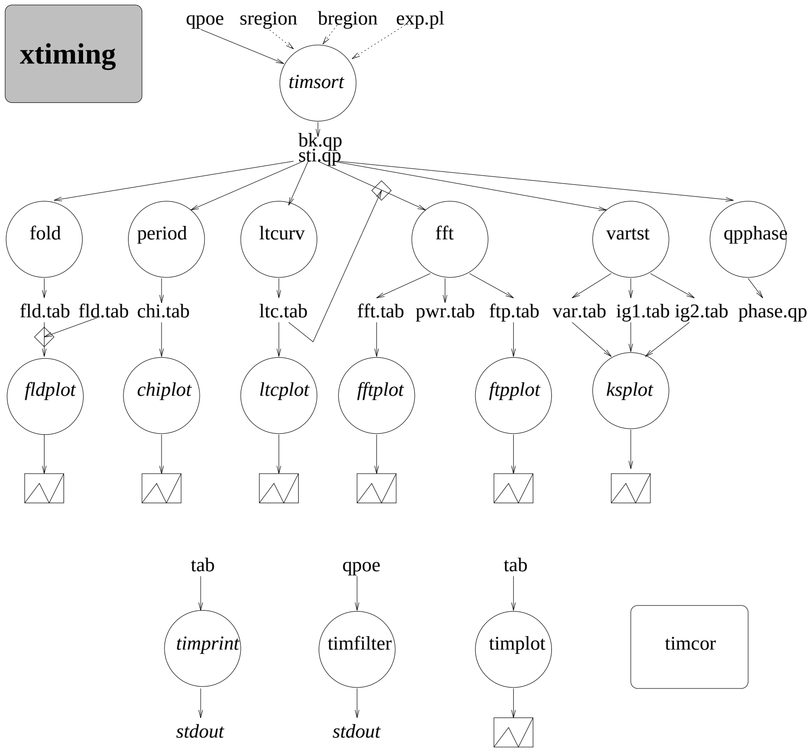

Figure 1. Task organization chart for the PROS package.

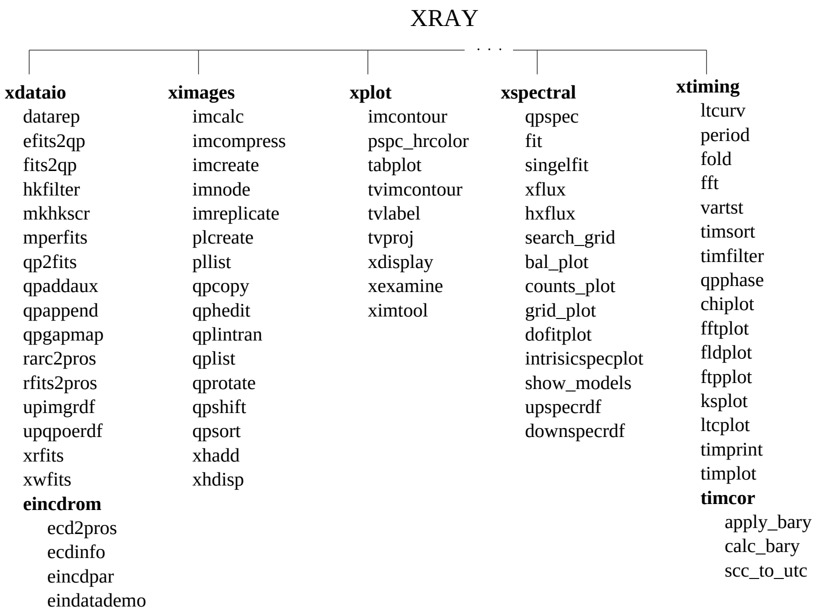

Figure 2. Bubble diagram for the PROS **xtiming** package.

Task	imio	qpio	plio	tabio	ASCII	bin	FITS	stdout	plt
ltcurv	–	x	–	x	–	–	–	–	–
period	–	x	–	x	–	–	–	–	–
fold	–	x	–	x	–	–	–	–	–
fft	–	x	–	x	–	–	–	–	–
vartst	–	x	–	x	x	–	–	–	–
timfilter	–	x	–	–	x	–	–	x	–
timplot	–	–	–	x	–	–	–	x	–
qpphase	–	x	–	–	–	–	–	–	–
ksplot	–	–	–	x	x	–	–	–	x

Table 1. Data I/O attributes table. The columns are I/O attributes. The rows are tasks in the **xtiming** package. Attributes that apply to a task are marked with an "x". Attributes that do not apply are marked with a "–".

5. Conclusion

The implementation decisions enabled us to produce useful products at each phase of the project. The high-level graphical representation of the system has already proved a useful tool in communicating PROS to others. The attribute vs. task tables identify library code, flag code that are trouble spots in preparing a software release, and aid in evaluating test procedures.

We anticipate that the procedures that we have established will be applicable to other large software projects. The effort we describe distinguishes each phase and can be customized to bring added understanding to a large software system.

Acknowledgments. This work is partially supported by NASA contracts to the *ROSAT* Science Data Center (NAS5–30934) and *Einstein* (NAS8–30751).

References

Tody, D. 1986, in Instrumentation in Astronomy VI, Part 2, SPIE, 627

Worrall, D. M., et al. 1992, in Data Analysis in Astronomy IV, eds. V. Di Gesù, L. Scarsi, R. Buccheri, P. Crane, M. C. Maccarone, & H. U. Zimmermann (New York, Plenum Press), p. 145