Astronomical Data Analysis Software and Systems VIII
ASP Conference Series, Vol. 172, 1999
D. M. Mehringer, R. L. Plante, and D. A. Roberts, eds.

From a "Launch Readiness" System to an Astronomical Data Processing System – a Review of Four Years of CIA Development

S. Ott

ISO Data Centre, Astrophysics Division, Space Science Dept. of ESA, Villafranca, P.O. Box 50727, 28080 Madrid, Spain, Email: sott@iso.vilspa.esa.es

R. Gastaud

DAPNIA/SEI-SAP, CEA/Saclay, F-91191 Gif sur Yvette Cedex, France, Email: RGastaud@CEA.fr

S. Guest[1,2], M. Delaney[1,3], B. Altieri[1], B. Ali[4], A. Abergel[5], J.-L. Auguéres[6], H. Aussel[6], J-P. Bernard[5], A. Biviano[1,7], J. Blommaert[1], O. Boulade[6], F. Boulanger[5], C. Cesarsky[6], D. Cesarsky[5], R-R. Chary[8], V. Charmandaris[6], A. Claret[6], C. Delattre[6], F-X. Désert[5], T. Deschamps[6], P. Didelon[6], D. Elbaz[6], P. Gallais[6], K. Ganga[9], G. Helou[9], M. Kong[9], F. Lacombe[10], D. Landriu[6], O. Laurent[6], P. Le Coupanec[10], J. Li[9], L. Metcalfe[1], K. Okumura[1,4], M. Perault[5], A. Pollock[1], P. Roman[1], D. Rouan[10], M. Rupen[11], J. Sam Lone[6], M. Sauvage[6], R. Siebenmorgen[1], J-L. Starck[6], D. Tran[6], D. Van Buren[9], L. Vigroux[6], F. Vivares[5]

Abstract. The ISOCAM Interactive Analysis System (CIA) is used to calibrate and to perform the astronomical data processing of data from ISOCAM, the infrared camera on board the Infrared Space Observatory (ISO).

[1]ISO Data Centre, Astrophysics Division of ESA, Villafranca del Castillo, Spain

[2]Rutherford Appleton Laboratory, Chilton, Didcot, Oxon, England

[3]University College Dublin, Belfield, Dublin, Ireland

[4]University of Rochester, Rochester, USA

[5]Institut d' Astrophysique Spatiale, Orsay, France

[6]CEA, Saclay, Gif-sur-Yvette, France

[7]Osservatorio Astonomico di Trieste, Trieste, Italy

[8]University of California, Los Angeles, USA

[9]Infrared Processing and Analysis Center, JPL and Caltech, Pasadena, USA

[10]DESPA, Observatoire de Paris, Meudon, France

[11]National Radio Astronomy Observatory, Socorro, USA

CIA is generally available to the astronomical community and runs under DEC VMS Alpha, Solaris, DEC Unix, Debian (PC) Linux and HP/UX. More details, including how to obtain CIA, can be found at `http://www.iso.vilspa.esa.es/users/expl_lib/CAM_top.html`.

We discuss the challenges faced in this multi-site, multi-environment project throughout the complete software development cycle, outline the chosen approach, and review the lessons learned.

1. Introduction

Starting mid 1994, the ISOCAM Interactive Analysis System (CIA)[12] was developed to support the calibration and operation of ISOCAM, the infrared camera on board of ESA's Infrared Space Observatory (ISO)[13].

As planned, this system now includes the functionality to perform the astronomical data processing of data from ISOCAM, and is used within the ESA ISO Data Centre, by the ISOCAM consortium and by many other institutes.

The system is mainly IDL based, and contains currently 1100 IDL modules with about 220000 lines of code and comments. CPU intensive tasks are coded in C++.

2. Challenges of CIA development

2.1. Requirements

Some requirements on CIA were

- expected to change with time (from a calibration/monitoring system for operational use to an astronomical data analysis system)
- conflicting (flexibility vs. user-friendliness)
- very broad (CIA had to be prepared for the unexpected and to serve as a development bench for new algorithms, which ultimately will migrate into the general pipeline)
- unforeseen additional major requirements, e.g., CIA had also to run on Unix.

2.2. Management

The development of CIA had to be spread over multiple development sites.

Furthermore, the schedule for the first version ("launch readiness version") was extremely tight — 9 months from the start of development to acceptance testing, and the development team also had to build up IDL[14]/C++ expertise.

---

[12]CIA is a joint development by the ESA Astrophysics Division and the ISOCAM Consortium. The ISOCAM Consortium is led by the ISOCAM PI, C. Cesarsky, Direction des Sciences de la Matiere, C.E.A., France.

[13]ISO is an ESA project with instruments funded by ESA member states (especially the PI countries: France, Germany, the Netherlands and the United Kingdom) and with the participation of ISAS and NASA.

[14]IDL is the trademark of Research Systems, Inc.

2.3. Other challenges

As in every other software project, special attention had to be paid to software quality control and documentation. The single most difficult and time-consuming problem was to get the astrometry, e.g., the correlation of image elements (detector pixels) with astronomical (sky) coordinates, right. (ISOCAM contains two detectors, which are read-out electronically in different directions. Also its images might be re-binned to non-square pixels.)

3. Chosen approach

3.1. Requirements

In order to keep the system on track, requirements were strictly prioritized. The system is reviewed and, if necessary, pruned from time to time. In order to fulfil "expert" and "normal" user requirements, two different types of data structures were implemented. CIA was coupled with the general automatic data reduction pipeline via its architecture, not via the code. This eased development for *both* systems.

3.2. Management

Project management and control was closely entwined with software configuration control: One VMS system was chosen as the master development/configuration control system and mirrored to the other VMS development system. Unix versions are created at monthly or bi-monthly intervals from the current development version and installed at the other development sites.

The collaborative and collegial approach within the software team and among collaborators was essential to overcome all difficulties.

3.3. Other challenges

A CIA configuration control board reviews and sets the priorities of the implementation of software problem reports and software change requests. This, together with a several-month-long testing period before the release of any major new version, proved essential to achieving a high quality for the software.

For the documentation, emphasis was put on the production of an accurate and helpful User's Manual. CIA's internal help system relies on the information contained in module headers. Therefore all contributors were continuously encouraged to maintain them in case of a code change or to modify them in case of deficiencies spotted.

4. Lessons learned

1. Management
    - the effort to code properly and to maintain a stable system is normally underestimated by all participants and by their hierarchy.
    - quite often it is assumed that coding can be done as a by-product of algorithmic development. However, it was found that a proper balance in staffing between astronomers and software engineers is not only essential to produce a maintainable system but actually more efficient!
    - frequent ($\approx$ monthly) workshops between developers, and bi-annual users meetings, are essential.

2. Architecture
   - a proper definition of data structures is important. Growing data structures and flexible export tools are preferable to hard coded data structures and export tools which have to be upgraded in case of new requirements.
   - link systems with a different primary programming language only via the architecture, not via the code. Otherwise the progress of *both* systems will be slowed down.
   - if possible, avoid non-standard data structures depending on pointers. If non-standard data structures are necessary, provide an intuitive access. (NB: IDL does not provide this functionality).
   - while collaboration is the key to success, some rules have to be obeyed:
     - avoid duplication of already existing functionality.
     - enforce consistent naming conventions for routines and for the tag names within data structures.
     - avoid private ports to other operating systems.
   - avoid operating-system-dependent data files.
   - store the processing history as part of the data instead of using log-files.
3. Implementation
   - IDL permits an easy port between systems, compared to classical languages. (But the support of different operating systems remains nevertheless an overhead). However, internal IDL changes from version 3.6 to 5.0 for the widget interface led to a major recoding effort.
   - the maintenance of module headers is a Sisyphean task which needs dedicated manpower.
   - knowledge of "expert" language features like *_extra* can speed up the development significantly.
   - look for solutions available in other packages (e.g., ASTROLIB). But the use of several packages can lead to compatibility problems (e.g., name conflicts between functions and variables).
4. Configuration Control & Multiple Development Sites
   - good network connections and easy-to-use configuration control systems are essential.
   - the use of mirrors reduces synchronization effort dramatically.
   - to avoid bad surprises for users, both a floating development version and a frozen user's version are necessary.