THE INTERACTIVE ANALYSIS WORKING GROUP FINAL REPORT

K. Leech[1], Otto H. Bauer[2], Diego Cesarsky[4], Helmut Feuchtgruber[2], Carlos Gabriel[1], Ken Ganga[5], Pedro García-Lario[1], Rene Gastaud[6], Gerard Hutchinson[8], Martin Haas[7], Do Kester[2], Ulrich Klaas[7], Steve Lord[5], Patrick Morris[4], Stephan Ott[1], Peter Roelfsema[3], Russell Shipman[3], Sunil Sidher[8] Ekkehard Wieprecht[2]

SAI/2000-008/Dc, Version 1.9

August 13, 2001

[1] ISO Data Centre, Astrophysics Division, Space Science Department of ESA, Villafranca, P.O. Box 50727, E-28080 Madrid, Spain

[2] Max-Planck-Institut fur extraterrestrische Physik, Postfach 1603, D-85740 Garching, Germany

[3] SRON, P.O. Box 800, 9700 AV Groningen, The Netherlands

[4] Institut d'Astrophysique Spatiale, Universite d'Orsay, F-91405 Orsay, France

[5] Caltech, IPAC, Pasadena, CA, USA

[6] Service d'Astrophysique, DSM/DAPNIA CEA Saclay, Orme des Merisiers, F-91191 Gif sur Yvettte CEDEX, France

[7] Max-Planck-Institut fur Astronomie, Konigstuhl 17, 69117 Heidelberg, Germany

[8] RAL, Chilton, Didcot, Oxon OX11 0QX, UK

Document Information

Document:	The IA WG Final Report
Title:	The IA WG Final Report
Reference Number:	SAI/2000-055/Dc
Issue:	Version 1.9
Issue Date:	August 2001
Authors:	K. Leech

Document Change Record

Date	Revision	Comments
Nov 2000	Version 1.0	First Draft
May 2001	Version 1.1	Second Draft
May 2001	Version 1.2	Third Draft
Jul 2001	Version 1.3	Fourth Draft
Aug 2001	Version 1.4	Fifth Draft – CIA paper included
Aug 2001	Version 1.9	Final Draft – PIA paper included

IA WG Webpage

The IA WG webpage is kept by K. Leech at http://isowww.vilspa.esa.es:1909/ kleech/WG/index.html.

Contents

Chapter 1

Introduction

1.1 Team Members

The Interactive Analysis (IA) WG members were the following: Helmut Feuchtgruber, Carlos Gabriel, Pedro García-Lario, Rene Gastaud, Gerard Hutchinson, Do Kester, K. Leech (chair), Stephan Ott, Russell Shipman & Ekkehard Wieprecht.

Observers were: Otto H. Bauer, Diego Cesarsky, Ken Ganga, Martin Haas, Ulrich Klaas, Steve Lord, Patrick Morris, Peter Roelfsema & Sunil Sidher.

1.2 Why have an IA?

It is not immediately obvious why an IA system should be developed for a space mission. It costs extra in manpower and there are many astronomical software packages available. Why go to the bother of developing an IA?

There are several reasons why IAs are more than a nice-to-have.

- If you are developing an Off-Line Processing (OLP) system you have to be able to develop the system, try routines out etc. An IA system can be the perfect testbed for such activities. As an example, the Short Wavelength Spectrometer (SWS) IA was primarily designed to help develop the SWS OLP system.
- An IA system is essential when deriving the calibration files that describe the instrument.
- IAs are a flexible approach to data reduction, whereas OLPs are, by definition, not flexible. An IA system can rely on users knowledge of how to reduce observations affected by rare events that are not worth coding into the OLP system.
- An IA system is essential to perform ad-hoc investigation of instrument anomalies

Furthermore, many of the IA routines can be re-used in OLP software, QLA, RTA as necessary. The added expense in developing the IA system can be minimal and can reduce the time needed to develop OLP.

1.3 Why have an IA WG?

The purpose of the IA WG, as set out in its Web page, is to:

- Coordinate the information flow between developers of the different instruments' IAs. Each team should be aware of what other teams are doing. S/W tools developed by one of the teams should be made available to other teams if they consider it useful.
- Stimulate the implementation of new S/W tools in the four ISO instruments' IAs.
- Raise the profile of the various IA packages as means to enhance data reduction.
- Investigate communication channels between the IA users and IA developers in both directions.

1.4 What did the IA WG do?

To achieve the purpose the IA WG held four meetings, minutes of which can be seen on the IA WG web pages (internal access only). The IA WG chairman also circulated e-mails concerning problems with IDL, issues to do with upgrading etc. And it produced this final report.

1.5 Why have an IA WG Final Report?

Individual developers have passed on their hard learnt lessons to other space missions. This has been done by moving onto other missions, attending conferences and by writing papers on their IA for the ISO Calibration Legacy Conference. This document attempts to gather many of their recommendations together into one place.

Chapter 2 reports on a questionnaire sent around to all IA developers to see what had been learned. Chapter 3 discusses the lessons learned from the development of the IAs. Appendix A discusses the lessons learned from the development of ISAP, appendix B gives the results of the questionnaire sent round to the IA developers, while appendix C reprints the various papers submitted to the legacy conference.

Chapter 2

Questioning the developers

2.1 Looking back on IA development

Near the end of the development of ISAP (ISO Spectroscopy Package) the developers gathered together to summarise their experiences. Their full paper is listed in Appendix A, while their recommendations are subsumed into the general lessons in section 3.

Similarly, the IA WG chairman distributed a questionnaire to all IA writers to try and capture their experiences. The full results are given in Appendix B, with some general trends apparent.

The one general trend that can be gleaned from these answers is that the IAs changed with time. Their prime purpose, their target users and even the programming language used to write them in changed over the mission. Change should be planned for in the sense that you want the systems you develop to be as flexible as possible so as to accommodate change, but none of the developers planned for this.

What is surprising is that the developers did not expect that astronomers would want to use their software for astronomical analysis to the extent they were used. Initially they were written either to calibrate the instruments or to develop software, and only later modified for general astronomical data reduction. Software written to calibrate an instrument is inherently useful for astronomers and it should be expected that they will want to use it. By the end of the Active Archive Phase (AAP), although the OLP data products are good for many observations, a large number of observations have to be processed by the IAs to extract the maximum science.

What is also interesting is that none of the IA developers have any real idea how many users their software has. CIA knows sites, and ISAP has a good idea, but it was felt to be a mistake not to keep track of users via some means.

In general the IAs were developed with a small team of people, ranging from 10 to 20, with a larger number of people contributing to a lesser extent. There was considerable turnover for the lessor contributors, with a small core retaining consistency. Despite the small team size, the IAs turned out to be fairly large software projects.

The IAs were started after the OLP pipelines were begun, which was felt to be a mistake. If they are meant to be a means to develop the pipeline they should be started before.

Chapter 3

Recommendations applicable to the Future

Combining the ISAP recommendations with other general recommendations from the IA WG developers produces the following list. Some recommendations appear more than once in different sections.

3.1 Systems

Start developing the system(s) early – 5-6 years before first light.

Hire a project manager to assist in the interaction between scientists and engineers. A scientist alone typically has insufficient time for that.

All routines must be speced and the specs must be used for documentation.

Formal test plan and test logs are needed.

Use an established configuration control system.

If writing software for more than one instrument, determine where the commonalities lie between the instruments early on in the project.

3.2 Architecture

Use the minimum number of programming languages. The more you have, the more it will cost in terms of manpower and money to support them.

Use the same language for any pipelines, RTA, QLA, IA etc.

Support the minimum number of systems for the same reasons.

Define the fundamental unit of data or object (e.g. arrays vs. structures) early on and try to stick to it.

Software lives longer and is much more expensive than hardware. Try to make sure the software is as much hardware and platform independent as possible.

3.3 Documents

Establish a software requirements document and enforce its implementation.

The URD must be written by the ultimate users – astronomers.

The ADD must be written by s/w engineers.

Formal test plan and test logs are needed.

Prepare detailed tutorials, user manuals etc for the users.

3.4 Developers

Development teams are usually at different institutes in different countries. Have the development team congregate in one location regularly to do work together (team building).

Regularly review of the code for content, style, approach is necessary.

Develop close relationship with software vendors (where used), enabling the team to be aware of changes to the commercial software before they're implemented.

URD writers should be astronomers and in close contact with (ideally part of) the instrument team(s).

Use a team with a healthy mixture of scientists and s/w engineers.

Remember that people are expensive, computers cheap.

3.5 Users

Ensure you know who your users are. This means that they must register to get the software, and to make this palatable keep registered users aware of what is going on via users communities, newsletters etc.

Develop a viable user community, via such things as meetings, newsletters etc.

Appendix A

Lessons learned from ISAP

A.1 Introduction

Steve Lord, IPAC, for the ISAP Group, June 1999

The following is the report from a three day review conducted at IPAC (June 28-30 1999) by members of the ISAP team, recording "lessons learned" in the development of ISAP, the ISO Spectroscopy Analysis Package.

Purpose of this report is to identify what our team did right and what we did wrong, so as to aid future efforts. Our summary recommendations may be found by looking for entries marked REC:

Participants:

S. Lord, E. Sturm, S. Molinari, I. Khan, J. Brauher, S. Unger, S. Sidher, J. Li, O. Bauer, K. King, D. van Buren.

A.2 Summary of ISAP

ISAP took 5 (real time) years to develop; starting 1.5 years before launch.

It required 18 person-years of work - split between designers (40%), testers (20%) and developers (40%). This was a internationally distributed team.

100,000 lines of code were written, including comments. More than half the code lines are non-comments.

The final package consisted of 280 IDL procedures

There are 11 major modules - Average, Photometry, Rebin, Arithmetic, Defringe, Blackbody Fitting, Line Fitting, Zodiacal Light Subtraction, Clipping, Smoothing, and Shifting.

There are about 30 minor modules, performing less fundamental tasks, and with some assisting major modules.

The package can be run in either a batch (command line) mode or a GUI mode

Our user base consists of more than 180 ISO Astronomers - and it is still growing by 20

The package is written solely in IDL code - all ASCII procedures.

We developed routines for both scientific analyses and ISO instrument expert routines

The package's most notable virtue is that is provides an efficient way to look at one's data.

During the five years, there were seven releases, all offering major improvements.

It received high marks from it's users in several IPAC surveys.

ISAP Lessons Learned.... A collection of the comments and recommendations of the the ISAP designers, developers, and testers.

Comments were made regarding the good and bad parts of the process and the software. We divided our collective comments into four areas: 1) work environment we were forced to work within, or else chose to work with; 2) decisions made regarding the software architecture; 3) how well our development process worked, and finally, the pros and cons of the software product. All four areas are listed below, with pros, and cons, and recommendations (RECS).

A.2.1 ENVIRONMENT

PRO

-we formed good interpersonal relationships -we were a dedicated, motivated team of stakeholders -we held useful engineering workshops ("hacks") -our product is supported under UNIX and VMS -we were assisted by good external process reviews -we started with and stuck to a good directory structure

CON

-IDL changed under our feet. (An especially hard setback for our team was the upgrade from IDL v. 3.6 to IDL v 4.00.)

-ISO pipelines changed

-the ISO data understanding changed

-we had a severe lack of realistic testdata before development began

-the distributed software needed implementation on many different window environments

-we were understaffed

-we experienced some personnel turnover

-the data always lacked true error bars,

-the combined UNIX and VMS version maintenance was difficult

-the data always lacked true error bars,

-we lived with complicated ISO politics (LIA, OSIA vs. ISAP)

-our schedule was driven by external factors: -other packages -user community needs -propriety data periods expiring

REC:

-Have international development team congregate in one location regularly to do work together (team building).

-Hire a project manager assist in the interaction between scientists and engineers. A scientist alone typically has not sufficient time for that.

-Establish a software requirements document and enforce its implementation

-Regular review of the code for content, style, approach is necessary.

-Develop close relationship with software vendors. (Predict change.)

-Designers should be astronomers and should be in close contact with (ideally part of) the instrument team(s).

-Define commonality between instruments early on in the project.

A.2.2 ARCHITECTURE

PRO

-We established routines that supported both GUI and command lines

-we used no binary files

-ISAP was coded in pure IDL

-It is easy to add command line routines to the product

-We established connectivity with the other interactive spectroscopy packages: LIA and OSIA

-we were efficient in using commonly shared modules (e.g., CW_DRAW)

-we captured IDL's astrolib and made a static and internal version of it.

-we found common routines for LWS and SWS

-we required procedures with standard headers (finally).

-the GUI provided flexibility and multiple routes to access software tools

-we made sure that the user could alternate between command line and GUI modes

-IDL modules interlaced data processing and graphics in one file.

CON

-our's was a changing architectural design

-we mixed of modal and non-modal widgets - adding to inefficiency

-the software architecture was such that it is not easy to add GUI routines

-compatibility with command line (especially SWS IA) necessitated complex structures

REC:

-Significant development effort could be saved by not requiring all routines to be callable from a central GUI. I.e., have no central GUI.

It is easier to develop command line routines which provide a GUI when

called (the "OSIA/LIA mode"). Some convenience is lost for the user, though.

-Define the fundamental unit of data or object (e.g. arrays vs. structures) early on and try to stick to it.

A.2.3 PROCESS

PRO

-we had an efficient production of code

-algorithms were designed by instruments experts

-we responded to ISO community

-we responded to Instrument Teams

-we learned development techniques (formal specs etc.)

-we had initial design manual

-we had good version control

-we found hosting workshops gave good feedback

CON

-we started way too late

-we formed changing specs

-we suffered at times poor code ownership control

-we had a lack of code development strategy to apply to all developers

-we forgot our programming rules

-testing was never a formal process

-we were short-staffed we used user community as beta tester

-we had continual redesign of modules

-we developed too quickly

-our astronomer algorithm designers had erratic schedules

-we didn't reuse specs for documentation

-we often started coding without hard specs

-we lacked a "look and feel" manager (style review)

-our contact to RSI not close enough

REC:

-start early - 5-6 years before first light.

-all routines must be speced and the specs must be used for documentation

-formal test plan and test logs are needed

-use established configuration control system

A.2.4 PRODUCT

PRO

-ISAP contains advanced science tools

-ISAP is versatile

-it is an expert LWS/SWS system

-ISAP has functional depth

-ISAP transfers expert knowledge to user

-ISAP is easy to use

-ISAP is easy to distribute

-Documentation and download have easy web access

-we wrote detailed, useful tutorials

-the code has potential for other missions (algorithms at least)

-it's very interactive

-it's use is intuitive

-All ISAP routines have good documentation (at last)

-All ISAP routines have documentation on the web

-ISAP has some important platform independence

CON

-our RSI stability dependence was not predictable

-the whole package use is based on a difficult conceptual model (especially memory usage and diversely located functionalities

-final code shows inconsistent programming styles

-some code has insufficient internal commenting

-use of product's memory access too complicated

-to merge S/LWS - we have data tags with dual meanings (e.g., LINES)

-we gave late documentation

-the software's pop-up show inconsistent button conventions

-some modules are way too large

REC:

-prepare detailed graphical tutorials

-this type of product should be planned to be made available for future missions

Appendix B

Questionnaire sent to other IAs

In order to canvass for views on the development of the other IAs, questions were sent around to the other IA writers. The views of the IRSKY developers was also canvassed, as this program was used heavily in the planning of ISO Observations.

The questions were directed at, and answered by, the following people:
OSIA Ekkehard Wieprecht, Kieron Leech
LIA Sunil Sidher, Gerard Hutchinson
CIA Stephan Ott
PIA Carlos Gabriel
ISAP & IRSKY Steve Lord

The questions were as follows:

B.1 Was your IA started before or after the Pipeline?

OSIA IA, separate from OSIA, was started after the pipeline. This, however, was a mistake. We should have developed PIPELINE within IA.

LIA After

CIA After the programming for the first pipeline steps (tdf-first-scan, derive-erd and derive-spd), but before the final pipeline processing step (auto-analysis) was begun.

PIA After. First prototype begin of 1994 - Arch. Design April 94 - Coding begin in June 94 (on IDL 3.6)

ISAP It was started in RAL at the same time.

IRSKY This was conceived and written (1st version) over 3 years before launch of ISO.

B.2 Was your IA started before or after the Launch?

OSIA IA was started before launch, but OSIA (designed primarily for the general user) was only started after the end of the mission. This was far too late.

LIA After.

CIA Before.

PIA Before. See above.

ISAP Before by 1 year.

IRSKY Before. See above.

B.3 What were the Design goals?

OSIA Initially IA was designed to test the pipeline and to calibrate SWS.

LIA The software allows LWS data users to inspect the quality of the pipeline produced data, to resolve some of the specific problems encountered (e.g. glitching, dark currents, detector responsivity drift) which were not initially resolved by the pipeline in a satisfactory manner.

CIA The design goals were that CIA should be able to be used to: calibrate ISOCAM; monitor the health of ISOCAM; perform any sort of investigation requested for problem diagnostics; debug, validate and refine the off-line processing system for ISOCAM; assess the quality of ISOCAM off-line processing products; and perform the astronomical data processing of ISOCAM data.

PIA From the very beginning there was a strong accent in developing a calibration tool, to be used later for interactive scientific analysis. Therefore user friendliness (eg. GUI, but also flat treatment of the ISOPHOT data from different sub-instruments, 'external' documentation, etc) was written with capital letters.

ISAP Originally for the LWS IDT to try out pipeline ideas during PV phase.

IRSKY To offer an observation planning tool that allowed the user to see the infrared sky in a region, see what other observations had been made in a region and also give expert knowledge about the instrument's view of the sky-pixels, apertures, rasters. The background and confusion estimators were tuned to the parameters of the observatory.

B.4 Did these goals change during the evolution of the IA Package?

OSIA Yes. In the end it turned into a general purpose data reduction package.

LIA Yes. The ultimate goal of LIA became the following: Software which allows LWS data users to inspect the quality of the pipeline produced data, to use different data reduction algorithms from the defaults and to optimally calibrate their data interactively to produce high quality data.

CIA Yes. A new requirement that CIA should be able to run under additional platforms (Solaris, DEC Unix, Linux, HP/UX) was introduced. And as expected, over time the emphasis shifted to astronomical data processing.

PIA Not much. However, the horizon was a bit expanded with the confirmation of the absolute necessity of the package for general scientific analysis, due to the strong limitations of the pipeline / default calibration outcome.

ISAP It became the standard post pipeline reduction tool for LWS data (and to some extent SWS).

IRSKY Not really; more was added as we discovered we could do it - Hubble Guide Stars, Object Visibility, Objects already proposed, eventually another mission: SIRTF

B.5 Who was it primarily designed to be used by?

OSIA Pipeline developers and calibration specialists.

LIA Specific LWS data users working closely in collaboration with expert user(s) who wrote LIA at IPAC or RAL.

CIA Calibration scientists and astronomers.

PIA Both calibration experts and general observers.

ISAP Initially the LWS IDT.

IRSKY Observers, planning their AOTs.

B.6 Did this change during the evolution?

OSIA Yes. Initially IA was aimed at the calibration specialists and pipeline developers. Then it was found that astronomers were using it for data reduction, so OSIA was released with them primarily in mind.

LIA Yes. The software became more accessible to the general user with a wider range of more flexible routines becoming available.

CIA In principle, no. But the ratio of calibration scientists to astronomers changed dramatically: from 10:5 to 0:100.

PIA No, although the accent changed. In the beginning the main calibration needs were covered. During the evolution the package development was more and more driven by general user's requirements. This was, however, foreseen from the very beginning.

ISAP Yes, to all ISO spectroscopic observers.

IRSKY No. But it proved useful for other projects - at other telescopes, and just looking at the sky for planning of other missions, etc..

B.7 Which languages were used?

OSIA Initially IA was supposed to Fortran code running under Midas, and was prototyped as such.

LIA IDL. Some pipeline FORTRAN code was converted into IDL.

CIA IDL, Fortran, C, C++.

PIA Exclusively IDL.

ISAP All ASCII IDL procedures.

IRSKY Originally a Widget Language called JAM, and then it was translated into Motif - in C. Some FORTRAN subroutines were tacked on as well.

B.8 Did this change during the evolution?

OSIA Yes. Before launch the use of Midas was rejected in favour of IDL. In IDL we connected the FORTRAN pipeline modules as shared libraries. Much later in the mission we recoded all of the pipeline in IDL. From OLP 9.0 on our complete pipeline was in the same language as IA.

LIA No. IDL remained the language used.

CIA C and C++ code was introduced for CPU intensive processing. The original idea to run Auto-Analysis within CIA was abandoned, as the IDL call_external was not stable.

PIA No.

ISAP No.

IRSKY Yes - see above.

B.9 How many authors were there, what was their distribution, and what was their turnover?

OSIA It had about twenty authors spread between Holland, Germany, Belgium and Spain. The core development team stayed fairly stable throughout development. These twenty were a mix of scientists and software engineers. Best progress was made when developers were co-located.

LIA There have been a large number of different authors. There have been about a dozen main contributors: Sidher, Swinyard, Lloyd, Unger, Molinari, Lord, Brauher, Hutchinson, Chan, Khan, Harwood, Leeks. There were two main centres working in collaboration: RAL and IPAC.

There was quite a significant staff turnover in most cases. Only Sidher and Lord have remained closely associated with LIA throughout. On average an individual would have made a significant contribution for approx. 18 - 24 months.

CIA Around 10 major contributors, and around 40 less involved contributors. The core contributors continued to develop CIA during the whole mission.

PIA The team was composed of up to 5 people. In average 3 persons built the team. The average FTE was 2. The total number was 8. 70% was done by ESA (including team direction), 30% by PI and in the late stage IPAC provided people. Continuity was assured by the ESA personnel (from the very beginning to the end of the development: April 1994-December 2000), average time of PI/IPAC provided collaborators was 2 years (min=1 year; max=3 years).

ISAP See Appendix A for a discussion of this point.

IRSKY There were about 8. One person did the user interface which was a major component. Others has a 1-3 year role. There was high turnover.

B.10 What was the final size of the package?

OSIA Approximately 1000 modules and 280000 lines of code (with comments).

LIA There are about 14 main routines with a larger number of associated subroutines. Most of the routines have several options and so the typical user should not need to use more than a few routines on a single dataset.

The next version(s) of LIA, will add about 8 new routines, so it is anticipated that LIA will eventually comprise of a suite of about 20 separate main routines.

CIA The grand total of the whole system, including configuration control files and test-files, comprises 18000 files and 2.4 GB. The delivered version (including documentation and calibration files) is only 3700 files and 160 MB.

PIA

ISAP 390 IDL procedures.

IRSKY 23,000 lines of original code - maybe 20% more. 2 MB for the binary object

B.11 How many users does it have? Number of new users per year?

OSIA We do not know. One of our mistakes was not to evaluate the users.

LIA 94 registered users to date. An average of around 27 per year, but probably more than this during the first 12 - 18 months of release. The exact yearly figures are not known.

CIA Difficult to guess, as we only keep track of the site where CIA is installed. Currently this is above 80. Around news 20 sites register per year.

PIA No firm numbers here. No users registration. My estimation: peak by 200-400, with about 50 strong users.

ISAP It has 230 users, with approximately 20 new users per year.

IRSKY Tough question - I wild guess would be 500 in the US, several hundred in Europe

B.12 How was it maintained and distributed?

OSIA It is maintained using CoCo, a configuration control system that locks modules being modified, distributes changes and ensures all sites remain up to date.

LIA Initially maintained jointly by IPAC and RAL, and released by IPAC. More recently, maintained and released by RAL, available via the UK ISO Data Centre website, with a US mirror site at IPAC.

CIA Development and maintainance was led by ESA and was a collaborative effort involving SAp, IPAC and IAS. The VMS development version is mirrored between SAp, VilSpa and IAS. The monthly Unix version is available for development sites via FTP. The annually released version can be downloaded from VilSpa via FTP.

PIA Maintained by the development team. Strict configuration control only in the beginning of the development (many developers + fast and deep changing S/W). Afterwards (during the ISO mission) configuration control by the maintenance of a central repository and "development" areas. "Acceptance" mechanisms in the hands of the team leader.

PIA was distributed as compiled package in tarfile through the web: PIA homepage. Documentation (html system + TeX documentation) also distributed through the homepage.

ISAP Downloaded from IPAC and RAL by FTP.

IRSKY A team maintained it with three prime people - an Astronomer, and two programmers with expert knowledge

B.13 Was it platform independent?

OSIA Initially it was designed to run under UNIX and VMS. Later we decided to support Linux, but not WINDOWS.

LIA No. It was initially written in IDL to run under VAX/VMS and had to be ported onto Unix IDL early on.

CIA It runs under VMS/Alpha, Solaris, DEC Unix, HP/UX and Linux.

PIA Up to 1996 Unix and VMS - VMS special support was ended mid 1996 (after confirmation of no need by the main PIA user's community). From then only Unix (all flavours since pure IDL).

ISAP Yes, it has worked on VMS and UNIX.

IRSKY It was portable to Sun OS, Solaris, Linux, UNIX

B.14 How necessary is it to use the IA's as opposed to OLP products?

OSIA In most cases IA products are preferable to OLP products.

LIA Most of the LWS problems have now been resolved by the off-line pipeline processing. But some types of data still (and will) have to be processed using LIA, e.g. L03, fringed data, debiased data.

Use of LIA is a preferred option for improving the dark current removal and the absolute responsivity correction.

Use of LIA is essential for resolving instrument problems affecting off-axis and extended sources (fringing), faint sources, bright sources (debiased observations) and unvalidated L03 FP data.

CIA This strongly depends on the type of observation. In most cases, for staring observations, OLP can be used directly. The same is valid for non-faint raster observations or CVFs with 1.5 and 3" PFOV, for which distortion effects are negligible.

PIA Indispensable. Main reason is given by the complexity of instrument and calibration.

ISAP In general yes.

IRSKY NA

B.15 Were there any changes to the package to use archive results?

OSIA No.

LIA No.

CIA CIA was modified to cope with changed product definitions (e.g. corrected pointing vs standard (filtered) pointing).

PIA No.

ISAP Yes, it uses survey products.

IRSKY After the mission, we used the total Obs log to show what had been observed

B.16 What lessons have you learned? And would you do things differently if you were starting writing it again?

OSIA See legacy paper.

LIA 1. LIA routines should have been made available earlier, and such a need for interactive tools recognised during planning of the off-line processing pipeline, so that the same data reduction algorithms can be used in both.

2. LIA and OLP code should be written in the same language so that there is no need to convert code, i.e. no need for FORTRAN−− > IDL

3. Code should also be platform independent, i.e no need for VAX/VMX−− > Unix

4. The data format to be read by the software should be a standard, widely recognised format i.e. no need for MCFITS −− > FITS

5. Involve the data users more directly when data becomes available so that user designed tools can be included.

CIA 1. Start early

2. Review requirements and classify them from really needed to unrealistic dreams.

3. Keep it simple.

4. Have a healthy mix of s/w engineers and astronomers.

6. Have enough manpower to keep headers and documentation up-to-date.

7. Try to avoid "doublons" — routines coded by different persons, which perform essentially the same job .

8. Enforce the same naming conventions and layout for all high-level routines.

9. Think about distribution mechanisms between development sites early.

10. Colocation during the definition/early development phase and frequent workshops are essential.

PIA First question: Too difficult to be answered by a questionnaire See papers published on PIA (specially on lessons learned).

Second question: If under the conditions of 1994 I wouldn't do it too differently. There are things on which I would put a stronger accent (eg. internal documentation). I would also implement some slightly different technical solutions in aspects of the data organization.

ISAP See Appendix A for a list of these.

IRSKY This, for example would never run on a PC. A choice of JAVA would be the modern decision. Interprocess communication had to be done by hand which was time consuming.

We used lessons from earlier mission - imbedded debugging printout was built in. An experienced developer knew how to write in a machine-independent manner.

Appendix C

ISO Calibration Legacy Conference Papers

The Calibration Legacy Conference, held at between 5 and 9 February 2001 at the ISO Data Centre, ESA-VILSPA, Spain, was a time when all developers presented papers on their IA packages. Their papers describe the IA systems themselves, cover the IA writers experiences, problems, successes etc, and will hopefully guide developers of other software packages. Also published is another paper on topics related to IAs.

These papers, published in ESA-SP 481, are also included as part of this report. The following papers are reprinted.

OSIA: The ISO SWS Data Analysis Software Systems, Ekkehard Wieprecht, et al.
CIA: The ISOCAM Interactive Analysis System CIA, Stephan Ott, et al.
LIA: LIA 10 – The Legacy LWS Interactive Analysis Package, Gerard Hutchinson, et al.
PIA: Development and distribution of data analysis tools: the ISO**P**HOT **I**nteractive **A**nalysis case, Carlos Gabriel
COCO: SWS COCO. Lessons learned about distributed multi-platform software development and configuration control, Rik Huygen, et al.

C.1 SWS Data Analysis Software Systems, E. Wieprecht et al.

1

THE ISO SWS DATA ANALYSIS SOFTWARE SYSTEMS

E. Wieprecht[1], B. Vandenbussche[3], D. Boxhoorn[2], D. Kester[2], F. Lahuis[2], O. H. Bauer[1], D. Beintema[2], H. Feuchtgruber[1], A. M. Heras[5], R. Huygen[3], D. Kunze[1], K. Leech[4], R. Lorente[4], D. Lutz[1], P. Morris[6], P. Roelfsema[2], A. Salama[4], R. Shipman[2], E. Sturm[1], N. Sym[2], E. Valentijn[2], and E. Wiezorrek[1]

[1]Max Planck Institut fuer extraterrestrische Physik, Giessenbachstrasse 1, D-85748 Garching, Germany
[2]Space Research Organization of the Netherlands, Postbus 800, NL-9700 AV, Groningen, The Netherlands
[3]Katholieke Universiteit Leuven, Instituut voor Sterrenkunde, Celestijnenlaan 200 B, B-3001 Heverlee, Belgium
[4]ISO Data Centre, Astrophysics Division, ESA, Villafranca del Castillo, P.O. Box 50727, 28080 Madrid, Spain
[5]European Space Agency - Astrophysics Division, Keplerlaan 1, NL-2201 AZ Noordwijk, The Netherlands
[6]California Institute of Technology , 1200 E. California Blvd. Pasadena CA 91125, USA

ABSTRACT

We present an overview about the SWS Data Analysis Software Systems, their respective features and the relations between the different systems. Specifically the development of the Data Analysis Software Systems and lessons learned from this development process will be discussed.

Key words: ISO, SWS, Software, Data Analysis

1. INTRODUCTION

The Short-Wavelength Spectrometer (SWS) is one of the four instruments on-board ESA's Infrared Space Observatory (ISO), launched on November 1995, successfully operating for nearly 29 months. The spectrometer covers the wavelength range of 2.38 to 45.2 μm with a resolution ranging from 1000 to 2000.

Different software packages have been developed by the SWS consortium to deal with different responsibilities of the instrument team. In order to monitor the behavior of the SWS in real-time, the *SWS Quick Look Analysis (QLA)* display a roughly calibrated spectrum, while the data from the instrument is being down-linked via the satellite telemetry or during instrument tests. The *SWS Interactive Analysis (SIA)* package was developed as the data reduction environment for the instrument specialists to characterize the instrument and translate that characterization in improved data analysis algorithms and calibration parameters. A sequence of standard data reduction steps formed the so-called *standard product generation pipeline*. All data produced by the SWS instrument are processed by this PIPELINE. Experience has shown that the standard PIPELINE result is often not sufficient to allow a reliable scientific analysis.

Several interactive tools in the *SIA* package were found to be very useful for the general observer. Those tools are gathered for distribution in a reduced version of the *SIA* package : The *Observers SWS Interactive Analysis (OSIA)*.

2. THE GLOBAL VIEW

Figure 1 show the SWS Software Systems and dependencies to each other. The SIA is the central system. It is separated into a development and production area. Within SIA the SWS PIPELINE and OSIA are developed. OSIA is an extraction of the production area, available to the community. SIA contains two PIPELINE versions; the most recent official SWS PIPELINE in the production area and

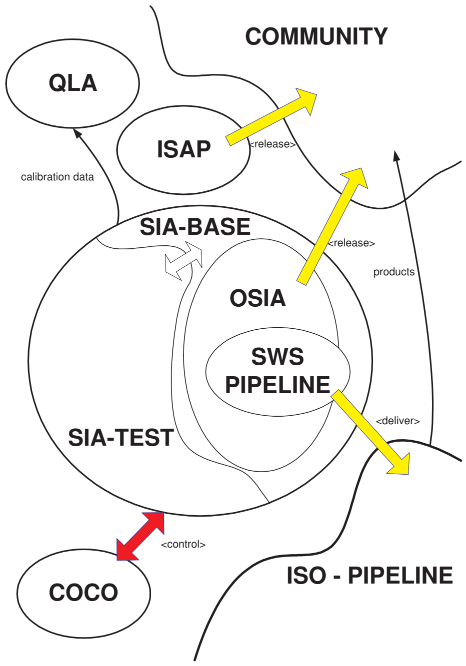

Figure 1. The SWS Software Systems

Figure C.1: The OSIA paper page 1.

the actual SWS PIPELINE including modifications in the development area. The validated SWS PIPELINE versions are delivered for integration into the common ISO PIPELINE.

SIA is controlled by a configuration control system, CoCo, (Huygen et al. 2001).

Although QLA contain similar or the same algorithms as the SWS PIPELINE and SIA, it was developed completely separately. QLA was used already prior to launch, for the instrument tests in the laboratory. SIA became available only shortly before launch with the functionalities required in the In-orbit-Calibration Document (IOCD), (Valentijn, Lutz, Salama 1995). Only calibration files, produced within SIA, were exported to QLA.

ISAP, the ISO Spectral Analysis Package, is a joint development by the LWS and SWS instrument teams, (Sturm et al. 1997). It was developed completely independent from other SWS systems. Special care was taken to guarantee a smooth interface to OSIA.

3. THE ISO SWS QUICKLOOK ANALYSIS

QLA was designed to allow a near Real Time monitoring of the instrument behavior. Pure IDL code was used to do basic PIPELINE processing steps and display data. The software is completely GUI driven.

QLA was mainly developed by two persons. As it was not possible to define QLA in detail beforehand, the software was developed in parallel with the usage of the system. The advantage was a very close interaction and quick turn around with the users who were just "looking over the shoulder" and setting requirements. For instrument tests and this kind of software it was very efficient not to communicate exclusively via documents between user and developer. The disadvantage was "grown code" and late documentation. But this caused never problems during the mission as QLA was only used by experts.

4. THE ISO SWS PIPELINE

The ISO PIPELINE automatically processes telemetry to scientific valid products for all instruments.

4.1. SWS DERIVE STANDARD PROCESSED DATA

After the instrument independent processing, where basically the telemetry file is split into a number of files in a certain time sequence, for every observation the instrument specific DERIVE SPD step is following. Here the data are processed in blocks of one reset interval :

- collect data
- determine usable data range
- correct reset pulse after effects
- measurement linearisation
- cross talk removal
- glitch recognition/removal
- compute grating/FP positions
- slope calculation
- convert to voltages
- assign wavelength
- write product

In almost every step FITS calibration files are used. The products are FITS SPDs (Standard Product Data) which contain for all 52 detectors a wavelength and a slope in Volts/second with some add on informations which are represented in bit code for each reset interval. The second result is a file containing further glitch information.

4.2. DERIVE AUTO ANALYSIS RESULT

This part of the pipeline covers the flux calibration. The modular design is even more transparent because every step is called with the complete data set of the whole observation to be processed.

- determine observation specific parameters
- dark current subtraction
- responsivity calibration
- flux conversion
- velocity correction
- write product

The resulting AAR (Auto Analysis Result) product is the starting point for ISAP, but within SIA and OSIA all further steps of scientific data processing can be taken as flat-fielding, sigma clipping, rebinning, unit conversions, etc.

4.3. PIPELINE DEVELOPMENT

The first version of the SWS PIPELINE was implemented in FORTRAN77 on a VAX/VMS. Nowadays SWS use pure IDL, embedded in the overall FORTRAN environment and Alpha/VMS. The adopted file format for the ISO PIPELINE is a non standard FITS format. This caused trouble throughout the whole mission and post mission.

The development started very early although only a minor number of requirements were known. Quite some time there was no interface available to access simulated data or data created during instrument tests. Therefore the SWS PIPELINE "saw" real data very late.

The development team was a mixture of programmers and scientists working closely together within SIA. It was essential to include scientists into the development process. This was also one of the reasons for the language change to IDL as scientists preferred to work with IDL rather than FORTRAN. Another reason is that IDL supports fast development/rapid prototyping. SWS had to pay for these comfort as IDL is an interpreter language and therefore not the optimal choice for PIPELINE systems. Also the IDL memory handling caused trouble.

Over the operational time the SWS PIPELINE had to cope with major design changes. Fortunately the system was flexible enough to allow this changes.

Figure C.2: The OSIA paper page 2.

5. The ISO SWS INTERACTIVE ANALYSIS

Originally it was intended to have a system which supports the following tasks :
- Debugging of pipeline software.
- Analyze the performance of SWS.
- Calibration
- Trouble shooting

After the development phase it turned out that SIA was used more and more as a very handy tool for Scientific Analysis of SWS data and more effort has been spent to add tools specifically for data analysis. Observers traveled to instrument data centers to get a training on SIA and use it for data reduction.

During the first phases of the development it was necessary to design an interface between the IDL routines and FORTRAN routines since the PIPELINE was implemented in FORTRAN77. This interface had to be robust against the differences of various hardware platforms and operating systems on which SIA was supported. This language mix complicated the system and was a permanent source of problems. Additionally SIA supports standard FITS format and therefore the PIPELINE interfaces are different. Nowadays SIA is pure IDL, still using the standard FITS format. SIA is designed as a tool-box (Figure

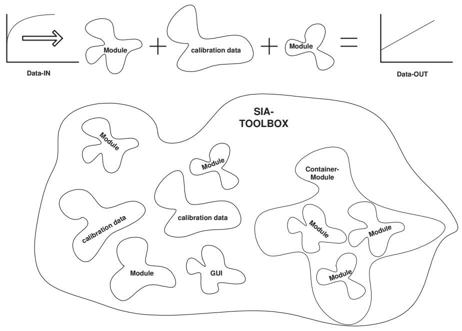

Figure 2. SWS Interactive Analysis Tool-box design

2). The tool-box contains all kinds of objects, like software, documentation, calibration files etc.. It is possible to combine objects in container objects. A very important property of this system is the easy plug-in of private modules into the environment. As IDL is an interpreter language, this requires no special effort like re-compiling etc. This architecture support combining tools for data processing or test software within a coherent system.

The tool-box design was implemented mostly as a command line driven system. But SIA contains also a number GUI driven container modules.

Today SIA contains about 1000 modules with 280 000 lines of code (including comments).

SIA was developed by a mixed team of about 20 instrument scientists and software engineers. Again the importance to include scientists into the development process has to be stressed.

SIA is more a living system and development environment than a data reduction system. Therefore it is not release driven.

The system was developed at different institutes by a geographically distributed group of developers. But the major progress was achieved when sitting together. Once the framework was coded the usage of regular meetings and teleconferences were essential to work together. The distributed development of SIA was controlled via CoCo.

SIA is based on a solid software design. Over the time the system grow a bit uncontrolled. It would have been better to assign one full person only to control the development.

PIPELINE deliveries were used as regular system tests of PIPELINE modules within SIA. The overall SIA system was not regularly tested.

6. ISO OBSERVERS SWS INTERACTIVE ANALYSIS

SIA installations have to consider different platforms (directory structures, setup procedures etc.) and all the hardware specifics, as sharable images (FORTRAN pipeline modules etc.) are called from the IDL shell. The embedding of the configuration control system require an installation and registration at the CoCo server and sites are required to name a person who takes the responsibility to maintain this SIA system. SIA is not version controlled.

Contrary to that OSIA is strictly version controlled which allows referencing on certain software versions. One of the major design aspects was to keep OSIA simple, platform independent and easy to install. Thus OSIA contains only pure IDL modules and FITS calibration files.

OSIA was issued after the mission. An earlier release would have been an important service to the community, which could have adapted earlier the successful pattern of visiting a datacenter once for training and then using OSIA at the home institute, with additional e-mail support.

OSIA is an extraction of modules for scientific data reduction from SIA. It supports data reduction from ERD (Edited Raw Data , starting point of PIPELINE DSPD) to AAR (Auto Analysis Result, spectrum). From AAR on it is possible to merge with ISAP for further data processing. OSIA is pure IDL and supports VMS and different UNIX flavors including LINUX.

OSIA contains the SWS PIPELINE which might be executed step wise. Many PIPELINE steps can be exchanged with interactive software to improve the result. OSIA supports SWS data product I/O, common FITS I/O, selecting, masking, combining, flat-fielding, defringing, rebinning, fitting, smoothing, line fitting, plotting, etc.

Figure C.3: The OSIA paper page 3.

OSIA is available from http://sws.ster.kuleuven.ac.be/osia

7. INFRARED SPECTROSCOPY ANALYSIS PACKAGE

ISAP, the data reduction system for the ISO spectrometers, is a joint development by the LWS and SWS Instrument Teams and Data Centers. Contributing institutes are CESR, IAS, IPAC, MPE, RAL and SRON.

For SWS ISAP support data processing from AAR onwards. This covers functionalities like averaging, arithmetic, subtraction, photometry, defringing, blackbody, line fitting, clipping, rebinning, smoothing etc. ISAP and OSIA can run in one IDL session.

About 15 developers wrote 280 IDL routines which have about 100 000 lines of code. 18.2 person years have been spent. More then 200 ISAP user have been registered so far.

Again the development was distributed, but regular workshops were essential for the success.

ISAP is available at http://www.ipac.caltech.edu/iso/isap

8. CONCLUSIONS

We draw a few conclusions for scientific software systems and their development. Of course, it depends on the type of system which conclusions apply.

8.1. PURPOSE

- Not the system is the aim. The data product should be the driver for software development, not a fancy system design.
- Use the systems from the beginning with real data. In case of SWS most of the software was not used already for pre-launch instrument tests.
- Make software available to users as soon as possible. In case of OSIA, this was done too late.

8.2. ENVIRONMENT

- An Interactive System environment should allow easy and quick plug-in of experimental modules. This was an essential advantage of the IDL environment.
- IDL, as interpreter language, was not the best choice for a PIPELINE kind of software

8.3. ARCHITECTURE

- Keep the framework as simple as possible
- Keep the framework flexible enough to support major design changes over time. SWS had to cope with major design changes for PIPELINE processing.
- The system should use standards as much as possible. The usage of non standard FITS caused enormous trouble.
- The architecture should support easy system tests.

8.4. DEVELOPMENT

- Standard software development principles do not always apply to scientific systems. E.g. requirements might be settled very late. The development runs in parallel with gaining experiences. This has to be considered and managed in the development process.
- Scientific Software Systems should be developed by a mixed team of instrument specialists and software engineers. Software engineers should also understand the instrument and instrument specialists contribute to software development. Both parties should act like a team together rather than having a customer-supplier relationship.
- Be flexible in the development process. The SWS QLA was quite different from the SIA development, but both were appropriate for the nature of the systems under development.
- Close interaction with users. The most efficient way to get requirements is direct contact to the users. This implies also to show users software prototypes as soon as they are available.
- Common Software development might delay the development process when it is not coordinated by a strong management. SWS had to wait for the common non standard FITS interface, which has been developed by another party as the schedules did not match. This is one of the reasons why SIA and PIPELINE wasn't used during instrument tests.
- Distributed development costs travel money and time. Regular workshops where developer met were essential for SWS systems.
- Distributed Software Development require significant management resources.

REFERENCES

Wieprecht E., Lahuis F., et. al., 1997, The Interaction of the ISO SWS Pipeline software and the ISO-SWS Interactive Analysis System, ADASS VII ASP Conference Series, Vol.145

Sturm E., Bauer O.H., Lutz D., Wieprecht E., Wiezorrek E., et al., 1997, The ISO Spectral Analysis Package ISAP, Astronomical Data Analysis Software and Systems (ADASS) VII (1997), ASP Conference Series, Vol. 145, 1998, R. Albrecht, R.N. Hook and H.A Bushhouse, eds.

Huygen R., Boxhoorn D., Roelfsema P.R., Sym N., Vandenbussche B., Wieprecht E., 2001, SWS CoCo : Lessons learned about distributed Software Configuration Control, ESA-SP-481, Conference Proceedings "The Calibration Legacy of the ISO Mission", this issue

E. Valentijn, D.Lutz, A. Salama, et al., 1995, ISO-SWS IOCD, ISO-94-23

Figure C.4: The OSIA paper page 4.

C.2 The ISOCAM Interactive Analysis System, S. Ott et al.

1

THE ISO*C*AM *I*NTERACTIVE *A*NALYSIS SYSTEM *CIA*: A REVIEW OF SEVEN YEARS OF DEVELOPMENT

S. Ott[1] & R. Gastaud[2],
S. Guest[3], M. Delaney[1,4], J. Sam-Lone[1,2], J-L. Starck[2], B. Ali[5], H. Aussel[2], A. Claret[2], D. Landriu[2], M-A. Miville-Deschênes[6], M. Sauvage[2] and F. Vivares[6],
on behalf of all our other collaborators[7]

[1]ISO Data Centre, Astrophysics Division, ESA, Villafranca del Castillo, P.O. Box 50727, 28080 Madrid, Spain
[2]CEA, DSM/DAPNIA, CE-Saclay, Gif-sur-Yvette, France
[3]RAL, Chilton, Didcot, Oxon, England
[4]Stockholm Observatory, Salsjöbaden, Sweden
[5]IPAC, JPL and Caltech, Pasadena, CA, USA
[6]IAS, CNRS, University of Paris Sud, Orsay, France

ABSTRACT

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

The system is mainly IDL based, with time consuming tasks coded in C++ and Fortran.

We will present the challenges faced in this multi-site, multi-platform project throughout the complete s/w development cycle, discuss

- initial design goals, their actual relevance and evolution
- chosen architecture and changes
- coding practise
- configuration control practise
- documentation
- schedule constraints
- human factor during development
- internal and external distribution
- feedback from the user's community

and present the lesson's learned during this project.

Key words: ISO, Software, Data Analysis, ISOCAM

1. INTRODUCTION

ESA's Infrared Space Observatory (ISO) is an astronomical satellite, operating at wavelength from 2.5 - 240 μm. ISO was launched in November 1995 by an Ariane 4 launcher, and operated successfully for 30 months, well above the requirement of 18 months.

The ISOCAM Interactive Analysis System (CIA) was developed to support the calibration of ISOCAM, the infrared camera on board ESA's Infrared Space Observatory (ISO) and to perform its astronomical data processing.

The development, a collaborative effort involving several institutes

- ESA/SSD (Villafranca del Castillo, Spain)
- CEA-SAp (Saclay, France)
- IAS (Orsay, France)
- IPAC (Pasadena, U.S.A.)

was led by ESA and started mid 1994. Currently the system is used by more than 80 institutes, including the ISO Data Centre at VilSpa and the CAM consortium.

The system is built upon the commercial Interactive Data Language (IDL), distributed by Research Systems Inc. CPU intensive tasks are coded in C++ and FORTRAN. It consists of 250.000 lines of code, more than 1000 modules, 150 MB of documentation and 80 MB of calibration products.

CIA is generally available to the astronomical community and runs, using IDL 5, under DEC VMS Alpha, Solaris, DEC Unix, Debian (PC) Linux and HP/UX. CIA can be obtained electronically via http://www.iso.vilspa.esa.es/archive/software/CIA/CIA.html

2. REQUIREMENTS

2.1. Initial requirements

The initial requirements on CIA were, in decreasing order of importance:

1. to calibrate ISOCAM
   - facilitating a frequent modification of calibration files
   - permitting a random access to data parts
   - logging all calibration activities for a later access by a data-base

[7] A. Abergel, B. Altieri, J-L. Auguéres, J-P. Bernard, A. Biviano, J .Blommaert, O. Boulade, F. Boulanger, C. Cesarsky, D. Cesarsky, P. Chanial, V. Charmandaris, R-R. Chary, C. Delattre, T. Deschamps, F-X. Désert P. Didelon, D. Elbaz, Y. Fuchs, P. Gallais, K. Ganga, G. Helou, M. Heemskerk, M. Kong, F. Lacombe, O. Laurent, P. Le Coupanec, J. Li, L. Metcalfe, K. Okumura, A. Pollock, P. Roman, D. Rouan, M. Rupen, R. Siebenmorgen, D. Tran, R. Tuffs, D. Van Buren, L. Vigroux, T. Võ & H. Wozniak

Proceedings of the Conference "The Calibration Legacy of the ISO Mission", VILSPA, Spain, February 2001 (ESA SP-481)

Figure C.5: The CIA paper page 1.

2. to perform fast data reduction to assess the ISOCAM performance during the Performance Verification phase of ISO
3. to monitor the health of ISOCAM
4. to provide the means to perform any sort of investigation requested for problem diagnostics
5. to assess the quality of ISOCAM pipeline data products
6. to debug and validate the ISOCAM pipeline
7. to be the development bench for data reduction algorithms (mainly for the ISOCAM pipeline)
8. to perform astronomical data-processing of ISOCAM data
9. to perform fast data reduction to help assessing the ISO performance during the Spacecraft Verification phase of ISO

2.2. Relevance and evolution of the requirements

While some requirements were expected to change with time, e.g. the evolution from a calibration/monitoring system for operational use within the project to an astronomical data analysis system, new requirements were introduced and the relevance of others changed:

1. CIA was the workhorse to calibrate ISOCAM during and beyond the mission. However
   - the frequency of modifications to calibration files was highly overestimated
   - the functionality to randomly access and merge data parts is rarely used by instrument experts
   - logging all calibration activities was found not to be necessary and consequently not performed
2. CIA was the development bench for data reduction algorithms. However, only few algorithms were found suitable to be employed automatically in the pipeline
3. as foreseen, the importance of ISOCAM scientific data reduction increased with time
4. following the preferences of astronomers, the new requirement that CIA should run under Unix was introduced

Requirement (1) drove a major part of the system design and a lot of manpower was spent implementing it. However, the resulting complex data structures needed special access functions. As users felt that the data was hidden from them, standard, directly accessible IDL structures had to be implemented (see also section 5).

3. SCHEDULE AND OTHER CONSTRAINTS

For historical reasons VAX/VMS was chosen as operating system for downlink processing within the ISO project. Furthermore the decision was made to have the data files within the processing environment not in generic, but in a project-specific variant of FITS. Therefore, in order to stay as close as possible to the ISOCAM pipeline, the operational CIA version had to run under VMS.
The schedule was extremely tight — especially as the development team had to build up IDL and later C++ knowledge:

start of IA development July 1994
IA readiness review April 1995
launch of ISO 17 November 1995
ISOCAM first light 28 November 1995
start of instrument commissioning 9 December 1995
delivery of first flight calibration file 10 December 1995
delivery of first CIA version December 1996

4. DEVELOPMENT/DISTRIBUTION POLICY

During the early development phase, all contributors were located at SAp, Saclay, France. With the approaching launch of ISO, part of the development team had to move to ISO's operation centre at Villafranca del Castillo, Spain.

In order to keep the system on track, CIA's Configuration Control Board (CCB) discusses and prioritises software changes.

4.1. Development Tools

IDL from Research System Inc., an interpreted language, was chosen as the coding language. The alternative was to use FORTRAN (a compiled language) with an astronomical library, for example MIDAS.
The advantages for IDL are

1. some team members had IDL experience
2. astronomical and scientific libraries for IDL exist
3. IDL is easy to learn
4. coding with IDL is quick (at least three times quicker than C++), and therefore permits fast prototyping
5. in principle, IDL is independent from the operating system and therefore portable

Its drawbacks are:

1. IDL is a commercial products, with expensive licenses
2. IDL is not regulated by IEEE standards
3. some language changes from one version to the next were not backward compatible

Actual configuration control was performed using a script built on the functionality provided by VMS' Code Management System (CMS).

4.2. Tests and versions

To avoid bad surprises to the users (who expect a proven system), both a floating development version and a frozen user's version are provided. This is now done by a cursorily tested monthly Unix version. In case of problems with this version, older versions remain accessible as a fall-back solution.

Figure C.6: The CIA paper page 2.

Before the annual "community" version is released, it is tested for several months, which ensures a high quality standard for the software.

4.3. Distribution

Until a mirroring system, which updates the main development sites at VilSpa and IAS from the VMS master version, located at Saclay, was installed, updating was done manually or by re-installation a new version delivered on a DAT tape. This procedure proved cumbersome and prone to errors.
The distribution policy changed over time:

– CIA Version 1 (December 1996) and CIA Version 2 (November 1997) were distributed only after a hands-on training of an astronomer (who would become the CIA contact point for his institute) at Saclay, VilSpa or IPAC
– As the experience how to handle ISOCAM data grew, the confidence into CIA's algorithms increased and its documentation improved, it was felt that a hands-on training was not necessary any more. Therefore CIA Version 3 (October 1998) was available upon request
– As of CIA Version 4 (April 2000), anybody interested can register and download CIA from the WWW

As we maintain a list of registered users, we can inform them about new versions and could invite them to fill out a questionnaire to get their feedback.

5. LESSONS LEARNED

In this section we list the lessons learned during the CIA development. Many of our experiences are shared by the developers of the other interactive analysis systems (Leech K et al.).

5.1. Technical side

1. Choose visible solutions. Don't have routines, which try to perform everything (black boxes). Don't hide data inside internal data structures, if these data will be viewed or directly manipulated by the users
2. Choose simple solutions. Let practical, and not theoretical reasons drive your system. Minimise the number of your principal data structures
3. Use generic language features wherever possible. Do not try to implement a shell over the chosen language in order to get an improved programming language only because some of your requirements are not fulfilled yet. Quite likely the language will evolve, and have these features in a later version
4. Be flexible. Data structures will evolve. Do not hard-code them, but have the possibility to add or delete tags on the fly
5. Be complete. Do not omit original telemetry information because you think it will never be needed. These information might be essential by more advanced data analysis algorithms later, and switch-cases will have to be implemented to treat data where this information is missing
6. Adopt and follow naming and calling conventions. Also widgets should have consistent layout and button conventions. While a collaborative approach is essential for developing a major data analysis system, all contributors have to understand that their efforts go into a common system, which has to look coherent for the user
7. Tag the processing history to the product. Don't rely on loosely coupled journal files to understand the data treatment
8. Think from the beginning about portability issues — it will avoid a lot of un-elegant coding later
9. Benefit from standard packages (e.g. ASTROLIB (Landsman W, 2001)). If you have to modify the code to implement additional features or to fix bugs, discuss the modifications with the author and feed the changed code back as soon as possible
10. Share your code with the community. Distribute standalone tools, as it will save work for somebody else, and for you as it is debugged and might be improved by somebody else
11. Taxonomy of the code must be simple. The tree of directories should be well balanced. Try to divide the main directory into seven to twelve sub-directories, with no more than two levels and less than 60 files per directory
12. Software development tools must be flexible and easy to use. Advertise them to collaborators
13. Create, update, and distribute among the programmers a catalog of tips and pitfalls of the coding language (e.g. one previous IDL feature under Unix was that the language did not permit more than 128 tags for a structure)
14. Maintenance of module headers is a Sisyphus task which needs dedicated manpower
15. Build tools (scripts or programs) to create simulated input data for testing the processing. These data are useful before the launch (because no real data are available) and also after the launch because real data sets might be too big or too complex to verify one single processing step
16. Build test scripts with representative and checked input and verified data, and use it before each release
17. Avoid "doublons" — different routines written by different authors which fulfill similar tasks and could have been merged into one routine

5.2. Managerial side

1. The cost of software development is frequently underestimated by higher management. Good and generally

Figure C.7: The CIA paper page 3.

4

usable software is not a free by-product of calibration or scientific work of one person

2. In the beginning, keep the team in one place — preferably in the same building, in the same floor
3. Have a healthy mix of astronomers and software engineers. In the beginning of a project a good ratio is 1:1. Do not let it slip below 2:1
4. Match tasks with interest. Choose the right mix of people
5. Multi-site developments are difficult. A lot of efforts are needed to maintain the communication. Have frequent (≈ monthly) workshops
6. Encourage the feedback of users. While the questionnaire helped to confirm that our users were pretty content (in average, 8 out of 10 points were awarded for software and documentation), we should have had more user's meetings

6. DOCUMENTATION

Documentation is as important as the code.

1. A good user's manual is critical to the success of the software package. It should be written by a typical user (an astronomer), preferably a native English speaker. Our choice was LaTeX and an ASCII dump to enable searches.[8] Today's preferences would be PDF and HTML
2. A paper reference manual, listing and describing all functions and their arguments is useless for a big programming system. Provide this information in an electronic format (HTML) instead, with hyper-links to subroutines
3. The most important functions should have a commandline help: After entering the routine name, a short description of the calling syntax is returned
4. Have a widget-based tool, which provides keyword-driven online help of all available routines

REFERENCES

Cesarsky C. et al., 1996, ISOCAM in flight, A&A 315, L27

Delaney M. (ed), 2000, ISOCAM Interactive Analysis User's Manual, Version 4.0, ESA Document, Reference Number SAI/96-5226/Dc

Kessler M. et al., 1996, The Infrared Space Observatory, A&A 315, L32

Landsman W., 2001, IDL Astronomy Library – a collection of low-level routines, mainly related to astronomy, written in the commercial language IDL, http://idlastro.gsfc.nasa.gov/homepage.html

Leech K. et al., 2001, The interactive analysis working group final report, ESA Document, Reference Number SAI/2000-008/Dc

Miville-Deschênes M-A et al., 2000, Optimizing ISOCAM data processing using spatial redundancy, A&A supplement, 146, 519, 2000

Metcalfe L., 2001, The ISOCAM Calibration Strategy, ESA-SP-481, Conference Proceedings "The Calibration Legacy of the ISO Mission", this issue

Ott S. et al., 1997, Design and Implementation of CIA, the ISOCAM Interactive Analysis System, ADASS VI, ASP Conference Series, Vol. 125, 34

Pérault M., 2001, CAM Pre-launch Calibration, ESA-SP-481, Conference Proceedings "The Calibration Legacy of the ISO Mission", this issue

Pollock A.M.T. et al., 2001, The CAM Automatic Data Analysis Pipeline, ESA-SP-481, Conference Proceedings "The Calibration Legacy of the ISO Mission", this issue

[8] An index would have been preferable. However, as it is very time consuming to generate an index, regrettably there was not enough manpower available to do so.

Figure C.8: The CIA paper page 4.

C.3 LIA 10 – The Legacy LIA Package, G. Hutchinson et al.

1

LIA 10 - THE LEGACY LWS INTERACTIVE ANALYSIS PACKAGE.

M.G. Hutchinson[1], **S.D. Sidher**[1,3], **B.M. Swinyard**[1], **S.J. Chan**[1], **C. Lloyd**[1], **A.S. Harwood**[1], **E.T. Polehampton**[1,4], **S. D. Lord**[2], **S. Molinari**[2,5], **J. Brauher**[2], **S.J. Leeks**[3], **and S. Unger**[3]

[1]UK ISO Data Centre, Space Science Department, Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, U.K.
[2]Infrared Processing and Analysis Center, Caltech, U.S.
[3]Queen Mary, University of London, Mile End Road, London E1 4NS, U.K.
[4]Astrophysics Department, Oxford University, Keble Road, Oxford OX1 3RH, U.K.
[5]Istituto di Fisica dello Spazio Interplanetario – CNR, Via del Fosso del Cavaliere, 00133 Roma, Italy

ABSTRACT

With the release of the Infrared Space Observatory Legacy Archive and Infrared Space Observatory Off-Line Processing pipeline version 10, in 2001, the Long Wavelength Spectrometer Interactive Analysis (LIA) software package version 10, will also be released.

LIA routines are available for post-processing of LWS data. LIA version 10 routines will be directly compatible with the Off-Line Processing pipeline version 10 products.

We outline some of the more commonly used LIA routines, particularly those which have been released over the past year, highlighting their use.

Key words: ISO – LWS: Software, Interactive Analysis, Data Reduction

1. INTRODUCTION

The Infrared Space Observatory (ISO) Long Wavelength Spectrometer (LWS) (Clegg et al. 1996) was one of four instruments onboard ESA's ISO satellite launched in November 1995. Infrared spectral data were obtained by the LWS between 43 and 197 μm at medium ($\lambda/\Delta\lambda \sim 200$) resolution in grating mode and high ($\lambda/\Delta\lambda \sim 8000$) resolution in Fabry-Pérot (FP) mode. The LWS data are pipeline processed and the Legacy Archive version – Off-Line Processing pipeline version 10 (OLP 10) – of this database will be released by the ISO Data Archive (IDA) at Villafranca, Spain, accessible at:

http://www.iso.vilspa.esa.es/ida/index.html.

The LWS Interactive Analysis (LIA) software allows users to inspect, and if necessary, optimally calibrate their OLP produced LWS data.

The final reduction of the calibrated data is then carried out interactively using the ISO Spectral Analysis Package (ISAP). ISAP was developed for both ISO Short Wavelength Spectrometer data and LWS data by programmers at a number of institutes and is maintained by the Infrared Processing and Analysis Center (IPAC).

As with ISAP, LIA has been developed over several years by a team of programmers. For LIA they were based at the Rutherford Appleton Laboratory (RAL) and IPAC. ISAP was available shortly after the launch of ISO. Users were introduced to LIA in 1997 (Sidher et al. 1997).

The LIA software has been written using Research Systems Inc. Interactive Data Language (IDL), typically offering the user a Graphical User Interface (GUI) to aid data manipulation. LIA is released and maintained on the internet by the ISO UK Data Centre, at RAL, with a US mirror site at IPAC.

When a new version of the LWS OLP was released, the concurrent version of LIA was updated to include the new calibration files and algorithms, as well as introducing new routines.

LIA version 10 (LIA 10) will be the appropriate tool for use with OLP 10 data, the Legacy ISO data set; both are scheduled to be released concurrently in 2001.

2. EXAMPLES OF LIA ROUTINES

LIA routines are intended to allow users to effectively process their LWS data and to resolve the calibration issues not dealt with by the pipeline processing, e.g. fringed data, uncalibrated L03 FP data (L03 – FP data obtained over an extended wavelength range), etc.

A full list of LIA routines with user guides and examples of usage can be found under the *software* link from the main LWS home page:

http://jackal.bnsc.rl.ac.uk/isouk/lws/lws.html

which is also mirrored at IPAC:

http://www.ipac.caltech.edu/iso/lws/lws.html

Routines have been designed to allow users to inspect, reprocess, and recalibrate their LWS data. Routines available within each category are listed and appropriate examples of a few commonly used routines are highlighted as an illustration.

2.1. Inspection Routines

These allow the user to inspect the Standard Processed Data (SPD) and Auto Analysis Result (AAR) pipeline products.

INSPECT_IPD – displays illuminator flashes and dark currents.

Proceedings of the Conference "The Calibration Legacy of the ISO Mission", VILSPA, Spain, February 2001 (ESA SP-481)

Figure C.9: The LIA paper page 1.

INSPECT_ABS – displays the absolute responsivity correction factors generated from the illuminator flashes.

INSPECT_RASTER – displays the pointing direction (RA, Dec) for observation types, including raster mapping observations.

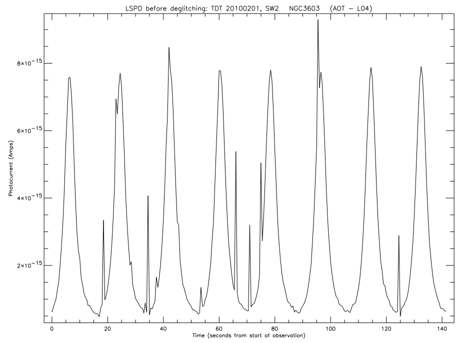

Figure 1. SPD data of NGC 3603 at TDT 20100201, obtained in L04 FP mode with detector SW2 before *deglitching. Glitches show up as intermittant spikes.*

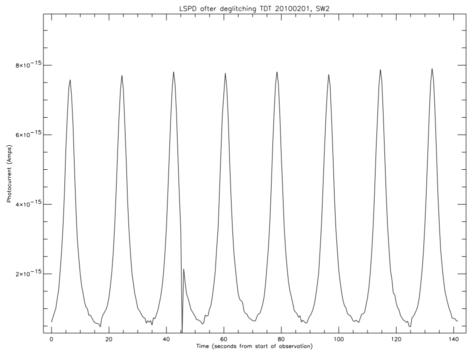

Figure 2. SPD data of NGC 3603 at TDT 20100201, obtained in L04 FP mode with detector SW2 after *deglitching.*

INSPECT_SPD – Inspects SPD *before* and *after* deglitching. Figure 1 gives an indication of how badly raw LWS data were affected by glitches, most of which were removed by the standard pipeline processing, Figure 2. The user can therefore see the effectiveness of the pipeline deglitching for their data. Non-removal of glitches results in spurious spike-like features in the calibrated spectrum.

INSPECT_DRIFT – displays SPD data *before* and *after* the responsivity drift correction is applied. In cases where there are offsets in the photocurrent at the same wavelength between different scans.

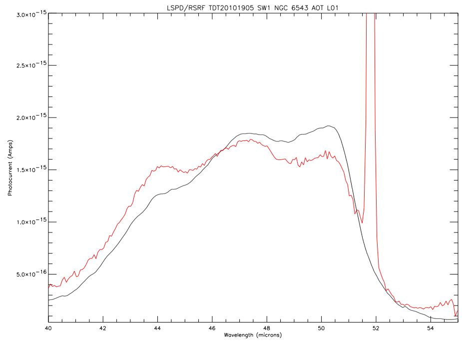

Figure 3. SPD (grey line) of NGC 6543 and the RSRF (dark line) at TDT 20101905, obtained in L01 grating mode with detector SW1.

INSPECT_RSRF – displays the SPD and Relative Spectral Response Function (RSRF). This is useful for identifying features in the spectrum which are suspect and may have arisen from dividing by the RSRF, which may itself contain spurious features. An example of such a comparison is shown in Figure 3; the RSRF is the darker line.

2.2. Interactive Routines

These allow the user control over the data reduction process and thereby customise the algorithms used for the specific calibration problem(s) affecting their observation.

IA_DARK – allows the user to remove the dark current interactively; particularly useful for determining the fluxes of faint sources. See the contribution by Chan et al. (2001).

IA_DRIFT – applies the responsivity drift correction, by interpolating a polynomial between user selected key points, as shown in Figure 4.

IA_ABSCORR – allows the user to refine the absolute responsivity correction factors.

GUI_FPMF – Interactive FP Multi Gaussian Fitting tool for a "blended" line in L04 FP data.

2.3. Recalibration Routines

These allow the user to recalibrate the data where necessary, either interactively, via a GUI interface, or non-interactively using calibration parameters different from those used as the defaults by the automated OLP pipeline.

DEFRINGE – This algorithm re-generates the fringe pattern casued by the LWS optics (Lloyd 2001b, Swinyard

Figure C.10: The LIA paper page 2.

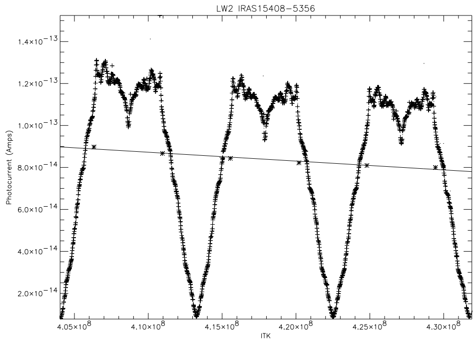

Figure 4. Drift correction applied to data (crosses) of IRAS15408-5356 at TDT 48202406, obtained in L01 grating mode with detector LW2, interpolating first order polynomial (solid line) between key points (asterisks).

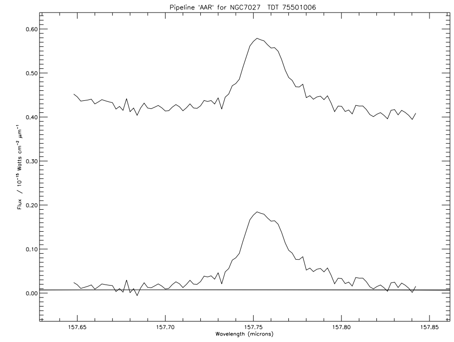

Figure 6. FP_Proc correction using measured dark current applied to original fixed dark corrected data (uppermost) of NGC 7027 at TDT 75501006, obtained in L04 FP mode with detector SW4. The lower plot is the dark current corrected data superimposed on the optimum L01 grating baseline.

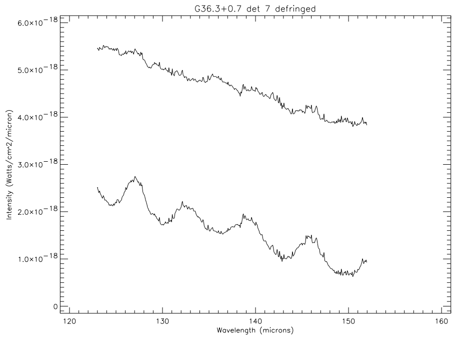

Figure 5. Defringed data of G36.3+0.7 at TDT 30003207, obtained in L01 grating mode with detector SW3. Defringed data (uppermost) and the original fringed data are offset in flux for ease of comparison.

et al., 1998) and then removes it from the data, resulting in well calibrated data, as shown in Figure 5.

FP_Proc – Gives the user various options to improve their L03 or L04 FP data (L04 – FP data obtained at specified line wavelengths), e.g. using user-defined dark currents and/or correcting for the uncertainty in the grating position (Swinyard et al., 1998). Both flux and wavelength can be recalibrated using data generated by the user after having used IA_drift and/or IA_abscorr. At present, running FP_Proc is the only suitable way of reducing and calibrating L03 FP data, a procedure primarily intended for specialist users.

Figure 6 shows an example of how the L04 FP continuum can be adjusted in FP_Proc by changing to the measured, or a user-defined, dark current, which then typically better matches the LIA optimised L01 grating continuum (L01 – grating data obtained over the whole LWS wavelength range).

Short_AAL – Similar functionality to FP_Proc, i.e. recalibrates wavelength and flux, but for grating data only.

3. LIA 10

LIA 10 will introduce some new routines, as well as updating some existing routines. Upon release, the LIA web pages will include full details of the new and updated routines. It is anticipated that the LIA 10 routines should resolve most of the remaining LWS calibration issues.

3.1. Updated Routines

GUI_FPMF – Gives the user greater flexibility in FP multi-fitting. Primarily intended for specialist users.

Defringe – A new, more robust defringing algorithm is being developed. See Lloyd (2001b) for more details.

Inspect_SPD – This utility will have more features enabling the user to list the number of scans, the length of scan, etc.

FP_Proc – This will incorporate a new throughput correction, improved grating profiles, order sorting and makes non-prime FP data available – these improvements should allow users to produce a greatly improved calibration of their L03 FP data. See Polehampton et al. (2001) for more details.

Figure C.11: The LIA paper page 3.

4

3.2. New routines

SS_corr – For very bright sources, the LW1–4 detectors were found to become non-linear due to debiasing, and for such sources, a non-linear correction for strong sources has to be applied. Users will be able to apply the correction to data obtained with 1/4 second ramps, although data which were obtained using 1/2 second ramps will have to be re-processed as 1/4 second ramps by staff at the UK ISO Data Centre first. See Leeks et al. (2001) for more details.

FP_order – Some of the LWS FP data suffer from order contamination, where adjacent FP orders overlap slightly between $43 - 47\mu$m and $83 - 90\mu$m. Where such data exist in an observation, they are highlighted and removed.

Beam_corr – Utility to improve the beam profile, taking account of beam problems. Fringing will still be removed using defringe. Beam_corr should be used in conjunction with the relevant beam profile data which is published in the LWS Handbook (Gry et al. 2000). See Lloyd (2001a) for more details.

Average – Automatically averages the data, using the algorithm applied within ISAP. Useful for averaging a large number of datasets for a statistical analysis, although as some bad data may not be eliminated, this routine is not recommended for processing individual observations, which is best done interactively within ISAP.

IRAS_phot – Calculates IRAS synthetic photometry data for L01 grating observations.

NIR_leak – Will correct for the near-infrared leak problem. See the contribution by Chan et al. (2001).

4. LESSONS LEARNED

No matter how well a project is planned, unforseeable problems almost always occur, and ISO was no exception. Some significant lessons have been learned, particularly in the development of LIA to resolve some of the problems which could not be solved by the Off-Line Processing, and these are now briefly highlighted, with reference to future similar missions, such as Herschel (Pilbratt 2001), formerly FIRST.

The need for interactive tools for users should be anticipated and made available as early in the mission as is practicable.

With the LWS, as with the other instruments on ISO, the pipeline processing software was written in FORTRAN, whilst the interactive LIA software was written in IDL. Hence, even if the same data processing algorithm was used in LIA as in OLP, code had to be completely rewritten in order to convert it into IDL. Both pipeline and interactive analysis software should be written in a common language, ideally being modular. Such an object orientated approach, using Java, is envisaged for Herschel, where Off-Line Processing and interactive analysis will form a single system.

Also software should be platform independent. With LWS, the initial OLP and LIA code was written to run on VAX/VMS systems, and later had to be ported to run on Unix systems. Again, Java being platform independent, would seem to be an ideal solution.

During the ISO mission the standard data format was changed from MCFITS to FITS format, due to the non-portability of MCFITS onto Unix. This meant that some software had to be changed. A widely used, flexible data format should be selected from the outset.

When the pipeline processing suite is being put together, it is important to recognize that not all calibration issues can necessarily be resolved in this way. Some degree of manual intervention using interactive analysis tools will invariably be required. Planning for this should help the two types of software to be written in such a way that they form a single, seamless software package.

Nobody is likely to know the nature of the sources being observed as well as the observers themselves. Hence, involving the observer at an early stage in the data processing, e.g. contributing data reduction algorithms, could be very beneficial to the mission planning, allowing calibration issues to be identified earlier and overcome potentially more quickly, and/or with a more strategic, scientific solution. Hence closer links with the astronomical community should always be encouraged.

Further information about using LIA can be obtained from either of the first two authors:
g.hutchinson@rl.ac.uk
s.d.sidher@rl.ac.uk

REFERENCES

Chan S.J., Lloyd, C., Swinyard B.M. & Lim T.L., 2001, This volume.
Clegg P.E., et al., 1996, A&A 315, L38.
Gry C., et al., ISO LWS Handbook, volume IV, 2000. ESA SAI-99-077/Dc.
Leeks S.J., Swinyard, B.M. & Clegg, P.E., 2001, This volume.
Lloyd C., 2001a, This volume.
Lloyd C., 2001b, This volume.
Pilbratt, G.L., 2001, This volume.
Polehampton E.T., Swinyard B.M. & Sidher S.D, 2001, This volume.
Sidher S.D., Swinyard B.M., Harwood, A.S., Lord, S.D. & Unger, S.J., 1997, First ISO workshop on Analytical Spectroscopy, ESA SP-419, 297.
Swinyard, B.M. et al., 1998, Infrared Astronomical Instrumentation, SPIE 3354, 888.

Figure C.12: The LIA paper page 4.

C.4 PIA, C. Gabriel

1

DEVELOPMENT AND DISTRIBUTION OF DATA ANALYSIS TOOLS: THE ISOPHOT INTERACTIVE ANALYSIS CASE

C. Gabriel

ISO Data Centre, Astrophysics Division, ESA, Villafranca del Castillo, P.O. Box 50727, 28080 Madrid, Spain
Now at the XMM-Newton Science Operations Centre, same location

ABSTRACT

The different interactive data analysis software packages ("xIA": CIA, LIA, PIA, SIA) developed for the ISO instruments were conceived primarily as calibration tools. However, all of them were used more and more for astronomical data reduction throughout the mission, imposing new requirements on user friendliness, documentation, distribution, compatibility and user help systems. The strategies followed in the case of the ISO**P**HOT **I**nteractive **A**nalysis as well as the achieved experiences, valuable in every current and future observatory, will be discussed in this paper.

1. INTRODUCTION

The original ideas and plans for the four individual interactive analysis systems corresponding to the four ISO instruments CIA (Gastaud et al. 2001), LIA (Hutchinson et al. 2001), PIA (Gabriel et al. 1997a) and SIA (Wieprecht et al. 2001), were in the beginning quite different. The necessity of a flexible software package for instrument monitoring, calibration files derivation and products checking was the common denominator. Despite of the primary design, all xIAs were increasingly used during the mission for scientific data analysis.

This gradual change was reflected also in a change by the xIA's user profile: originally corresponding almost exclusively to an instrument expert (who is possibly at the same time developer of the tools), it is later replaced to the profile of a general user with no deep knowledge neither on the instrument nor on the data structures.

The rational behind this metamorphosis is the exploitation of the large commonalties in the core functions of data reduction between calibration and astronomical analysis, taking advantage of the necessary flexibility of calibration / monitoring tools. The "price" to be paid for the development of a calibration tool to a scientific analysis package is given by the rather different requirements concerning user friendliness, help systems, distribution, documentation, compatibility across different systems but also of the orientation / focus of the developers.

2. THE GENERAL ISO EXPERIENCE WITH IAs

The absolute necessity of interactive calibration tools was recognized in the ISO project development rather late. IA tools were considered for a long time a second priority with respect to automatic pipeline systems for data reduction.

The primary conception of scientific instrument builders, who deliver them for integration to ESA, calibrated on ground, and together with data reduction packages for in-flight calibration, was not only unrealistic. It hindered in addition the early development of those tools in a collaborative manner between ESA and the instruments' consortia.

Around fifteen months before the nominal ISO launch, the situation was that very poorly calibrated instruments were delivered to ESA, who was supposed to operate them in flight with a low level instrumental knowledge. The non existence of calibration tools was a further difficulty. The qualitative change cutting this gordian knot was the formation of mixed teams with personal from the instruments' consortia and ESA, the "Instrument Dedicated Teams (IDTs)"(Gabriel 1993). They should provide on the one side the instrumental knowledge cumulated through assembly of instrument and the on-ground characterization experience. On the other side they should get the general experience in space astronomy, the resources and the view / interests of the general astronomical community necessary for the success of the envisaged operations.

It was in the framework of these IDTs that the development of calibration / interactive analysis tools began to crystallise. However, time and resources constrains brought all four ISO IDTs to follow different development paths for their IAs, with almost no interaction among the development teams in this respect. The result: four complete different and independent systems. A remarkable commonality was that all IAs were IDL[1] based. The most important reason for this was probably the possibility of building very fast extensible data reduction packages at very low costs.

The independent approach by the different teams made it possible to reach a very high efficiency in the beginning of the S/W development, since the IA tools developed focused on specific calibration and monitoring of the own instrument. However, in the long term this approach re-

[1] IDL is a trademark of Research Systems, Inc.

Proceedings of the Conference "The Calibration Legacy of the ISO Mission", VILSPA, Spain, February 2001 (ESA SP-481)

Figure C.13: The PIA paper page 1.

sulted in quadruplicating efforts in several aspects. Another disadvantage is that users of different ISO instruments, eg those performing cross calibration analysis, were forced to deal with complete different analysis tools. Now that all the IA's have developed further more or less to general analysis packages this is the situation with which a "general" ISO user is confronted, after eg retrieving data from several instruments from the archive.

3. THE PARTICULAR PIA EXPERIENCE

The most distinctive element in the particular PIA development (with respect to the other ISO IA's) is probably the fact that PIA was from the very beginning conceived as both calibration tool and general scientific analysis of ISOPHOT data. The foundation documents (baseline requirements in Gabriel et al. 1994a and architectural design in Gabriel 1994b) already foresaw PIA's development up to a full data analysis package to be used by the astronomical community.

The necessary elements of user friendliness, including full graphical user interfaces, were in this sense planned form the very beginning, and were reflected also in the design of the calibration tools integrated in the package.

Concerning the distribution of the package and help and information systems, the very early decision of creating a general analysis tool gave also in these aspects PIA a pioneering rle within the ISO project. So was PIA the first ISO software package distributed to the astronomical community for wide use (from July 1996 on, 4 months after ISO launch). It was also the first one fully documented using hypertext (coincident with the first public release) as well as the first one using the web as the source for distributing package and documentation to as well as communicating with the users. In this sense it was also showing the way to go to the ISO Data Archive (Arviset et al. 2000)).

Since an early paper (Gabriel et al. 1997b) already summarizes the main lessons learnt in the development of PIA, we will concentrate here in some aspects which could not be discussed in that one:

– a general historical description of the development, which should situate the reader in the context of the ISO mission and its different stages,
– the internal data organization, which is in our opinion the key element for the PIA versatility,
– PIA's graphical user interfaces, perceived by most of the users as the differentiating element for the user friendliness of the package.

3.1. PIA's historical development

The first PIA development team was constituted in the framework of the ISOPHOT IDT one year before the planned ISO launch, in April 1994. The ISOPHOT instrument was delivered to ESA for integration in the ISO payload module 4 months before, after an extensive on-ground calibration campaign. This could conclude with a characterization of the different ISOPHOT sub-instruments and a proof of the reproducibility of the data taken, but leaving the main aspects of instrument calibration to the on-board calibration.

The already referenced foundation documents (Gabriel et al. 1994a, ??) together with a first workplan (Gabriel & Klaas 1994c) were the basis for the work. A small prototype built in the months before the official development start helped for the crystallisation of several ideas expressed in the documents. However, real coding could not be started until the next IDL version, already announced begin of 1994, was publicly released. The reason was the absolute necessity of pointers (virtual addressing of data structures), the main change introduced to IDL V3.6, for the internal data handling foreseen for PIA. This aspect will be extensively covered in the next section.

Six months of work by the PIA development team resulted in the first ISOPHOT consortium internally distributed version, PIA V1.0. This version was used for ground data calibration analysis (using data taken during 1993), as well as for Assembly and Integration Verification tests (AIV) performed after each integration phase of the ISO payload. Intensive feedback was achieved from these campaigns concerning the reliability and usability of the software.

The operational version (V2.0) was ready in September 1995, just in time for the (delayed) ISO launch, two months later. It contained the tested sequences for derivation of all ISOPHOT OLP calibration files, including their formal correctness. Intensive regression testing was designed and performed successfully also in this part of a mission, which was characterized by extreme time constraints, caused by the limited ISO life expectation.

A period of many upgrades (and consequently many internal versions) started with the ISO launch and the analysis of real sky data, together with the first instrument calibration steps. It concluded with the first public release of PIA, (V4.5). The decision of distributing externally the package (not officially considered before) was based on several elements:

– the maturity of PIA V4.5: providing the users with an easy to use package for processing raw ISOPHOT data including deep handling of all calibration steps up to final products to be used for scientific analysis,
– the complementing good documentation quality: which made possible to the users, together with the general user friendliness of the package, the work from the very beginning without experts' help, in the own home institutes, and without training,
– the extremely simple and reliable installation of the package: PIA was distributed compiled in binary version for different IDL versions and operating systems,

Figure C.14: The PIA paper page 2.

the installation required just decompression of one distributed file,

– the anticipated impossibility of doing science directly with the official OLP products: the pipeline products were conceived as examples of reduced data for giving the observer the impression of the scientific contents of the observation performed. For real scientific analysis, interactive work on the data is essential, given the instrument complexity, the flux dependence of many effects with data ranging over 5 orders of magnitude in flux, and the difficulties related to several aspects of the ISOPHOT calibration.

Since that first public distributed version, PIA went through many changes and upgrades reflected in a large number of versions released. The main source of the changes has been the deeper understanding of the instrument calibration, with better and more sophisticated algorithms for correction of instrumental effects. In addition to that the general user friendliness of the package was continuously increased as well as the capabilities of "final" work on products (image processing, spectral fitting, combination of diverse observations). A data simulator was also developed under PIA Gabriel et al. 2000 for a better understanding of ISOPHOT observations. Finally, a special data package analysis for the most demanding ISOPHOT observing mode in terms of observational complexity (AOT P32) was adapted and integrated into PIA (Tuffs & Gabriel). The whole PIA source code was made available to the community together with PIA V9.0, in July 2000 with PIA V9.0.

The actual PIA version, V9.1, composed by 850 files with almost 2500 routines and 220000 lines of pure IDL code, is considered almost final. Only very minor updates are foreseen in the future, for coping with upgrades in OLP V10 (the final ISO official pipeline) as well as for adapting it to future versions of IDL, if this is considered necessary.

The total investment on PIA resulted in approximately 11 man years of work, spent by three institutions: ESA's Astrophysics Division ($\tilde{7}0\%$), ISOPHOT consortium ($\tilde{2}5\%$) and IPAC ($\tilde{5}\%$), in a good example of successful collaboration.

3.2. PIA's internal data organization

The heart of PIA (and probably one of its strongest points) is its internal data organization. In order to discuss this more closely a couple of terms have to be clarified:

– an ISOPHOT **observation** is defined as the data gathered from an ISO spacecraft pointing in the sky for a continuous period of time,

– an ISOPHOT **measurement** corresponds to a continuous period of time within an observation taking data, in which the instrument is fixed configured, eg using one sub-instrument with a given filter,

– because of the chopping capability of ISOPHOT, different (but close) sky positions can be part of one single measurement (eg of a source and its neighbour background).

One of the main points concerning data access in the first requirements document established the necessity of processing and visualizing several aspects of different observations simultaneously. This, together with the fact that the average observation with ISOPHOT is not too demanding in terms of data volume, decided us to define an architecture based on multiple measurements kept in memory, once they are read. Furthermore, all the products of data reduction of one measurement should be kept also in memory. In this scheme, all the read data and their products are persisting in memory for easy and fast access, until they are explicitly deleted from it.

The measurement should be considered the basic data unit, understanding it as the sum of relevant data corresponding to this period. This includes voltages and temperatures measured, but also position of the focal plane chopper, time elapsed, etc, together with main information defining it, like sub-instrument, filter and polarizer used, etc. All these data are combined in a multi-variable data entity (IDL "structure"), which fulfils the needed capability of combining single values with one and two dimensional arrays, as the different variables of a measurement are.

Several layers of data reduction ("buffers") are defined flatly across instruments and observation types according to principal steps of processing the fundamental data variable. Starting from its primary raw stage, voltage measured from the integrating capacitor in the Cold Readout Electronics (CRE) circuit of an ISOPHOT detector, finishing by the final stage, flux density or surface brightness observed on a sky position.

Every data buffer can be filled up with as many measurements on this level of reduction as desired. The measurements are identified by a keyword, which is a combination of sub-instrument, the unique ISO target observation identifier (TDT) and the position of the measurement within that observation. However, this does not represent a limitation for a measurement being several times within a buffer. This is especially important for keeping representations of the same measurement reduced through different methods / parameters to a certain level.

While the structures of a certain buffer are completely defined a-priori with respect to the variables contained, the actual dimensions of these variables are not predefined, since they depend principally on the time the measurement took and the sub-instrument used. The possibility of piling up different measurements within a buffer was offered first by IDL with the introduction of a type of pointers ("handles"). So, technically speaking, a buffer of a certain reduction level is an array of structures, each of them composed of several fix dimension variables and addresses in memory. These addresses point to variable length one- or two-dimensional arrays.

Figure C.15: The PIA paper page 3.

4

The fact that all the data are kept in memory in clearly addressable entities makes possible to the user an easy simultaneous and fast access of in the one or the other manner related data. Examples are looking (and analyzing) the same measurement on different levels of reduction, but also different measurements of the same (or similar) object(s), or just related data by any thinkable element.

This internal data organization is the real basis of the extremely high user friendliness of PIA. The easy build up of graphical user interfaces upon this basement is a consequence of it and will be discussed in the next section.

Figure 1 is a scheme of PIA data reduction. Every level goes to the next one through a reduction process (in certain cases different processes are possible for the same type of reduction), on every level several data corrections are possible, for cleaning up data without affecting their structure. The central role played by PIA in the development of the calibration and data processing of ISOPHOT data is clearly visible through the colour differentiation. There are processing steps defined prior to the existence of PIA(in turquoise), developed using PIA and then exported to the OLP (blue) and the ones only possible within PIA (red).

3.3. PIA's Graphical User Interfaces (GUI)

Although PIA can be run fully command line driven, only a small fraction of PIA users use command line possibilities in marginal parts of their data reduction, while the vast majority run it exclusively through its GUIs. This picture is given by the results of a questionnaire sent around among PIA users, which showed a high level of satisfaction. One of the most common answers in the reasons for satisfaction with PIA is even its GUI system. This not only demonstrates the user friendliness of this approach, but also the completeness of the GUI, covering ALL aspects of the data handling, correction, calibration and information provided by PIA.

We can classify the different GUIs in PIA in four classes:

- **file access and general data handling**: for accessing the different files and internal buffer types.
- **data level full access**: for visualizing all the variables of a measurement on a certain level of reduction, initiating data corrections, accessing related information, defining selection criteria, processing to the next level of reduction. There is one GUI specific for every data reduction level. However, a very high common use of s/w is made, because of the common internal data organization. Also to this type of GUIs should count the interfaces for accessing and handling of calibration tables.
- **dialogue interfaces**: a high use of dialogue GUIs allows the user eg to set parameters on corrections to be applied to the data and judge the results of these parameters use in form of text and / or plots.

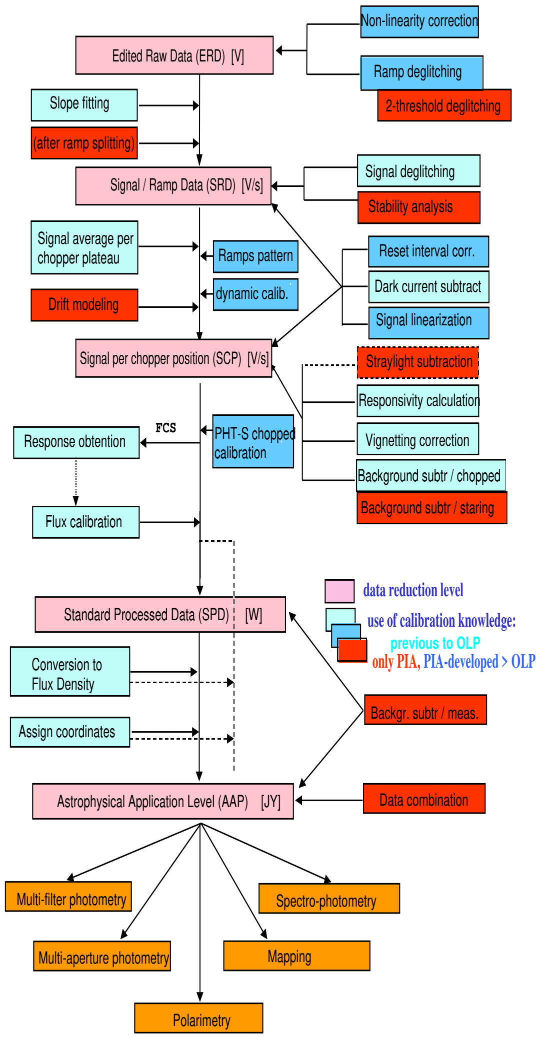

Figure 1. PIA data reduction scheme

- **general purpose GUIs**: many of the high graphical capabilities of IDL for plotting and image display, but also the different possibilities for input and output, are made easily accessible through interfaces facilitating their use. Although they have special features for PIA use, they are not PIA specific and can be used by any kind of data.

The question if a s/w package better include GUIs or not is, in our opinion, fundamentally a question of how

Figure C.16: The PIA paper page 4.

necessary these are and how expensive they are to create. The same s/w package is ALWAYS better if it provides GUIs and does not force the user to read lots of documentation to know which are the necessary parameters, calls, etc, for performing different steps of data analysis [2].

The more complex the data to be handled the more necessary are user interfaces. ISOPHOT, several instruments under one name, with a very large complexity in several items, and therefore, with the corresponding complex data structures and observational modes, is an eminent case.

Finally, the question of the costs for implementing PIA's GUIs can be measured in two different ways, in comparison to:
a) the time saved to users eg in reading documentation, making mistakes and coming out of situations caused by misinterpretation,
b) to the general implementation costs of the rest of the package.
In our opinion the point (a) is more than enough justification for the "GUI approach" in the case of a package like PIA. Concerning (b) our case shows that, on the basis of a good defined architecture in terms of internal data administration, only a relative small add is necessary to make the package really user friendly. IDL is one alternative for this approach, since it offers the integrated possibility of building easily graphical user interfaces. Our estimation concerning costs in the relation GUI / rest of PIA is of the order 10-15%. A difficulty for such an estimation is that the upgrading work of the whole package, which was continuously going on during the ISO mission in all the different phases, is also made easier and more efficient to the s/w developers through integrated graphical user interfaces.

The importance given by users to good and comprehensive GUIs as part of a software package is reflected in the high ranking of this aspect in the general (excellent) judgement of PIA, as obtained from a questionnaire sent to users.

4. Conclusions

A strong common element of the interactive analysis packages developed for ISO was, in addition to the late recognition of the necessity of their development as calibration tools, their later use as astronomical data analysis packages. This evolution, which has taken place from a certain point of the development of the packages to now, 4 years after the end of ISO operations, has implied several changes in the user friendliness of the package, in its documentation, distribution and support.

PIA is a good example of planning for that evolution from the very beginning. This was reflected from the definition of the different development phases up to the development team profile at the different stages. The result is a user friendly data analysis package, used in more than 200 astronomy institutes all over the world, used, partly or exclusively for almost all scientific publications based on ISOPHOT data so far.

[2] Of course there are some examples of bad and misleading GUIs, but we do not count PIA GUIs among these.

ACKNOWLEDGEMENTS

The main acknowledgement of the author should go to the whole PIA development team members, which not only did an excellent job, but also had a lot of fun working together. Madame and sirs, it has been really a pleasure to work together with you:
Jose Acosta-Pulido (IDC, now at IAC, Spain)
Wai Ming Tai (DIAS, Dublin, Ireland)
Detlef Skaley (MPIK, Heidelberg, Germany, now at MPIfR in Bonn, Germany)
Ingolf Heinrichsen (MPIK, Heidelberg, Germany, now at IPAC, Caltech, USA)
Huw Morris (RAL, Didcot, UK)
Paola Merluzzi (AIP, Postdam, Germany)

REFERENCES

Arviset, C., Dowson, J., Hernndez, J., Plug, A., Osuna, P., Pollock, A., & Saxton, R. D. 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco: ASP), 191

Gabriel C., 1993, "An Instrument Dedicated Team for ISOPHOT: Operations, Development, Workload", Internal MPIA document

Gabriel C., Klaas U. and Laureijs R.J., 1994, "The minimum Interactive Analysis for PHT - Scope and Baseline Requirements", MPIA document RS-PHT-0000 MP/03

Gabriel C., 1994, "The minimum Interactive Analysis for PHT - Architectural Design", MPIA document TN-PHT-0000 MP/04

Gabriel C., Klaas U., 1994, "The minimum Interactive Analysis for PHT - Workplan", MPIA document PL-PHT-0000 MP/04

Gabriel, C., et al, 1997, in ASP Conf. Ser. Vol. 125, Astronomical Data Analysis Software and Systems VI, ed. G. Hunt & H.E.Payne, (San Francisco: ASP), 108

Gabriel, C. and Heinrichsen, I., 1997, in ESA SP-401, The Far Infrared and Submillimetre Universe, 261

Gabriel, C., Acosta-Pulido, J. A., & Hur, M. 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veilet, D. Crabtree (San Francisco: ASP), 619

Gastaud R., Ott S. et al, 2001, The ISOCAM Interactive Analysis System CIA: a review of seven years of development, *this conference*

Hutchinson et al., 2001, LWS Interactive Analysis Software, *this conference*

Tuffs R. and Gabriel C., in preparation

Wieprecht et al, 2001, The ISO SWS data analysis software systems, *this conference*

Figure C.17: The PIA paper page 5.

C.5 SWS COCO. Lessons learned, Rik Huygen et al.

1

SWS COCO: LESSONS LEARNED ABOUT DISTRIBUTED MULTI-PLATFORM SOFTWARE DEVELOPMENT AND CONFIGURATION CONTROL

Rik Huygen[1], Danny Boxhoorn[2], Peter Roelfsema[2], Nico Sym[2], Bart Vandenbussche[1], and Ekkehard Wieprecht[3]

[1]Katholieke Universiteit Leuven, Instituut voor Sterrenkunde, Belgium
[2]Stichting Ruimteonderzoek Nederland, SRON, The Netherlands
[3]Max Planck Institüt für Extraterrestrische Physik, MPE-Garching, Germany

ABSTRACT

This paper describes how the ISO-SWS development team developed the Interactive Analysis (IA) software in a distributed environment. When it became clear that IA would be developed by at least three institutes that were geographically distributed, a platform-independent configuration control system (CoCo) was designed that could control the software development in terms of version control and access control, and distribute the software in a consistent and automatic way. The CoCo system incorporates also tracking of problem reports.

Over the years the development team has gained experience in distributed software development and maintenance. The lessons learned from this experience are discussed.

Key words: ISO SWS Distributed Configuration Control

1. INTRODUCTION

Instrument scientists and observers of the Short-Wavelength Spectrometer (SWS) (De Grauw et al. 1996) aboard the Infrared Space observatory (ISO) (Kessler et al. 1996) use a dedicated software package called SWS Interactive Analysis (SIA) (Wieprecht et al. 2001). The ISO-SWS IA was developed for several years and is still maintained in a geographically distributed environment and on a variety of computer hardware and operating systems (VAX-VMS, AXP-VMS, HP-UX, Solaris and OSF). Teams from the SWS consortium institutes but also other astronomical institutes actively participated in this development. In order to manage and control the development a software configuration and version control system (CoCo) was designed. CoCo had to be system independent, portable and apply a client-server technology. The system controls all SIA software items like sources, documentation, calibration parameter files and calibration reference data. The different files that form one logical software item are grouped in a CoCo module. The resolution element for access control and distribution is the CoCo module.

2. SWS INTERACTIVE ANALYSIS DEVELOPMENT

The SWS Interactive Analysis and other SWS software systems are described in Wieprecht et al. 2001. This paper emphasises on how we developed SIA more than what it is and how it's used.

The SIA system consists of three environments: a stable production environment (the so-called BASE version), a shared development and test environment (the TEST version), and the private test environment for the individual user (USER). The user or developer works in her private USER area to develop and test individual new modules or changes to functionality of existing modules. When a module is sufficiently tested by the developer it is put in TEST i.e. the module is distributed to all contributing sites and made available to all SIA developers. The TEST level is a playground for developers and instrument specialists. TEST is the most up-to-date system to work in but is not considered stable and is therefore not distributed to normal SIA users. The module is then used and tested by all developers in different platforms and on different kinds of data for some time. After it is confirmed to be stable on component level and within the system, a configuration control board (CCB) decides that the module can migrate to BASE i.e. be brought into production. This decision is not based on bureaucratic rules but on indepth knowledge of the system under development. The BASE versions of the system are configuration controlled. In order to avoid concurrent changes to the same version of a module, modules can only be changed by a developer after they have been reserved by her.

Conflicts in configurations for BASE that were discovered while assembling a BASE configuration were efficiently fed back to the development team. Feedback of overlap in functionality between modules and inconsistency in implementations and coding standards was not very efficient. The reason for this was the lack of a software co-ordination group that controlled the development.

Since the development was geographically distributed, all sites had to be up-to-date at all time. CoCo provides several mechanisms to accomplish this (1) the possibility to manually update the whole system or individual modules, (2) overnight distribution of new or updated modules and (3) overnight automatic system rebuilds after a full system update. Updates include documentation and data since these are considered as normal CoCo objects. Each

Proceedings of the Conference "The Calibration Legacy of the ISO Mission", VILSPA, Spain, February 2001 (ESA SP-481)

Figure C.18: The COCO paper page 1.

client keeps track of its own update status and compares this with the server. Requests to update always are initiated by the clients. The development environment is thus identical for each developer independent of his or her location.

The design of SIA was done by a small dedicated group. The flexibility and the Toolbox design (as described in Wieprecht et al. 2001) allowed a clear separation of tasks and therefore promoted distributed development. At regular time we held TIGER TEAM meetings that were organised to solve well defined problems by specialists in the field. These meetings were dedicated to one subject and were held in isolation for not more than a few days.

SIA was not distributed outside the Core SWS Development Team i.e. to the astronomical community for several reasons: (1) the system was not designed for easy distribution of software releases and (2) the functionality of the SIA system exceeds the need of the general observer. Stable versions of SIA modules containing useful functionalities have been extracted into a dedicated software package called 'Observers SWS IA (OSIA)'. The OSIA distribution is release-driven.

3. VERSION AND CONFIGURATION CONTROL

The main reason for version control of software and documentation was traceability and reconstruction of older BASE level environments. Version control was implemented by a locking mechanism on CoCo objects, i.e. on a module that contains several files, not on an individual file. Core developers have to reserve a module before making changes. Reserving a module puts an exclusive lock on it. Inserting a module back into the system releases the lock and makes the module available for distribution to all participating sites.

SIA is not released software, it is a continuous development and slowly grows in functionality, capacity and quality. Every day there are just a few modules changed but for the basic purpose of SIA i.e. calibration software these minor changes can have a great impact on data reduction. Configuration control of SIA is established through the TEST and BASE level strategy where the BASE level consists of a snapshot of a consistent set of stable modules. This could be older stable versions of modules than those currently in the TEST level.

4. CONFIGURATION CONTROL ISSUES

This section summarises the lessons learned on configuration control issues.

- Version and configuration control is a must, especially in a distributed development. The tree-level system we used i.e. USER-TEST-BASE turned out to be quite adequate for a relatively small development group. Building stable configurations (that can be compared to software releases) was still a sizeable job.
- Distributed configuration control requires a maintenance manager at each site. It is not so much the work that counts here but the assigned responsibility to maintain the local environment. Maintenance on a daily basis didn't require more than 20% of someone's time.
- Documentation must be up-to-date and synchronised with source code and between sites at all times during the development. New functionalities and small software changes that are not well documented and synchronised with the code will not be tested and used. The effort spent in the development will be wasted.
- For releasing production software a configuration control board (CCB) should be established. For small and simple systems this CCB could consist of just a few people sitting together and decide which software modules are stable enough and consistent with each other to be released.
- Do not leave for holidays the day after a release. Users expect maintenance and support especially shortly after a software release. Because of the diversity in computer hardware, operating systems and framework software (we used IDL as our basic reduction environment on which SIA is build) it is impossible to foresee all possible situations while assembling a configuration.
- Dependencies should be resolved and maintained from the start. This is very useful for building configurations. In a system with several hundred modules it is impossible to resolve all dependencies by hand.
- Creation of new modules should be controlled and co-ordinated. The simplest solution for this is to set up a software co-ordination group that examines the system in terms of coding standards, overlapping functionality and commonality.
- Track changes and keep statistics. This helps in controlling the parts of your system that undergo most (frequent) changes and thus identify the weakest parts of your software.

5. DISTRIBUTED DEVELOPMENT ISSUES

This section summarises the lessons learned on distributed development issues. The problems described here are more on an organisational and managerial level.

- Provide a well defined forum for distributed development. Software development where small groups of developers are geographically distributed is difficult to manage on the level of human interactions. It soon becomes completely hopeless if resources have to be spent on tasks that can be performed by underlying infrastructure software providing automatic code distribution, version and configuration control as described in section 3.
- Keep the development team small and staff your project with the right people at the right time.
- Regular meetings and short term co-location are a must.

Figure C.19: The COCO paper page 2.

- Train and educate your developers. Training keeps your developers up to date with new technologies and might prevent them from running away for more exciting jobs. Education is needed to organise and orient your team in one and the same direction following the same approach in design, implementation, support and maintenance.
- Management should stick to management tasks. Beware of political decisions driving your design and development. Technical decisions should be made by people with knowledge of the problem domain and an overview of the affected system.

6. OVERALL SOFTWARE DEVELOPMENT ISSUES

This section summarises the lessons learned on pure software development issues. Many of these items may sound familiar or like forcing an open door but practice has shown that the most obvious items are most often forgotten or bluntly ignored.

- Keep the design of your system as simple as possible.
- Define clear data structures (or classes) and interfaces. In SIA there are only a few robust data structures defined to describe the data of an observation at different stages in the data processing. For these structures an infrastructure library is available that provides input/output routines, plotting and typical operations like spectral merging or rebinning. Scientists associates with the SWS consortium found it natural to make the prototypes of their data reduction procedures operate on these structures. This facilitated a fast integration in the SIA system.
- Set firm development standards and guidelines early in the project.
- Users, not developers, should drive the prioritisation of new system features. Quite some features available in our CoCo system were never used by the software developers. The development cost involved in adding these features to the system might seem marginal, but it increases the complexity and hence the maintenance cost.
- Administrators of the system are also users of the system. Capture their requirements early in the development and take them into account when making your system design. Administration should not be confused with software maintenance.
- Do a retrospective at each major milestone in your development. Involve users at any stage in the development and give them an active role. This will focus your attention to the ultimate goal of your development and give you the opportunity to make the necessary corrections in planning and development.

ACKNOWLEDGEMENTS

We like to thank all those who supported us during development and maintenance of this system, especially the members of the SWS Instrument Dedicated Team (SIDT) and the SWS Instrument Support Teams (SIST) who constantly use the system and continuously give feedback on its use and possible improvements.

REFERENCES

De Graauw Th. et al., 1996, Observing with the ISO Short-Wavelength Spectrometer, A&A 315, L49.

Kessler M. et al., 1996, The Infrared Space Observatory, A&A 315, L27.

Wieprecht E. et al., 2001, The ISO SWS Data Analysis Software Systems, ESA-SP-481, Conference Proceedings "The Calibration Legacy Mission", this issue.

Huygen R. et al., 1997, A Configuration Control and Software Management System for Distributed Multiplatform Software Development, Astronomical Data Analysis Software and Systems VI, ASP Conference Series, Vol. 125.

Figure C.20: The COCO paper page 3.