MINUTES OF THE OBST / ISDT WORKSHOP, 6TH AND 7TH APRIL, IMPERIAL COLLEGE LONDON

Participants:
Matt Fox, Helen Bright, Mattia Vaccari, Kenton D'Mellow, Steve Guest, Trevor Fulton, Evanthia Hatziminaoglou, Christian Surace, Pasquale Panuzzo, Sarah Leeks, Ta nya Lim, Ken King, Sunil Sidher, Dave Clements

The meeting did not strictly follow the original agenda, so these minutes are organised according the order that issues arose during the meeting.

Quality Control

The meeting started with a short discussion of quality control. DC stated that there needs to be a standard way of communicating quality control, which needs to be integrated. MF stated that we need to know what ESA requires in terms of quality control.

Schedule

MF introduced the expected schedule for the data processing pipeline over the next few months (see viewgraph). He stated that we hope to have a simple product design thought out over the next few days.

DC stated that the pipeline structure needs to be in place before real data is sent through it.

It was stressed that every stage of the pipeline impacts on every other stage - and we need to know the form of the products passed between them.

There was a discussion of who should be responsible for acceptance-testing the software. KK stated that it should be the OBST accepting the software team code, with possibly a user group as well.

KK stated that the delivery of the next iteration of common IA software will impact on the SPIRE IA schedule. SG stressed that we must make our requirements on common IA known because mini iteration 5#4 is not well defined yet, so there is room for requests. SG is the main contact for new functionality requests, or SCRs can be used for more stable code. The SPIRE SPR / SCR system will be used for our own problem reports. KK suggested that the SPIRE SPR /SCR system could be used as a means of requesting HCSS / common IA functionality, which could be forwarded on by SG. SL stated that as master tester she saw prioritising such requests as part of her role, in collaboration with SG.

DC stated that contacts like this need to be documented. Imperial will be the information centre for SPIRE. A WIKI will be set up. HB warned that we should have one person controlling the WIKI structure or there is a danger that it can get complicated and unusable. There was a discussion as to whether a message board could be used in parallel, or whether email was better.

Returning to the schedule, KK pointed out that discussion and development of the processing steps was missing from the schedule. MF stated that the 4 or 5 experts on the instrument need to meet and discuss these.

Scope of Data Processing (see viewgraph)

MF stated that it is hard to know when a data product stops being SPIRE specific, and that data products should be easily exportable. DC stated that producing a data product that can be used by IDL, IRAF etc is a good enough way to validate the process and instrument. These products should therefore be the stopping point. SG stated that it is a financial rather than a user requirement to stop processing at a particular point. He also pointed out that we may be able to use code from other instruments, especially PACS. MF restated that the philosophy of the pipeline should be that it is pluggable together.

Dataflow (see viewgraph)

SPIRE data reduction should be able to be done either from the GUI or the command line. It is possible that making a particular GUI selection will generate the correct command at the command line. The GUI will guide the user through the pipeline steps.

Use of Java and Jython

MF stated that the pipeline is a Jython script with a Java core. Java has the benefit of stricter controls and type checking, so is best for long term maintainable code. SG pointed out that Java also provides better performance than Jython. SG showed the SPIRE statement to ESA with regard to use of Java and Jython. Jython should be used for scripting and prototyping, and Java for products, lower level algorithms and mathematical functions. The general astronomer is not expected to know Java. There was a discussion as to whether the general astronomer should be encouraged to edit the Jython scripts, and it was concluded that it depended on their level of expertise.

Coding Standards

SG stated that we will use the ESA coding standards available on the HSDT web page. MF stated that coding standards for Jython are still under discussion. SG stressed that it is important to use the same naming conventions in Java as Jython.

Error Messaging

It was stated that the usefulness or otherwise of error messages annoys many people. Coders were encouraged to write useful error messages. MF stressed that error messaging and exception handling were different concepts. There was a discussion as to where in the GUI or command line error messages would appear. It was concluded that this would be configurable by the user. It was discussed whether the existing logging system could be used by Quality Control. There was also a discussion of how Quality Control messages would be passed along the pipeline - as part of the history of the product. There was discussion as to whether checking the history for such messages was part of the Jython script or part of the core Java modules. SG was keen for it to be part of the modules. However, finding such an error message should mean the issue of a warning, perhaps through the logging system, rather than an outright refusal to continue with processing.

Help system

HB stated that the common IA command line help was still being produced, but that GUI Help using JavaHelp was well developed and known about. SG pointed out that the Java toString() method can be useful as a help mechanism. TF asked if there was a template to fill out in terms of providing help for a particular task, and HB answered that there wasn't yet. SG suggested the possibility of an automated process on JavaDoc to generate HTML help. KDM suggested that there could be a required help method for each task.

Export and Import

MF stated that at every stage of the pipeline the products should be exportable to FITS format. As products can be more complex than the FITS format, a certain amount of selection will be needed in order to do this. There was a discussion as to whether a product could be saved locally for development purposes, but there is no means to do this currently. KK pointed out that serialization can cause problems as it changes in implementation between different Java releases. Saving products locally may be a requirement on the HCSS if it is needed for complex products - simple ones can easily be saved to FITS. KDM stated that we need a list of all the products in the system at any point, so that imports and exports can be designed for each of them.

Java and Jython

SG stated that Java can easily be called from Jython, but you should only call Jython from Java if you have very good reason to do so, and if you know how to do it. HB pointed out that if you need to call Jython from Java, this should be a hint that you ought to think about converting your Jython code to Java, because it has become reusable.

GUI System

MF demonstrated his vision of the integrated GUI system. There was a discussion of the relative importance of a friendly and usable interface as compared to a help system. It was concluded that both were very important in terms of the user's overall impression of the usefulness of the software.

The Task class

MF demonstrated the basics of a Task as conceived by JJM (see viewgraph). This included the basic parts of a Task - preamble, execution and postamble, and an explanation of the inputs and outputs. Core algorithms are Java classes, but they may have non-HCSS specific inputs and outputs. They have a wider Herschel-IA applicability.

A Jython code example of a Task was shown by MF. Some Jython shortcut notations were noted.

The Product Class

SG gave a demonstration of the concept of a Product class in IA. (see viewgraph and SG's doc circulated before the meeting). SG stated that Product classes are designed for use in both Java and Jython, which makes them very easy to use. He also mentioned some useful Jython shortcuts. A Product class was defined as containing a description, metadata, zero or more datasets and a processing history, although this is not currently well defined. It roughly corresponds to a FITS file, but has more flexibility. There are readers and writers to translate between Product classes and external formats.

SG then went on to demonstrate the concepts of Dataset, Data, and Metadata (see the viewgraph for details).

Dictionaries are available to translate to and from FITS files. Keywords for products should follow Java conventions - the dictionaries will translate to FITS. Dictionaries can be added if they are needed. At the moment only IA FITS files can be imported, but soon it will be possible to import others as well.

Task coding demo

MF gave a demonstration of coding a Task in both languages (see viewgraph for details). An important point, stressed by SG, is that the default output of a Task is the first one defined in your signature. The three important stages in defining a Task are:

- inherit
- inputs and outputs
- execute

Description of a SPIRE data product

KK demonstrated a hierarchical description of an example SPIRE data product (see appendix). Other products will probably be modelled on this.

PHOT and SPEC Processing Pipelines

A preliminary list of product types was drawn from a discussion of Photometer and FTS pipelines. The pipelines are reproduced here, alongside a complete list of the products and their acronyms(?!), even if not used explicitly in the pipeline.

PHOT Pipeline:

Generic Products

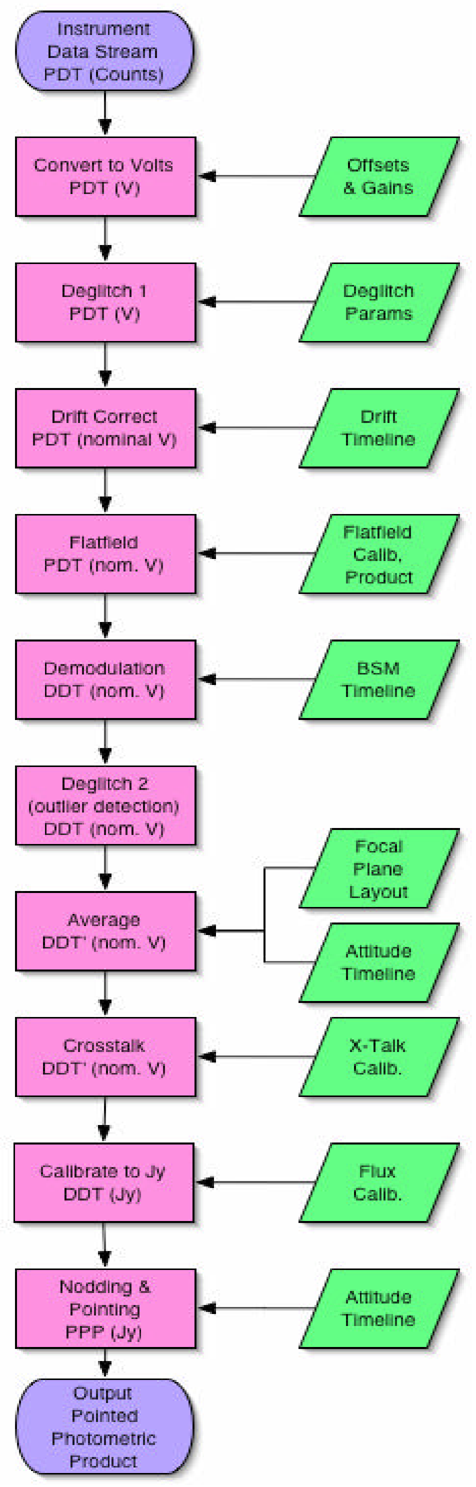
Attitude Timeline (AT)
Beam Steering Mirror Timeline (BSMT)
Secondary Mech. Timeline (SMECT)
SCU Timeline (SCUT)
Spacecraft Housekeeping Timeline(SHT)
Events Timeline (ET)
Orbit Timeline (OT)
Calibration Products (entire class)

Phot Pipeline Specific Products

Pointed Phot product (PPP)
Phot Detector Timeline (PDT)
Demodulated Detector Timeline (DDT)
Time Averaged DDT (DDT')

Notes

PPP has signal and position on sky for each bolometer.



FTS Pipeline:

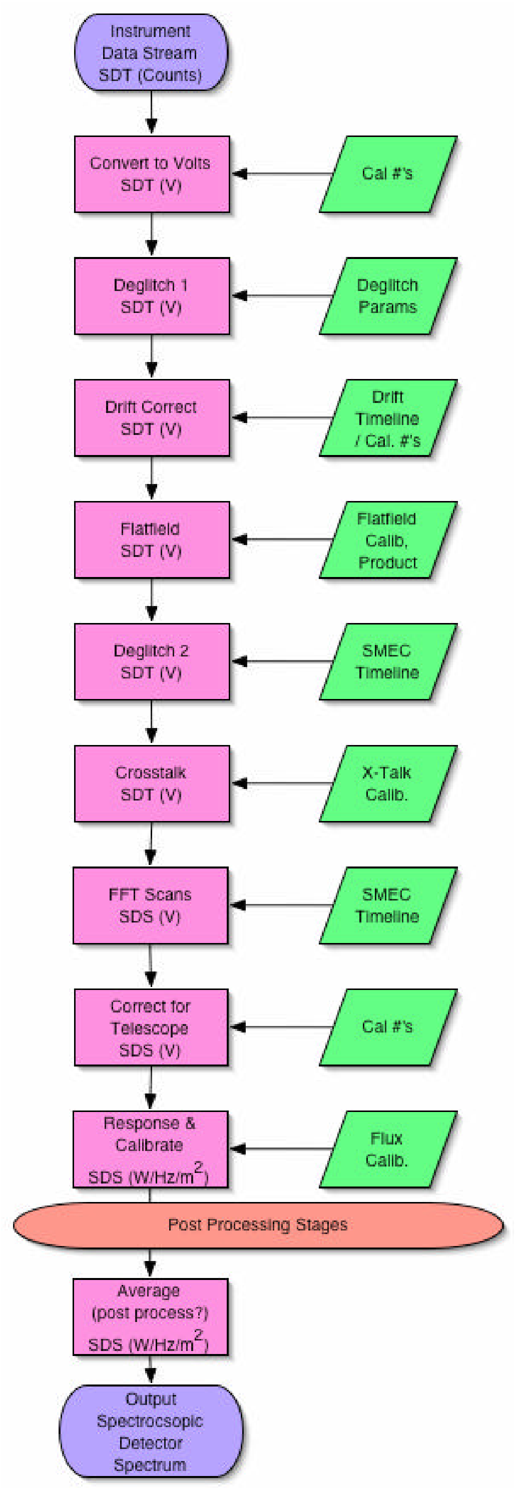
FTS Specific Products

Spec Detector Spectrum (SDS)
Spec Detector Timeline (SDT)

Notes

Calibration type products (green) should all be variants of a generic calibration product type.

Post-processing stage to be implemented but not inserted as part of standard pipeline.



Defining Products

There was a discussion as to how products should be defined, and whether to use KK's description as a baseline. SG stated that the description would not necessarily be complicated, for example instead of listing all parameters needed, one could just state "use all Housekeeping parameters". KK suggested one could create a general template for callibration files, which could be used as a basis for specific ones. There was also a discussion as to how flexible a product could be in terms of eg, altering the no of arrays as it passes through the pipeline. It was concluded that it can be fairly flexible, although one should bear in mind that certain processing steps will rely on the product being in a particular state. It was decided that people could request product types for RAL to define. They will also implement Product classes and ship them to those creating Tasks.

AI: all developers to let SG know which core products are needed, in addition to the ones identified below.

It was decided that the end of April was the deadline to have prototypes of the most urgently needed products in place, and May 15th was the deadline for them to be properly implemented. The following products were identified as urgent: PDT, Attitude, BSM, SMEC, SDT, SCU, DDT, some callibration products, SDS.

There was a discussion as to whether the prototypes should contain any data.

AI: MF to provide dummy data for Photometer; Lethbridge to provide dummy data for the Spectrometer.

DC asked whether BLAST data could be used for testing. TF confirmed that it could be used for testing, but not for any scientific purpose. DC added that it might not be needed for a few months.

AI: MF to find out about possible data from Bruce and John.

AI: TF to distribute the work he has already done on FITS headers, and provide a list of metadata to KK.

Mailing list

SS has created a mailing list for the OBST team.

Libraries

It was discussed as to whether we could recode the FTS C libraries. TF said that someone (to be appointed) will be seeing if it's viable to translate them into Java, and if it is, whether it is efficient to do so. MF asked to be updated once they have got going. SG said that it is possible to call C from Java, but the code is not portable, a C compiler is required, and it is not acceptable to ESA in the long term. SG also pointed out that translating from C to Java is more complicated than it initially looks.

IDL libraries

MF stated that it would be good to flag IDL libraries that are obviously needed, so that it can be investigated a) if they are in the HCSS and b) if not, whether they can be incorporated. KK stated that a request should include a priority of urgency. MF stated that the onus of coding in Java should be on the person who requests it. KK expressed some concerns with the transparency of functionality in the common IA. He will raise this at a management meeting soon.

AI: DC to go through the minutes and extract information useful for the SPIRE WIKI to keep our documentation up to date.

AI: HB / SG to produce a list of the functionality available in common IA (to the best of their knowledge).

It was suggested that the SCR system could be used for IDL to Java requests, as long as there are regular checks to make sure requests are being responded to.

DC stated that people should aim to produce a technical note for their packages in the next month.

IDL Routines produced during testing

There was a discussion as to what to do about IDL routines that are produced during testing. It was agreed that we should refuse to accept pure IDL code, but that we would accept detailed descriptions to convert to Java. TL warned that such routines might be instrument specific and therefore not that useful anyway. She stated that a list of requirements not currently in the common IA system is on the Scott1 website.

SURF routines

It was decided that SURF routines are not relevant to the pipeline at the moment, rather they will be needed at the end of the pipeline. SURF needs to be able to read in our final data products.

Quality Control

There was a short discussion of quality control. DC stated that it was the responsibility of the developers to keep track of the quality of the data in some shape or form. It should be included in the description of the Task, and the technical note should describe how you intend to keep track of it. DC is working on a quality control product. It is not yet decided how the quality control pipeline will work. SG warned against unnecessary abortions of the processing, as this can cause difficulties in analysing what went wrong. DC assured that the presence of an error would not inevitably lead to abortion of processing.

Combing products

TF enquired about combining products - what happens to the history? KK stated that the histories would be appended to each other, despite the size issues, because this is necessary for rerunning the process. MF pointed out that you can set the verbosity of the history, and SG explained that there may be 2 types of history: one would be human readable, and the other would be for rerunning the process. It was agreed that there should be a standard method in Task to write information to the history. DC stated that we need a checklist of everything that should be included in a complete Task.

Data Flagging / Masking

A sophisticated masking / flagging the data was recommended by SG, DC, ++, to be used to determine the processability and quality of the data at various stages (bit masking). Exactly how a process treats the mask should be specific to the processing involved. Outstanding issues are:

- How many bits required? (32/64/??)
- Should there be specific instrument bits and processing bits.
- Naming or any other conventions.
- Should there be specific mask reading (and writing) classes.

Flags need to be identified and requested by processing module coders. KK referred to an initial technical note of all reasonable flags. It was also emphasised that it should be documented within processing modules, exactly how each module treats, creates, and/or ignores flags. In cases where data could possibly be overwritten such as deglitching, it was decided that the original data should probably be preserved and included alongside any modifications made. The mechanism for including this data was not discussed in detail.

Coding Management, CVS, SPR and SCR system, Documentation

SG gave a presentation on CVS, coding management, the SPR and SCR system, and Documentation. The viewgraphs contain the details. There was a discussion about where documentation should be stored. KK was keen for Livelink to be used, as it has become more user-friendly recently, and it is important to keep a complete record of documentation in one place. Judy can upload it and provide URLs which can then be put on the WIKI.

The meeting ended with participants playing with some sample Task code.

APPENDIX: SPIRE Data Products (KK)

Timeline Product:

product (type=”***********”, description=”*********”)
   metadata:
      string   creator (description=”Creator”, quantity=””)
      date   creationDate (description=”Creation Date”, quantity=”**UTC**”)
      string   instrument=”SPIRE” (description=”Instrument”, quantity=””)
      string   modelName=”CQM” (description=”Instrument Model Name”, quantity=””)

```
        date        startDate (description=”??????????”, quantity=”UTC”)
        date        endDate (description=”??????????”, quantity=”UTC”)
        long        obsid (description=”Observation Identifier”, quantity=””)
        long        bbid (description=”Building Block Identifier”, quantity=””)
        long        bbtype (description=”Building Block Type”, quantity=””)
    composite dataset(description=”Detector Timeline”)
        metadata:
            string subsystem=  “PHOTOMETER” (description=”Instrument Subsystem”,
quantity=””)
        composite dataset (description=”Array Timeline”)
            metadata:
                string   arrayName=”PLW” (description=”Detector Array Name”,
quantity=””)
                string   modelName=”CQM “ (description=”Model Name”, quantity=””)

            table dataset(description=”Pixel Timeline”)
                metadata:
                    integer pixelId=”035” (description=”Pixel Identifier”, quantity=””)
                    string   pixelName=”C5” (description=” Pixel Name”, quantity=””)
                    string   source=”PHOTLW035” (description=”Source packet location”,
quantity=””)
                double-1d   onboardTime (description=”On Board Time”,
quantity=”Seconds”)
                float-1d      signal (description=”Detector Signal”, quantity=”Volts”)
                float-1d      error (description=”Error on signal”, quantity=”Volts”)
                integer-1d   mask (description=”Data Mask”, quantity=””)
            table dataset(description=”Pixel Timeline”)
            …
            …
        composite dataset(description=”Array Timeline”)
        …
        …
    history:
```

Offset Product:

```
product (type=”Offset”, description=”Detector Offsets Table”)
   metadata:
      string      creator (description=”Creator”, quantity=””)
      date        creationDate (description=”Creation Date”, quantity=”UTC”)
      string      instrument=”SPIRE” (description=”Instrument”, quantity=””)
      string      modelName=”CQM”  (description=”Instrument Model Name”,
quantity=””)
      date        startDate (description=”??????????”, quantity=”UTC”)
      date        endDate (description=”??????????”, quantity=”UTC”)
   composite dataset(description=”Detector Offsets”)
      metadata:
      table dataset(description=”Photometer Offsets”)
         metadata:
         double-1d      onboardTime (description=”Time of Offset”,
quantity=”Seconds”)
         integer-1d     offset000 (description=”Offsets for pixel 000”, quantity=””)
         integer-1d     offset001 (description=”Offsets for pixel 001”, quantity=””)
         …
         integer-1d     offset287 (description=”Offsets for pixel 287”, quantity=””)
      table dataset(description=”Spectrometer Offsets”)
         metadata:
         double-1d      onboardTime (description=”Time of Offset”, quantity=”
Seconds”)
         integer-1d     offset000 (description=”Offsets for pixel 000”, quantity=””)
         integer-1d     offset001 (description=”Offsets for pixel 001”, quantity=””)
         …
         integer-1d     offset071 (description=”Offsets for pixel 071”, quantity=””)
```

Actions Summary

Actioned Person	Deadline	Details
ALL Developers	End April	Send details of products not listed in these minutes to S. Guest
Steve Guest	End April	Produce prototypes for the most urgently needed products, then properly implement by 15 May
Matt Fox	5 May	Provide dummy data for photometer and to find out about possible real data from Bruce and John
U. Leth	5 May	Provide dummy data for FTS
Tanya Lim	5 May	Distribute work already done on FITS headers and

		provide list of metadata to K. King
Dave Clements	5 May	Go through minutes and extract information useful for info dissemination system
Helen Bright/ Steve Guest	End April	Provide list of functionality available in Common IA (to the best of their knowledge)
ALL Developers	15 May	Produce technical note for their packages including description of: • what it will do • how it will do it, • what products are needed as inputs • what products are produced as outputs • what quality control measurements can be produced • some consideration of flag handling for bad or dubious data
?? (Ken King, Steve Guest?)	5 May	Produce a checklist for all things that should be included in a complete Task