SPIRE Instrument Control Center: Use-Case Definitions

Version 2.1
July 9th, 2002
Compiled by M. Sauvage

Document Change Sheet

Issue	Date	Changes
1.0	15/03/02	First full Issue
2.1	09/07/05	
2.2	12/01/05	**Update by Tanya** **Usecases updated:** **UC CON 101 – issue 1.1 to issue 1.2** **UC CON 102 – issue 1.0 to issue 1.1** **UC HSC 101 – issue 1.1 to issue 1.2** **UC ICC 104 – issue 1.0 to issue 1.1** **UC ICC 110 – issue 1.0 to issue 1.1** **UC HSC 002 – issue 1.0 to issue 1.1** **UC SDR 001 – issue 1.0 to issue 1.1** **UC-CAL 001 – issue 1.1 to issue 1.2** **UC-CAL 004 - draft 0.2 to issue 1.0** **UC-ENG 004 – draft 0.2 to issue 1.0** **UC-CAL 101 – issue 1.2 to issue 1.3** <u>Dave Clemments Comments</u> **UC CON 101** Preconditions, 4 sentence, spurious capital in Are Some aspects of this use-case are now handled by the SPIRE bulletin board system at DAPSAS London. Adoption of commercial or open-source 'knowledge base' software is a possibility for this system. We do not yet have a formalised 'expertise database', but it is beginning to become clear where some centres of expertise lie. **UC CON 102** Information dissemination - is the usecase to call for distributing information outside the ICC, so this should be called if necessary. I think the open issues here could be converted into comments, with the possible exception of the ICC-HD or a broader group should assess the importance of the information. **UC HSC 101** Description: add Jython and Python to the list of languages; add 'systems' after 'software' since its not just software we might need training on. The comments just seem to be a list of stuff. Something should be added to explain the things listed. **UC ICC 104** The actor DCC isn't identified in the user case. It might be some confusion between DOC and CC? **UC ICC 110** Multiple we sites are still being used but we are working towards a portal system unifying the sites.

		The SPIRE BBS is functioning as a good way of distributing information within the ICC. No use case for issuing usernames and passwords still? **UC AOP 101** Still blank - is HCSS UCF 484 still the matching HCSS use case, and does it cover everything we want? **UC HSC 002** Brief description line 3: 'from through' - choose one of these words **UC SDR** Comments: para 2: 'a given processing *step*' - add step each observations -> each observation Indenting seems very inconsistent for some of the processing step comments. Phase correct: they may be -> there may be Some general formatting cleaning would be good (eg. extraneous full stop at end of DAC-101 summary). Is there anything more to say about special engineering modes and memory effects, both of which are highlighted at the end of this usecase as possible problem areas? <u>Tanya's Comments</u> **UC-Cal 001** Removed open issue on CT actor as this has now been resolved. **UC-Cal 004** Changed opening description to reflect current calibration documentation plan. Removed first comment. **UC-Eng 004** Updated open issues section. **UC-Cal 101** New open issues section to reflect current status.

Contents

Introduction

What are and how to read use-cases?

Writing use cases is an approach to system definition that, although it requires a change of perspective from the requirements-to-design approach, makes much more sense in the end, and allows for much more flexibility. To easily understand it, simply remember that it is based on the following question: How are we going to use this system we are trying to design, and for what purpose? Asking this instead of the more classical “What must the system do?” frees the imagination of the people designing the system and in the end allows for a much more thorough investigation of all the areas of the system to be designed.

If we have reached our goals, the complete list of use-cases should completely map all the possible ways to use the ICC system (system here refers to both the actual system, i.e. software and hardware, and the people in the ICC). Therefore reading the title and the brief description of the use-cases should give a rapid idea of the scope and completeness of the document.

There is a further subtlety to the use-case approach: its distinction of various levels of use-cases. They reflect that fact that when you are interacting with a system, you are sometimes using some of its fundamental properties, and sometimes only some marginal but necessary properties. For instance, let’s say the system we want to design is one that would be able to turn electronic object into physical ones (i.e. a printer). One of the fundamental use-cases of this system will be the “make physical object” (i.e. print) one. Lower-level ones would be those dealing with supplying to the system the substance it needs to create physical objects (ink, paper) or connecting it to the electronic object (network). The number of levels to use is the choice of the designers. For the design of the SPIRE ICC, we have decided to use three levels: summary, user and sub-function. The scope of each level is defined in the glossary part of this document. In this document we present first the summary-level use-cases, then the user-level ones, then the sub-function ones.

For further enlightenment on the use-case methodology, we strongly recommend "Writing effective use-cases", by Alistair Cockburn, which is available on the Liveling repository.

Our labelling scheme for use-cases

All the SPIRE ICC use-cases are labeled with a three-letter three-digit code. The three letters refer to a section of the ICC, i.e. a type of functions that the ICC will perform. They are:

AIV	Assembly, Integration and Validation
AOP	Astronomical Observation Preparation
DAC	Data Analysis – Common Parts
DAS	Data Analysis – System Parts
CAL	Calibration

CON	Relations with the Consortium
CUS	Common Uplink System
ENG	Instrument Engineering
FTS	Analysis of Fourier Transform Spectrometer Data
HSC	Relations with the Herschel Science Centre
ICC	Instrument Control Centre as a whole
KNO	Knowledge base
OBS	On-Board Software
OTH	Relations with the other ICCs
PHT	Analysis of the Photometer Data
SDR	SPIRE data reduction

As for the numbers, they allow first to distinguish between the summary-level use-cases, whose first digit is always 0, and the user-level use-cases, whose first digit is non-zero. The second 2 digits of the user-level use-cases try to connect the user-level to the summary-level. This is not always possible so user-level numbers are most of the time arbitrary.

Note that you will find references to other use-cases, called UCF-###. These are use-cases from the Herschel Common Science System. They are compiled in the document First Common Science System: Use-Case Definitions (ref: FIRST/FSC/DOC/0158).

We have added in an appendix use-cases related to Instrument Level Tests and QLA. There are two reasons for this: (1) this group of use-cases form a self-contained group and (2) they imply strong interaction with the HSC, and in fact the majority of these use-cases are HSC ones, only slightly modified to suit the ICC needs.

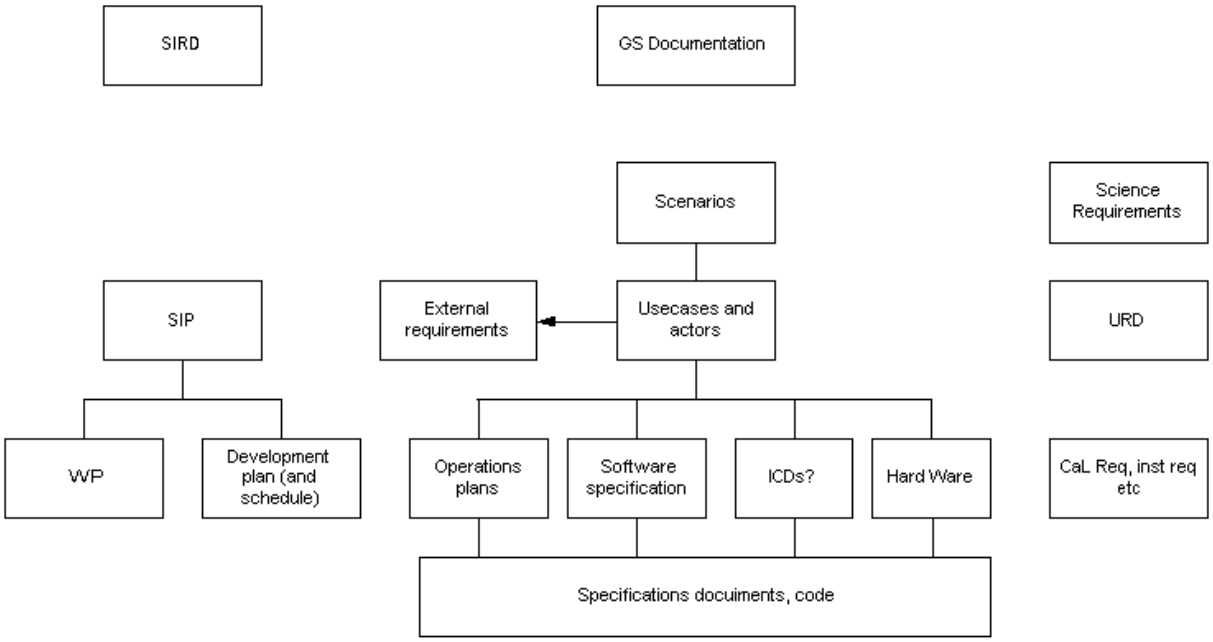
Finally a use-case document is not complete without its attached scenarios and actors list documents. These can be found under the following references in the SPIRE documentation repository: SPIRE-ICS-DOC001295 for the actors description, and SPIRE-RAL-DOC-001195

Reference Documents

SPIRE-ICS-DOC-001295	SPIRE ICC: Actor descriptions
SPIRE-RAL-DOC-001195	SPIRE ICC: Scenarios
SPIRE-RAL-PRJ-000018	SPIRE Science Implementation Plan
FIRST-FSC-DOC-0158	HCSS: Use-case definitions
FIRST-FSC-DOC-0157	HCSS: Actor descriptions

Spire ICC documentation tree

The insertion of the SPIRE ICC documents into the broader scope of the Herschel documentation tree is shown on this graph:


SIRD
GS Documentation
Scenarios
Science Requirements
SIP
External requirements
Usecases and actors
URD
WP
Development plan (and schedule)
Operations plans
Software specification
ICDs?
Hard Ware
CaL Req, inst req etc
Specifications docuiments, code

Glossary

This glossary compiles the definition of terms that the ICC Definition Team uses to refer to a number of concepts. It is ordered in no particular way at the present time. Please be aware that some of the terms defined here may have another meaning in a different context (HSC for instance), although we try our best to avoid that.

Terminology for the different ways of commanding the instrument:

- Mnemonics: these are defined in the SCOS2000 database, these are the basic commands we can send. They are contained in the MIB. (unlikely to change). They can have parameters. We (ICC) will never use directly these.
- Command sequence: a series of mnemonics build in SCOS2000 and also stored in the MIB. A sort of macro of mnemonics. Here again a command sequence can have parameters. We (ICC) will never use directly these.
- Script: available in EGSE/ILT as part of the Test Control. These scripts can combine command sequences and mnemonics, plus some control loops, if/then…Test Control is able to request from the HCSS that an observation request is converted into an observation execution.
- Observation execution: this is an instantiation of an observation. It has the form of a collection of mnemonics with their parameters and with a list of relative times between them.
- Observation request: resides in the HCSS and is the equivalent to an AOT (AOTs are in fact a subset of the observation request). The CUS is the environment in which these requests (which are scripts) can be defined.

Helpdesk:

Our helpdesk is officially responding to the HSC helpdesk, but there should be a possibility to access it directly from outside.

ICC Test environment:

A software environment common to all ICC members, where changes to the versions of the system are agreed and monitored. Inside the test environment, it is impossible to modify operational systems (e.g. databases).

ICC Sandbox:

A software environment completely isolated from the ICC system in the sense that elements of the ICC system can get in, but not out. The sandbox is a personal environment. It can contain any piece of software, including or exclusively the test environment. This is where developments will occur. We are not using the term development environment because it is used in the HCSS with a different meaning.

To validate:

By this we mean that the new object is safe for the system it operates in or with.

To verify:

By this we mean that the new object does what we want it to do.

Scientific validation

This means going through the process of verifying a system (typically the IA) with respect to scientific criteria negotiated between the ICC and the HSC.
Scientific validation of a calibration/IA update is a purely HSC concept. When a calibration update is made, it is the HSC that will require us to scientifically validate the new system.

Foreseen software development processes.

For all the software systems that the ICC will develop, including the attached data, one need a process that will both guarantee that ICC members can always operate with the latest version of the system, while maintaining a necessary stability of the system for non-expert users (including in fact most of the consortium).

Following up on the development of CAM IA, I can propose the following set-up. A number of development sites are identified (the ICC and possibly some contributing institutes). It is only at the development sites that changes to the system are performed and tested. Once an element is changed, verified and validated, the author of the change places it into CC. This author does not have the possibility to issue a new release of the system. New releases of any system are decided by a configuration control board (one for the developers' release and a larger one for the users' release). This ensures that the development site system is not chaotically evolving. Releases to the development site can be rather frequent (typically a month but not less).

Developers' releases:

To perform a release of the development site system (this is the test version), at a given date, CC is frozen (i.e. no more changes, except bug fixes, are allowed, elements under development have to be placed back in CC, even if incomplete) and the current version of all subsystems is released and tested (for a given period, e.g. 1 week). If these tests are good, then this is the new development site release. If not, we stay with the previous release and work out the bugs. On demand, the developers' release can be made available to consortium members. Current baseline for developers' release period is 4 weeks.

Users' releases:

Official or users' releases are much less frequent (6 months is the smallest period acceptable for software changes). Their release process is no different from that of the development site except that the CC board is different. Outside users may be frustrated by this slow process. But this minimizes the number of different versions of the system we are sending out in nature, and thus the problems we will have to handle.

For safety reasons, the users' release has to be identical to a developers' release.

Development site

As of today, the foreseen development sites are all within the ICC. However we are open to having more development sites as this holds the potential for faster and better developments of the systems. The privilege of a development site is to always be able to access the latest version of the system. The duty of a development site is to (1) regularly contribute to the development by providing elements of the system, (2) follow the CC convention, (3) install all developers' releases, and (4) participate in the different test stages of a release.

Data product:

A data product is not just anything. An error message or a report is not a data product.

The MIB database

The MIB (Mission Information dataBase) contains among other items, definitions of all the telemetry and telecommand packets, their parameters and their valid ranges. It is used to create telecommand packets from command mnemonics and their arguments, and to extract telemetry parameters from the incoming telemetry.

The CUS database

The CUS database contains all the information required by the CUS in order to create relative time-tagged command mnemonics from user input (time-ordered sequence of mnemonics, see that term definition above).

A "data" database

There will also exist another database, holding SPIRE data (telemetry content, formatted data, reduced data). During operations, this database will be the responsibility of ESA, and this is where all observers (including the SPIRE consortium) can retrieve data. Before that, the SPIRE consortium will maintain a database and use it to store all test and ground calibration data. During operations, the SPIRE consortium will have the option to import in this database all the data that gets stored in the HSC database. It will subsume other databases such as the MIB and the CUS databases.

Configuration control and review boards

We are assuming that the ICC will have its own internal configuration control and problem review boards. In some occasions, these boards will merge in the HSC configuration control and problem review board.

Data-frames

A data-frame is the lowest level of the data that we will do processing on. It is an unpacked telemetry source packet tagged with some additional data (including time stamps, observation identification, possibly pointing information). Data-frames will be

stored in the HCSS database. They represent the lowest starting point for IA. To completely process one observation, one will need to assemble many data-frames.

Simulators

The issue of instrument simulator is complex and divided into two aspects:

Engineering simulator

An instrument simulator that is mostly used to test the safety of operational procedures, the completeness of the house-keeping information, that the various system understand the telemetry produced by the instrument. In essence, this is the DRCU (Detector Readout and Control Unit) which contains the MCU (Mechanical control unit), DCU (Detector control unit) and the SCU (Subsystems control unit).

Science simulator

In that case, the emphasis is placed on the science data that comes out of the simulator. This data should contain all sources of noise and artifacts introduced by the instrument, so that the efficiency of observing modes and data reduction algorithms can be tested.

Use-case levels

Our use-cases are grouped into three different levels. We have tried here to summarize the rules we applied to attribute a level to a given use-case. Note that these rules are more guidelines than strict rules given that in some cases the scope of the use-cases may be slightly smaller than the ICC as a whole. Starting from the bottom we find:

Sub-function

A use-case describing a single action with an almost immediate result. Examples of these (not particularly drawn from our use-cases) are: "fit a curve", "display data", "access data storage".

User-level

A user-level use-case is one which satisfies an immediate goal of the primary actor. Such a use-case does not make sense on its own, but is part of a larger "mission" or "scenario". For instance, "reduce data" or "determine calibration value" would be user-level use-cases.

Summary-level

A summary-level use-case describes a top-level scenario of ICC use. "Calibrate SPIRE" is typically a summary-level use-case.

Summary-level Use-cases

UC-CAL001: Calibrate SPIRE

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.2**
Status: **issue**
Date: **12 Jan 05**

Brief description:
This use-case is the top-level use case to describe the actions taken when calibrating SPIRE.

Phase:
All phases.

Actors:
CS: Calibration scientist
IT: Instrument Tester
HSC: Herschel Science Centre
IH: Information Handler
IM: ICC Manager

Triggers:
The SPIRE performance needs to be measured.
Periodic, dependent on mission phase.
The result of a problem investigation requires that the calibration is updated.

Preconditions:
An instrument model (e.g. CQM, FM, FS) requiring calibration is available.
The HCSS is available and able to store data.
A documentation system exists.

Minimal post-conditions:
No observations are made which damages the instrument.
Previous calibration artefacts are retained and uncorrupted.

Success post-conditions:
The SPIRE calibration meets requirements.

Stakeholders and interests:
CS, TS, IE, HSC

Main Success Scenario:
1. CS: Produce or update the calibration plan (UC-CAL101).
2. IM: Approve plan and distribute associated work packages
3. IH: Circulate plan to all interested parties
4. CS: Schedule the observations. (UC-CAL102)
5. IT/HSC: Perform the calibration observations

6. CS: Check observations have been completed successfully.
7. CS: Retrieve observations and associated data from HCSS data base
8. CS: Check quality
9. CS: Data reduction of relevant observations for this time period
10. CS: Update relevant calibration artifacts (UC-CAL105)
11. CS: Determine which updates will be made persistent this period
12. CS: Verify the planned updates for this time period
13. CS: Approve all updates and enter into CC
14. CC: Updates HSC system with calibration artefact.
15. CS: Scientifically validates the updated system (UC-CAL103)
16. CS: Update calibration report (UC-CAL104)
17. IH: Distribute updated report to interested parties

Extensions:

2a: Plan is not approved
2a1. CS: go to step 1

4a: Unable to schedule observations.
4a1. CS: go to step 1

4b: Scheduling system doesn't work.
4b1. CS: issue problem report

5a: Unable to perform calibration observations due to instrument problem.
5a1. IT/HSC: issue problem report

5b: Unable to perform calibration observations due to lack of contact with the instrument.
5b1. CS: reschedule observations (go to step 4)

5c: Unable to perform calibration observations due to problem with the observation execution.
5c1. IT/HSC: issue problem report

7a: Unable to retrieve observations
7a1. CS: Notify Database manager of problem.

8a: Quality check fails
8a1. CS: issue problem report or document reason for failure.

9a: Unable to reduce data
9a1. CS: Issue problem report or software change request

12a Verification of planned update fails
12a1. CS: Add reasons for failure to calibration report
12a2. CS: Return to step 11

14a HSC does not accept update

14a1. CC: Place reasons for non-acceptance in the calibration report (UC CAL104)

14b: This use case takes place during ILT so we stop here

16a: The calibration updates relate to a problem report
16a1. CS: Feedback solution to problem reporting system
16a2. CS: Include references to problem report in delivery to CC

References:
UC-CAL101
UC-CAL102
UC-CAL103
UC-CAL104
UC-CAL105

Open Issues:
Assuming the LWS model, calibration updates are likely to be approved via regular group meetings, with the PI present, rather than by any individual asserting authority.

This group would oversee the development and monitoring of calibration activities including responsibility for the calibration plan and the calibration report. The exact structure of these documents, particularly in operations is still TBD.

Step 8 It is not yet clear how the Herschel quality checking procedures will be implemented. SPIRE ICC is required to provide automatic processing capability that will allow the HSC to inspect observations. Depending on criteria, presumably supplied by the ICC, failed or doubtful observations will be forwarded to the ICC for further investigation.

It is unlikely that the HSC will have the capability to quality check calibration observations as these may implement unusual instrument modes. Therefore we should expect to have ICC procedure to do this on a routine basis.

Comments:
There will actually be several documents giving various levels of calibration plans for SPIRE. At the top level there is a calibration requirements document detailing how the calibration will enable SPIRE to meet the science requirements. This then feeds into a calibration plan listing all aspects of SPIRE requiring calibration information

For ILT, there will be a calibration plan for ILT which will consist of a top level list of tests, a cross-reference with calibration requirements, a cross-reference with instrument requirements, a top level description of test procedures, then a detailed description of test procedures. Finally these procedures will be scheduled over the test campaign and it will be this schedule which is referred to in this use case.

During PV phase there will have to be a similar scheme, with a detailed plan on how the in-orbit calibration will be established down the level of planned observations which will be in place before launch.

During operations, we will have both long and short term plans, which are likely to be more fluid as understanding of the instrument will evolve and we may wish to schedule observations in the short term to investigate a particular aspect of the calibration.

Step 3: Here the interested parties are expected to be the ICC, the PST, the PI and the IS plus any members of the consortium actively working on calibration issues.

Step 4-5: In ILT, the observations will consist of smaller units than operations and will be scheduled via Test Control. 'Observations' can be measurements performed in the lab or astronomical observations.

Step 7: Here the associated data may be similar observations i.e. to build up a set of calibration knowledge, to improve S/N on an object or to check for calibration changes with time. The associated data may also be from test equipment, from uplink, from other instruments or from the spacecraft.

Step 9: The data reduction may not necessarily lead to the update of a calibration artifact therefore this is a separate step.

Steps 14-17 describe the delivery to the HSC but there is no real delivery just the activation of certain calibration artefacts.

Related Work Packages:

- Define a calibration documentation system e.g. plans requirements documents, what contents etc.
- Define calibration requirements (includes both ground and in-orbit requirements)
- Define calibration tables required (including format)
- Define calibration plan(s)
- Define and create calibration database (uplink and downlink)
- Define calibration analysis procedures
- Define and create calibration data processing and analysis software.
- Create a reporting system including problem reporting.
- Perform ground-based observations.
- Perform space-based observations.
- Perform laboratory measurements.
- Perform archival research.
- Perform Herschel observations.
- Perform theoretical modeling
- Perform observation scheduling
- Analyse calibration observations and populate calibration database
- Maintain calibration (including software, database and plan)
- Perform MIB updates

- Perform CUS updates
- Build Configuration controller
- Build Configuration control system
- Build time estimator
- Define quality control procedure.
- Training HSC staff.
- Create/maintain sandbox and test environment
- Negotiate scientific validation criteria
- Perform scientific validation of updates for the HSC

UC-CAL004: Prepare the calibration plan

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **12 Jan 2005**

Brief description:
In this use-case, we outline how the calibration plan (see definition in the comments section) is created. In this use-case, as well as in all the calibration use-cases, calibration is meant in a rather extended sense: it also covers measurements that are usually labelled “characterization”. We assume that the ICC has the leading responsibility in designing, performing, and analyzing the observations that will go into the calibration plan.
For the characterization work, the ICC will share this responsibility with the instrument team and/or various subsystem groups. Listing the SPIRE items that have to be calibrated and/or characterized is however a joint endeavour between the ICC, the instrument team and the various subsystem groups.
Note this use-case refers to an ongoing plan specific to a particular mission phase rather than the overall calibration plan where the high level list of calibration items are captured.

Phase:
All, since characterization of the instrument properties will start at the first instrument test, and we will still be calibrating SPIRE long after Herschel has ceased operating.

Actors:
CS: Calibration scientist

Triggers:
SPIRE has to be calibrated

Preconditions:
The instrument performance requirements are known.
The different SPIRE instrument models are defined enough so that we know which characterization/calibration is possible for which model.
An interface between the ICC and the instrument team is defined.
An interface between the ICC and the subsystem groups is defined
The instrument team has produced a "calibration requirement document".

Minimal post-conditions:
A calibration plan has been produced and is now in configuration control.
The instrument team and the ICC approve all items in the calibration plan.

Success post-conditions:
The calibration plan is deemed complete by the instrument team.

All observations in the calibration plan have been budgeted and scheduling constraints checked (when applicable, it is unlikely that characterization tests on the ground will have strong scheduling constraints).

Stakeholders and interests:
Instrument team: wants the calibration requirements to be met.

Main Success Scenario:
1. CS: Obtain calibration requirement document from instrument team.
2. CS: Interact with instrument team to make sure each requirement is well understood (UC-CON101).
3. CS: Investigates consequence of possible instrumental artefacts on the calibration strategy (UC-CAL106, UC-ENG101).
4. CS: Collects result from test phases to consolidate them into calibration measurements (UC-AIV103, UC-ICC107).
5. CS: Create calibration observations (UC-AOP101).
6. CS: Budget the calibration observations (UC-AOP102).
7. CS: Compile all observations into the calibration plan (UC-ICC104).
8. CS: Check that the calibration observations can be scheduled (UC-CAL102).
9. CS: Validate and verify the calibration plan with the instrument team.
10. CS: Place the calibration plan into configuration control (UC-ICC105).

Extensions:
4a: Other instruments may place requests on SPIRE calibration plan
    4a1: CS: get requests from other instruments (UC-OTH001)
    4a2: CS: go back to main success scenario at step 5.

5a: Calibration cannot be implemented with the observing templates
    5a1: CS: specify how the observation should be done (UC-ICC104).
    5a2: IE: generate new observing sequence (UC-AIV102)

6a: The sum of calibration time exceeds SPIRE's allocation.
    6a1: CS: Interact with instrument team to descope calibration items.
    6a2: CS: go back to main success scenario at step 7.

    6a1a: Calibration items cannot be descoped.
        6a1a1: CS: Inform SPIRE PI to plea for more calibration time.

8a: Some calibration observations cannot be scheduled
    8a1: CS: Interact with instrument team to see how these can be modified.

    8a1a: Observations can be modified
        8a1a1: go back to main success scenario at step 8.

    8a1b: Observations cannot be modified
        8a1b1: CS: Inform SPIRE PI to plea for more schedule flexibility

9a: The plan is not validated by the instrument team.
    9a1: CS: Obtain reason for validation failure from the instrument team.

9a2: CS: confront with calibration requirement analysis.

9a2a: First analysis of calibration requirements was incorrect.
9a2a1: CS: go back to main success scenario at step 3.

9a2b: First analysis of calibration requirements was correct.
9a2b1: CS: ask instrument team to check calibration requirements.

References:
UC-AIV102: Generate, validate and verify scripts and observation requests
UC-AIV103: Access data storage
UC-AOP101: Plan an observation
UC-AOP102: Estimate observation time
UC-CAL106: Simulate science performance
UC-CAL102: Schedule a calibration observation
UC-CON101: Capture consortium expert knowledge
UC-ENG101: Simulate instrument behaviour
UC-ICC104: Create or update a document
UC-ICC105: place an item into configuration control
UC-ICC107: Retrieve artefact from the database
UC-OTH001: Collaborate with other ICCs or HSC

Open Issues:
I am a bit perplexed regarding how to represent someone or something which is clearly outside of the ICC scope. To derive the calibration measurements that have to go into the calibration plan, the calibration scientist (the main actor of this use-case) has to interact with the instrument team. How do we represent this team, which is outside the ICC. It cannot be an actor, it is more a system, but clearly not the system under design.

Comments:
This is clearly an interface use-case as it relies heavily on the existence of something called the instrument team, which is outside of the scope of the ICC.
There are a number of calibration issues, which may be more appropriately named characterization issues. This distinction is not implemented in that use-case, but in some case it may lead to the characterization work being done outside of the ICC (simply because the ICC will not be competent in particular areas). We however believe that the ICC should be involved in all aspects dealing with the characterization of SPIRE, if only to be aware of the work going on.

Related work packages:

UC-CON002: Evaluate ICC-external algorithm

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **8 August 2001**

Brief description:
This use case describes how an externally (i.e. outside of the ICC) developed data reduction algorithm is, firstly, evaluated for possible inclusion in a later release of the Interactive Analysis software. Integration is not part of this use-case. This is described in the HSC use-cases dealing with Software Change Requests (UCF-361 “Analyse an SCR, UCF 371 “Implement an SCR”, UCF-421 “Submit an SCR”).

Phase:
Pre- through Post-Operations

Actors:
ST: Software Tester
SSD: Scientific software developer
SPA: Scientific product analyst
CC: Configuration controller
EKS: Expert knowledge system
ICC-S: ICC scientist

Triggers:
The ICC becomes aware of the availability of a new algorithm (e.g. through ICC helpdesk, consortium meeting or literature browsing).

Preconditions:
A ICC sandbox is available.
A configuration control system is available.

Minimal post-conditions:
An evaluation report is produced.

Success post-conditions:
A software artefact conforming to software standards is created. An SCR is filed.

Stakeholders and interests:
The ICC scientist needs to have an optimal IA. The algorithm supplier wants to have his algorithm available in IA. The SSD does not want unnecessary workload.

Main Success Scenario:
1. ICC-S: obtains from algorithm supplier, the EKS, (1) the algorithm, (2) a description of the algorithm, and (3) a set of reasons explaining why the algorithm is superior to the current IA.

2. ICC-S: checks that a similar algorithm has not already been submitted.
3. ICC-S: Makes sure algorithm is relevant for IA.
4. ICC-S: assigns priority level to the algorithm (balancing benefits against costs).
5. ICC-S: Defines algorithm test plan.
6. ST: Evaluates the algorithm quantitatively and provides report.
7. ICC: approves algorithm.
8. SSD: Re-writes the algorithm according to software standards – (see UC-ICC102).
9. ICC-S: submit an SCR – UCF-421.

Extensions:
2a: Algorithm is rejected as already submitted.
2a1. ICC-S: Files rejection report.
2a2. IH: Forwards that report to all interested parties.

3a: Algorithm is rejected as irrelevant.
3a1. ICC-S: Files rejection report.
3a2. IH: Forwards that report to all interested parties.

4a: Algorithm is rejected as too low priority.
4a1. ICC-S: Files rejection report.
4a2. IH: Forwards that report to all interested parties.

7a: New algorithm is rejected on the basis of its quantitative results.
7a1. ICC-S: Files rejection report.
7a2. IH: Forward that report to all interested parties.

References:
UCF-361 “Analyse an SCR
UCF 371 “Implement an SCR”
UCF-421 “Submit an SCR”
UC-ICC102 "Create or update a software artefact (within the ICC)"

Open Issues:
Can we really use UC-ICC102 for our step 8? There is some sort of flaw in the HCSS use-cases regarding SCR: they assume the software exists already. If we include the coding of the software in the SCR normal processing, then step 8 disappears. The “Analyse SCR” use-case in the HCSS should include coding. It is unclear what exactly is covered by the handling of SCR in the HCSS, and what is the scope of an SCR.

Comments:
Step 6 of the MSS may or may not include re-coding of the algorithm.
The algorithm might not be in the language/format of the IA system.

Associated Work Packages:
- Information handling (includes ICC helpdesk)

- IA software development
- ICC Configuration Control system
- Science verification
- ICC Sandbox environment

UC-CUS001: Test, validate and verify observing modes

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **4 October 2001**

Brief description:
This use case describes how the instrument observing modes are tested and validated to ensure that any observation execution does not compromise the safety of the instrument and performs the basic functions required by the observing mode.

Phase:
ILT, IST, PV, Operations

Actors:
IT: Instrument Tester
ICC-S: ICC Scientist
IE: Instrument Engineer
OE: Operations Engineer
IH: Information Handler

Triggers:
A new observing mode is needed or an update needs to be made to a pre-existing observing mode.

Preconditions:
The new observing mode definition is identified.
An up-to-date sandbox environment is available.
Instrument performance tests have been completed successfully (they define the instrument parameters that are needed to design observing modes).

Minimal post-conditions:
A report is issued describing the fate of the observing mode definition.

Success post-conditions:
The observing mode is validated, verified and assessed, and is ready to be included in the CUS database (using UC-CUS102).

Stakeholders and interests:
Astronomer: needs observing mode that will achieve the science objectives of SPIRE
ICC-S: needs to make sure that the instrument is used within its safety margins.

Main Success Scenario:
1. IT: Translates the observing mode definition into a CUS script.
2. IT: Stores the script in the sandbox CUS database.
3. IT & ICC-S & OE & IE: Define a test plan for testing the new observing mode

4. IT: Generate, validate and verify the observation executions resulting from that test plan (UC-AIV102).
5. ICC-S & IT & IE: Review test results and assesses the release of the new observing mode.

Extensions:
1a: The definition cannot be translated into a CUS script
1a1. IT: produce a report describing the reason of the non-feasibility.
1a2. IH: forwards report to all interested parties.

References:
UC-AIV102 Generate, validate and verify scripts and observation requests.
UC-CUS102 Update the CUS database.
UCF-752 describes how an observing mode is defined and placed into the CUS.

Open Issues:

Comments:
This use-case does not extend to a full science verification of the observing mode.

Step 1&2 of the main success scenario are similar to step 1-5 of UCF-752 Define an observing mode.

At the end of step 5, the assessment may include recommendations to modify the original observing mode definition and restart the use-case.

After a positive assessment, UC-CUS102 can be invoked to include the new observing mode in the CUS database starting at step 4 of its main success scenario (it is actually the only way to include it in the CUS database).

Work-packages:
- Information handler
- Produce the CUS (this is a common systems work-package)
- Create the CUS database
- Provide QLA
- Provide IA (at least basic functionality as the use-case does not extend to a full science verification of the observing mode.
- Define AOT test plans
- Define the AOTs
- Training in using the CUS
- Provide sandbox environment
- Provide instrument test environment and instrument simulator
- Test the AOTs
- Access Spacecraft parameters (to know what the satellite is able to do)
- Perform instrument testing (to know what the instrument is able to do)

UC-ENG002: Investigate external SC/instrument effect on SPIRE instrument

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6th November 2001**

Brief description:
This use case describes the steps involved when a (potential or actual) problem is encountered with the instrument and it is suspected or known that the source of the problem is the spacecraft or another instrument. It covers the steps from identifying a problem, through the investigation of the problem, and, if necessary, its solution via the System Change Request (SCR) mechanism.

Phase:
ILT -> Operations

Actors:
PR: Problem reporter
PA: Problem analyst
MRB: Material Review Board
CC: Configuration controller
IM: ICC Manager

Triggers:
An abnormal behaviour that cannot be attributed solely to the functioning of the instrument. There may be a number of sources that identify such abnormal behaviour, e.g. routine calibration, status warnings from MOC.

Preconditions:
Information is available on the status of the spacecraft and other instruments during the period(s) when the effects are found to occur.

Minimal post-conditions:
The effect is characterized and its impact is assessed.

Success post-conditions:
The effect is found not to be a problem or a solution is found to the problem.

Stakeholders and interests:
HSC and Other ICCs: In order to remove a detrimental effect on the instrument it would be necessary to change the functionality or procedures of the spacecraft and/or other instruments. It is in their interests that such changes are not detrimental to them.

Consortium, Astronomers: It is in their interests to be notified if any effect may have implications on the scientific performance of the instrument.

Main Success Scenario:

1. PR: Reports the problem to the PA (mechanism TBD).
2. PA: Determines what supplementary information is needed for investigation and gathers the required information.
3. PA: Analyses the problem, produces a report and passes it to the MRB.
4. MRB: Determines if any updates are needed to the instrument logic or data processing.
5. CC: Issues an SCR on the appropriate system.

Extensions:

3a: If the analysis shows that no updates are needed, the appropriate documentation for SPIRE users should be produced and distributed.

4a: The problem report suggests that one (or both) of the other instrument(s) needs to change its operating parameters but analyses are needed from the other systems before an SCR can be made.

4a1: The MRB pass the report to the IM.
4a2: The IM contacts the relevant parties UC-OTH001
4a3: The IM passes recommendations from outside investigations to the MRB

References:

UC-OTH001

Open Issues:

The internal mechanism in the ICC for logging and analysing problems is not yet defined. TBD
What is the mechanism for getting S/C or other instrument data, will there be an issue with data rights?

Comments:

Work Packages:

- Create a problem reporting system / Software change request system
- Analyse problem report.

UC-ENG004: Perform instrument test

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **12 Jan 2005**

Brief description:
This top-level use-case describes how an instrument test is designed, performed and analyzed. By instrument test, we mean a ground-based test on one of the instrument models (including the flight model), or during commissioning of the instrument in-flight. This use-case however does not cover tests that could occur during PV or operations, as these will likely follow a different route (i.e. involve the HSC).
Note that this use case is similar in essence to the HCSS use-case UCF-701: instrument test during ILT.

Phase:
ILT, IST and commissioning

Actors:
IS: ICC Scientist
IT: Instrument Tester

Triggers:
The major trigger is the Herschel schedule that stipulates that the instrument development has to go through a number of test phases (IST, ILT).
Another trigger can be the investigation of an instrument problem (Note that after commissioning investigation of instrument problems will go through UC-ENG005).

Preconditions:
An instrument model is ready for testing.
A test facility is available.
A system to analyze the test data is available.
A data storage facility, to store the test data, is available.
An instrument commanding system is available.
Performance requirements expected from the instrument have been defined.

Minimal post-conditions:
The instrument model is unharmed. A report on the outcome of the test is produced.

Success post-conditions:
The test has been performed and analyzed successfully. Test procedure(s), test data and test report are stored in a legal copy of the HCSS database (or the database itself). All software created/updated for the test is documented and configuration controlled.

Stakeholders and interests:

The instrument team needs the instrument to be thoroughly tested, specifically against its performance requirements.

Main Success Scenario:

1. IS: Consult with experts to design the test objectives (UC-CON101)
2. IT: Write the test code (UC-AIV102)
3. IT: Execute the test on the instrument model with the test facility
4. IT: Retrieve the test result from the instrument database (UC-AIV103, UC-ICC107)
5. IS: Analyze the test data (UC-QLA001, UC-PHT101, UC-FTS101)
6. IS: Validate the results of the analysis (UC-CAL103)
7. IS: Document the test results (UC-ICC104)
8. IT: Store the test procedure, analysis and report (UC-ENG102)

Extensions:

3a: The test has to fit into a test schedule
    3a1: IT: schedule a test execution (UC-CAL102)

3b: The test does not execute
    3a1: IT: write a problem report and submit it

5a: The test requires to be compared with results from a simulation
    5a1: IS: simulate instrument behaviour (UC-ENG101)
    5a2: IS: simulate science performance (UC-CAL106)
    5a3: IS: go back to main step 6

5b: The test requires specific software to be developed
    5b1: IT: create software (UC-ICC004)
    5b2: IT: put the software in configuration control (UC-ICC105)

6a: The test results cannot be validated because of a problem
    6a1: IS: write a problem report and submit it

References:

UC-AIV102: Generate, validate and verify scripts and observation requests
UC-AIV103: Access data storage
UC-CAL106: Simulate science performance
UC-CAL102: Schedule a calibration observation
UC-CAL103: Scientifically validate a calibration/IA artefact
UC-CON101: Capture consortium expert knowledge
UC-ENG101: Simulate instrument behaviour
UC-ENG102: Store analysis data
UC-FTS101: Reduce spectrometer data
UC-ICC004: Create or update a software artefact
UC-ICC104: Create or update a document
UC-ICC105: place an item into configuration control
UC-ICC107: Retrieve artefact from the database
UC-PHT101: Reduce photometer data
UC-QLA001: Use QLA

UCF-701: Instrument test during ILT

Open Issues:
We are missing a use case for writing and submitting an ICC problem report which is not a software problem. We have a use-case to deal with such problems (UC-ICC002: Handle problem report) but this may not encompass all types of problem and we may have to utilise other problem reporting systems such as NCRs.
We are also missing a use case for the execution of the test. It is an open issue whether the test execution is considered to be an ICC task or an instrument team task.

Comments:
As mentioned above, there is a thin line that divides what the ICC is supposed to do with respect to instrument testing, and what the instrument team expects to be doing. This line may be artificial because a number of the ICC members also have instrument team hats. It is also blurred by the fact that ICC use-cases may also be "acted" by people outside the ICC.
We believe this use-case can cover commissioning as well because this will mostly be a re-run of some of the IST-ILT. For further phases, it is clear that any "test" will have to involve the HSC (if only to agree to it).
The HCSS use-case document also covers the test of instrument during ILT in its use-case document (UCF-701: instrument test during ILT). Even though they have been written independently, these two use-cases are very similar (which is a good thing). There will probably exist a transition in the SPIRE development process from this use-case to the HCSS one (and note that there are no extensions to the HCSS use-case so that it is probably meant to cover the “acceptance” tests at each instrument model delivery).

Related work packages:

UC-ENG005: Investigate instrument problem

Level: **summary**
Scope: **SPIRE ICC**
Version: **0.2**
Status: **draft**
Date: **5 July 2002**

Brief description:
A problem that involves the SPIRE instrument (e.g. revealed by unexpected signal in the data, or no signal at all, or error messages in the telemetry) has occurred and needs to be analyzed and solved. This use-case is restricted to the phases where the instrument is in flight. During these phases, the ICC has the principal responsibility over the instrument health and safety. During earlier test phases (i.e. ground-based tests), we can expect a strong involvement, or even leadership from the instrument team.

Phase:
Commissioning, Performance Verification and Routine

Actors:
PA: problem analyst
IT: instrument tester
IS: instrument scientist
MRB: Material review board

Triggers:
A problem has been noticed, in the data or telemetry, and preliminary analysis reveals that the instrument is likely to be responsible (for instance at step 4 of UC-ICC002).

Preconditions:
Instrument is in flight.
Simulators, both for the science and technical properties of the instrument are available and updated to reflect the actual flying instrument.
The Flight Spare is available for tests.
Instrument commanding software is available.

Minimal post-conditions:
The problem is characterized, its analysis is documented, and this document is placed in configuration control.

Success post-conditions:
The instrument problem is identified and solved. The analysis and solution are documented and placed in configuration control. Any changes to the system software have been tested, validated and verified. Possible changes of the OBS are submitted, but not performed (subject of another use-case, UC-OBS001).

Stakeholders and interests:
Problem reporter: needs to see the problem fixed or at least identified and documented.
SPIRE PI: need to have an operating instrument.

Main Success Scenario:
1. PA: Tries to reproduce the problem with the simulators (UC-CAL106, UC-ENG101)
2. PA: Checks for possible influence of other Herschel instrument (UC-ENG002).
3. PA: identifies work-around or solution to the problem.
4. PA: test work-around or solution to the problem on simulators (UC-CAL106, UC-ENG002).
5. PA: proposes changes to the SPIRE operations to implement work-around or solutions.
6. MRB: accepts proposed solution.
7. IS: implements solution.
8. CS: update calibration plan to reflect new operation mode of SPIRE (UC-CAL101).

Extensions:
1a: problem cannot be reproduced with the simulators
- 1a1: PA: design test do be done on instrument model
- 1a2: IT: performs test on instrument model (UC-ENG004)
- 1a3: PA: go back to main step 2.

1a2a: Problem cannot be reproduced on the instrument model
- 1a2a1: IT: reports failure to PA
- 1a2a2: PA: Questions previous analysis that lead to the diagnostic of an instrument problem

3a: PA does not find work around or solution (problem is not solved)
- 3a1: PA: reports to ICC manager.

4a: work-around or solution cannot be tested on simulators
- 4a1: PA: design test to be done on instrument model.
- 4a2: IT: performs test on instrument model (UC-ENG004)
- 4a3: PA: design test to be done on flight model
- 4a4: CS: plan and schedule the test (UC-AOP101, UC-CAL102)

5a: Changes involved are far-ranging (new observing mode, change to the OBS,…)
- 5a1: PA: Suggested changes, new operating modes or tests are documented in an official document (UC-AIV102, UC-CUS001, UC-ICC104)
- 5a2: IM: Set-up a review board including members of the consortium and of the HSC to examine the proposed changes
- 5a3: This review board accepts the changes.
- 5a4: PA: go back to step 7.

5a3a: The review board does not accept the changes
- 5a3a1: PA: go back to step 3 to propose a new solution

6a: MRB rejects the proposed solution.
6a1: PA: go back to step 3.

References:
UC-AIV102 Generate, validate and verify scripts and observation requests
UC-AIV101 Update OBS
UC-AOP101 Plan an observation
UC-CAL106 Simulate science performance
UC-CAL101 Update calibration plan
UC-CAL102 Schedule calibration observation
UC-CUS001 Test, validate and verify observing modes
UC-ENG101 Simulate instrument behaviour
UC-ENG002 Investigate external SC/instrument effect on SPIRE instrument
UC-ENG004 Perform instrument tests
UC-ICC002 Handle problem report
UC-ICC104 Create or update a document

Open Issues:
What is the level of this use-case? Obviously it is not a user-level one. It may not be a top-level summary one as it is located below the “handle problem report” use-case (UC-ICC002). So I would guess that UC-ICC002 is a top-level use-case.
Looking at UC-ICC002 “handle problem report” I can see that maybe the present use-case goes a little too far: its steps 6 and beyond are included in UC-ICC002.
At step 1a2a2 we clearly have a problem: we cannot reproduce the instrument behavior either with simulators or instrument model. What do we do then?

Comments:
The proposed solution can be far-reaching. It is clear that in case a change to the OBS is proposed, UC-AIV101 has to be used. In case a new operating mode has to be designed, UC-AIV102 and UC-CUS001 will come into play.

Related work packages:

UC-HSC002: Support HSC query

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **12 Jan 2005**

Brief description:
This use case describes the steps involved in responding to, and dealing with, a request for help from the HSC. A request originating from an astronomer will come via the HSC Helpdesk. A request from the HSC will come through various formal and informal channels. The support includes the formal open, tracking and closing of queries. The generic name for these channels is the ICC-helpdesk by analogy to the Helpdesk.

Phase:
ILT, IST, PV, Operations

Actors:
HSC: Herschel Science Centre
ICC-HD: ICC Helpdesk (an adaptation of the Helpdesk Actor)
PA: Problem Analyst
IM: ICC Manager

Triggers:
The ICC helpdesk receives a query from the HSC

Preconditions:
ICC – HSC interface must exist
An ICC helpdesk infrastructure is in place
An ICC documentation system exists

Minimal post-conditions:
The query is logged internally, information is provided to the HSC as to the fate of the query (i.e. under investigation/priority ranking/does not concern ICC…).

Success post-conditions:
The query and the subsequent investigation results are archived and is answered to the satisfaction of the HSC.

Stakeholders and interests:
ICC manager: the query is dealt with to the satisfaction of the HSC
ICC-HD: Information for HSC support is easily retrievable.
PST: has to deal with the general astronomer appropriately.

Main Success Scenario:
1. ICC-HD: Register and log query, identifying it as coming from HSC

2. ICC-HD: Makes sure the query has not been answered yet
3. ICC-HD: Determines who in the ICC is able to deal with the query
4. PA: Analyses the query and produces a report answering it
5. ICC-HD: logs the report and forwards it to HSC
6. ICC-HD: closes out the helpdesk query

Extensions:
2a: ICC-HD identifies the query as already answered
    2a1: ICC-HD notifies HSC of repeated query and points to archived answer
    2a2: ICC-HD closes out the helpdesk query

2b: ICC-HD identifies the query as too low priority under current ICC workload
    2b1: ICC-HD notifies HSC of this decision

3a: ICC-HD concludes the query cannot be dealt with in the ICC because the problem is larger
    3a1: ICC-HD: notifies ICC manager of the problem in dealing with the query
    3a2: IM: defines work-package with other ICC/HSC (UC-OTH001)

4a: PA discovers that the query is related to a problem in the system/instrument
    4a1: PA: Notifies ICC-HD of the problem
    4a2: ICC-HD: files a problem report

4b: PA discovers that the query is related to a problem with the spacecraft or the other instrument
    4b1. PA: Notifies ICC-HD of the problem
    4b2. ICC-HD: files a problem report on the HCSS

References:
UC-OTH001 – collaborate with other ICCs or HSC.

Open Issues:

Comments:
This use-case assumes that the Helpdesk is able to take such decisions as (1) who will be able to deal with the query, (2) the query is actually related to a problem larger than the SPIRE ICC. This capacity clearly puts constraints on the design of the Helpdesk. The visibility of the ICC structure to the ICC helpdesk is also of key importance for the ICC-HD to operate correctly.

It is also likely that the scenario that will mostly be used will be the one going through 4a (i.e. the query corresponds to an actual problem).

We expect that the HSC helpdesk will be trained enough in the functioning of SPIRE so that they can deal with part of the queries they receive from the general astronomer.

Associated Work Packages:

- Provision of ICC help-desk (includes staff)
- Provision of information database
- Problem handling system

UC-ICC001: Manage the ICC

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **11 March 2002**

Brief description:
This Use Case sketches the high-level responsibilities of the ICC manager.

Phase:
ILT, IST, PV, Operations

Actors:
IM: ICC manager

Triggers: N/A

Preconditions: N/A

Minimal post-conditions:

Success post-conditions:

Stakeholders and interests:

Main Success Scenario:
1. IM: Establish jointly with ESA the detailed list of ICC tasks and deliveries.
2. IM: Generate the ICC SIP.
3. IM: Establish and maintain the ICC schedule.
4. IM: Manage the ICC interfaces with the ESA Project Team, the other ICCs, the HSC and the MOC.
5. IM: Support the ground segment reviews.
6. IM: Attend the meetings of the F-GSAG.
7. IM: Establish jointly with SCI-SA the set of documents to be produced by the ICC.
8. IM: Provide the infrastructure and facilities to support the work of the ICC.

Extensions:

References:
SIRD-ICCF-005 to SIRD-ICCF-045
SIRD-ICCF-200
SIRD-ICCO-080
SIRD-ICCA-050

Open Issues:

Comments:

UC-ICC002: Handle problem report

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **19 June 02**

Brief description:
This use case follows the whole process attached with problem reporting and analyzing, from the time an ICC member files in a problem report, to the time when a solution has been found or implemented, or the problem has been declared a feature, and the report has been closed.
The ICC will use the HCSS problem report system so this use case only contain sections which are relevant/specific to the ICC.
As explained in the comments section, this use case also covers ICC assistance to HSC regarding Quality Control of data products.

Phase:
All phases including post-operations

Actors:
CC: Configuration controller
PA: Problem analyst
MRB: Material review board
IH: Information handler

Triggers:
A problem has been reported to the ICC area of the problem report system

Preconditions:
A system to record and track problem reports is in place
A clear distribution of responsibilities within the ICC is defined and available.

Minimal post-conditions:
A decision is taken regarding steps to be taken with respect to the reported problem (could be none), it is recorded in the problem report system, and the originator of the problem is notified.

Success post-conditions:
Problem is identified and solved (possibly involving modification of the system), the analysis and chosen solution are recorded in the problem report system as well as the identification of the system version in which the problem has been solved.

Stakeholders and interests:
CCB: Configuration Control Board needs to be able to track all developments and/or modifications of the system
ICC Manager: needs optimum performance from the ICC and the instrument.

Main Success Scenario:

1. CC: Validates that the problem report is a new one
2. CC: Identifies ICC member/group in charge of the system incriminated in the problem report
3. CC & PA: performs first analysis to establish priorities and deadlines
4. PA: analyzes the problem and identifies modification to be made
5. PA: Adds problem analysis and recommended solution to problem report
6. MRB: Recommends the modifications to be made
7. PA: modifies the elements of the system
8. CC: Enters the modified elements in configuration control
9. CC: Closes the problem report.
10. IH: distributes closed problem report to all interested parties

Extensions:

1a: Problem is not entered in the ICC Problem Report system
- 1a1. CC: Enters the problem report in the ICC PR system
- 1a2. CC: goes back to step 1 of MSS

2a: The problem is not a new one
- 2a1. CC: Identifies previously closed problem report
- 2a2. IH: Distribute this report to all interested parties

7a: The MRB does not recommend the modifications to be made
- 7a1. CC: labels the problem a feature and closes the problem report
- 7a2. IH: distribute the closed problem report to all interested parties.

References:

Open Issues:

There clearly is a potential for conflict in the extent of responsibilities between the ICC and the HSC over the systems being developed by the ICC.

Comments:

A problem report here also covers the issue of a change request (actually there may be no difference between the two).
Note that the scenario outlined here is rather the ideal one. A problem report can actually end in a Software change request (SCR).
It is currently foreseen that there will be one problem report system for all three ICCs and the HSC, inside which there will be a SPIRE ICC area.
The current MSS does not include possible iterations between the PA and the MRB if they do not agree on the solutions. I'm not sure it is necessary to show this process in the MSS
We are keeping this use-case in the ICC list as is, even though the problem report system is now covered at the HSC level. This is for two reasons (1) as a placeholder to remind us that this aspect is covered, and (2) because it defines our needs with respect to the implementation of the actual system.
Finally we mention here the issue of Quality Control. Formally the ICC only has to deliver to HSC the tools to perform QC. Actually performing QC is not part of the ICC

tasks. Therefore there are no use-case describing this activity. It is however clear that the ICC will have to do some QC, mostly on HSC request. The present use-case adequately describes the process the ICC will follow in these cases.

Related work packages:
Define/Create/Maintain problem report system (now an HCSS work package mostly)
Define/Create/Maintain information handling system.
Manage the ICC

UC-ICC003: Plan and deliver a new user release

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **9 August 2001**

Brief description:
This describes the course of actions followed in the ICC to plan and deliver a new user release of a system. A user release means a release of a version of the system that will eventually be distributed to users outside of the ICC. This is different from a development release, which is accessible mainly to the ICC as a development site. The actual release will be made through the HSC.

Phase:
Ground-segment test phase and onward.

Actors:
CC: ICC configuration controller
SSD: Scientific software developer
ICC Scientist: ICC scientist
IH: Information handler
HSC: Herschel Science Center

Triggers:
The CCB decides a new release date based on a decision taken in the consortium (formal procedure is TBD).

Preconditions:
A configuration control system is available.
Changes have been made to elements in CC.
A stable developers' release of the system is available (this stability notion is related to the phase of the mission).
A list of new features to include in the new release is available.

Minimal post-conditions:
It is possible to revert to the previous release.

Success post-conditions:
A new user release is delivered to the HSC.
Changes with respect to the previous version are documented.
Science verification has occurred.
HSC accepts the release.

Stakeholders and interests:
AST: wants to benefit from the latest improvements as soon as possible.
HSC and ICC want to guarantee the best relationship with Herschel observers.

ICC CCB wants to keep software development under control.

Main Success Scenario:

1. CC: issues time table for next release based on what is available for release and what is still needed for release.
2. SSD: create or update software artifact(s) within the ICC (UC-ICC101)
3. CC: plan and deliver a new developers' release (UC-ICC102)
4. ICC Scientist: test, validate and verify scientifically the release (UC-ICC101).
5. CC: documents the release.
6. IH: delivers release document to all interested parties.
7. CC: delivers the new release to HSC.

Extensions:

2a: SSD: software artefact is not created/updated.
    2a1. CC: goes back to step 1.

3a: CC: delivery of the new developers' release fails.
    3a1. CC: issues a report on the failed delivery.
    3a2. IH: delivers report to all interested parties.
    3a3. CC: goes back to step 1.

4a: ICC Scientist: test, validation or verification of the release fails.
    4a1. CC: issues a report on the failed test/validation/verification.
    4a2. IH: delivers report to all interested parties.
    4a3. CC: goes back to step 1.

7a: HSC: rejects new release.
    7a1. ICC Scientist: analyzes HSC report and take appropriate action (could include going back to step 1, going back to PI…)

References:

UC-ICC101: Create or update a new software artifact.
UC-ICC102: Plan and deliver a new developers' release.

Open Issues:

We do not yet know who instructs the CCB to make a new release and what it should include. This body should include the SPIRE consortium.
It is not completely clear what actor the ICC Scientist really is (it may be a new actor to add to the list).
Note that the HCSS does not have a use-case to describe what they do upon reception of a release (such as in step 5 of this use-case). UCF-005 "Analyze and plan a release" is the closest use-case to that we can find.

Comments:

The formal decision procedure to release includes a possibility for the CCB to make a new release if it is urgently needed.
The planning of the release may have to be negotiated with HSC.

Related work packages:

- Provide ICC Configuration Control system.
- A system for recording decisions/actions (by ICC, consortium…) is available.

UC-ICC009: Maintain computing environment

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **10th Jan 2002**

Brief description:
This use case describes how 1) the computer system manager maintains the computing environment of the ICC or DAPSAS (includes hardware, third party software, security, etc), and 2) the database manager maintains the local node of the HCSS (and possible a local non-distributed database).

Phase:
All phases

Actors:
CM: Computer system manager
DM: Database manager

Triggers:
None – this is a continuous process.

Preconditions:
A working computer exists.
An authenticated software product exists.

Minimal post-conditions:
The computing environment can be restored to previous functioning state.

Success post-conditions:
- System security is not compromised.
- The system is kept up-to-date.
- A recent backup is always available in case of problems.
- User data has not been deleted or corrupted

Stakeholders and interests:
CM,DM: responsible for system maintenance.
All users of the system.

Main Success Scenario:
1. CM: Upgrade OS and install patches as necessary.
2. CM: Update computing environment as necessary.
3. CM: Procure and update new hardware as necessary.
4. CM: Make regular backups.
5. DM: Maintain database

Extensions:

References:
UR-ICC3.3.5
UR-FSC2.3.1
UR-CUS3.2.1
SIRD-ICCF-175
SIRD-ICCO-080

Open Issues:

Comments:

UC-KNO001: Manage ICC Knowledge base

Level: **summary**
Scope: **SPIRE ICC**
Version: **0.2**
Status: **draft**
Date: **June 20th 2002**

Brief description:
This use case describes how the knowledge base of the SPIRE ICC is managed. The knowledge base consists of all the information concerned with SPIRE known and collected by members of the ICC, as well as other applicable specialist knowledge known or acquired by them. The aim of this use case is to ensure efficient information transmission, acquisition and maintenance inside and outside the ICC.

Phase:
All phases

Actors:
IM: ICC Manager
IH: Information Handler

Triggers:
Information regarding SPIRE, the ICC, submillimetre/FIR astronomy or the systems used by SPIRE/ICC has evolved or appeared. This needs to be reflected in the ICC knowledgebase.

Preconditions:
A system to store and access information exists

Minimal post-conditions:
No information is lost to the ICC knowledgebase

Success post-conditions:
The ICC knowledgebase is updated to incorporate the new development

Stakeholders and interests:
Expert Knowledge System – to ensure easy and effective access to the knowledge base

Configuration controller – to ensure configuration control on documents and items in the knowledge base

Documenter – to ensure that the system set up is effective and easy to use for document preparation

ICC Helpdesk – to maintain and have easy access to the knowledge base so they can answer questions and distribute relevant information

Information Handler – to ensure efficient and effective movement of data to and from the knowledgebase

Infrastructure Software Developer – to have easy access to information relevant to their development responsibilities, and to have an effective system for storing and managing documentation that they produce

Scientific Software Developer – to have easy access to information relevant to their development responsibilities, and to have an effective system for storing and managing documentation that they produce

Main Success Scenario:

1. IM evaluates the scope of the new or evolving information
2. IM determines how best this information is to be stored in the knowledgebase
3. IM takes stores the information in the knowledgebase
4. IH notifies all interested parties of the knowledgebase change

Extensions:

3a: IM determines that staff training is the best way of storing the new information
3a1: Trigger UC-HSC101 (Training)

References:

UC-CON101 Expert Knowledge Capture
UC-CON102 Disseminate Knowledge
UC-HSC101 Train Personnel
UC-ICC010 Maintain ICC Web Page
UC-ICC104 Create or Update a document
UC-KNO002 Organise Meeting(s)
UC-KNO003 Attend Meeting(s)
UCF-091 Search PR Material
UCF-123 Provide User Support
UCF-061 Scientifically Evaluate Proposals

Open Issues:

The existing usecases UC-CON101, UC-CON102, UC-HSC101, UC-ICC010 will need to be reclassified as User Level usecases beneath this summary level usecase, and may need to be renumbered to fit into the KNO hierarchy. UC-ICC104 may need to be renumbered to come into the KNO hierarchy.

Other referenced usecase could also be incorporated into the extensions to specify calls to Create/Update a document, expert knowledge capture etc. Is this necessary?

Comments:

The goal of this use case is to establish a system to maintain the expertise of the ICC and to preserve and distribute the knowledge gained by ICC members.

The trigger can come about through both external and internal developments. Thus, for example, the changing state of an ICC project in development (such as these usecases) can itself be the trigger for a meeting or documentation. The changing state of a software system used by ICC (eg. an upgraded Java compiler), or the changing state of the ICC knowledgebase brought about by a new team member (who might not know Java) could both be triggers for staff training in the Java language.

Associated Work Packages:

- Provide and maintain ICC help-desk
- Provide and maintain information database
- Provide and maintain expertise database
- Help desk staff
- Information Handler
- Information mining tools
- Staff training
- Create and maintain website
- Evaluate Scientific Proposals*
- Provide User Support*
- Provide PR materials*

*These work packages are probably HCSS work packages since they’re derived from HCSS usecases.

UC-OTH001: Collaborate with other ICCs or HSC

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **5 October 2001**

Brief description:
This use case describes the formal interface that is used to either initiate or respond to a demand regarding a potential area of common work between the SPIRE ICC and the PACS and HIFI ICCs. This demand may originate inside or outside of the SPIRE ICC.

Phase:
ILT, IST, PV, Operations

Actors:
PA: Problem analyst
IM: ICC manager
IH Information handler

Triggers:
A request is received from another ICC regarding a possible area of common work.
A member of the SPIRE ICC identifies a possible area of common work with another ICC.

Preconditions:
The ICC managers are in regular contact with each other. This is already guaranteed by the existence of the HCSSMG (HCSS management group) and the Herschel Ground-Segment Advisory Group, on which the ICC managers sit.

Minimal post-conditions:
A response is provided to whoever triggered the use-case.

Success post-conditions:
New work packages are created and allocated manpower within the ICC. Milestones are set so that the ICC manager can follow the progress of the task. Contact points between the SPIRE ICC and the other bodies are designated.

Stakeholders and interests:
IM from other instruments: avoid duplicating the work.
SPIRE IM: rationalize developments

Main Success Scenario:
1. IM: Receives a request concerning a possible common work area
2. IM & PA: Assess the request internally in the SPIRE ICC
3. IM: Contacts other ICC/HSC managers for work-package distribution

4. IM: Designate contact point for work-packages and milestones
5. IM: Sets-up a team inside the ICC to complete the designated work-packages
6. PA: Reports to IM on completion of milestones
7. IH: Distributes this report to all interested parties.

Extensions:
2a: IM: Concludes that the request cannot be satisfied:
2a1. IM: Writes a report explaining the negative conclusions
2a2. IH: Distributes this report to all interested parties.

6a: PA: Reports to IM on the impossibility to complete a milestone
6a1. IM: Contacts other ICC manager for coordination and possibly loop to step 2 or 3, depending on the nature of the problem.

References:

Open Issues:
At the end of 7 and 2a2, the IH will sometimes have to distribute information outside of the SPIRE ICC. A method for achieving this is needed.

I could not find a general use case for interfacing with other ICC, there is one for ICC interface with the MOC, but not with the HSC or with other ICC.

Comments:
We expect most of the common work to be initiated in fact by the HSC.

This use-case does not preclude direct communication between team members of different ICCs. But ICC managers should be made aware of any significant joint project and milestone.

The main success scenario handles both inside-out and outside-in situations. Step 1 of the MSS is “IM receives a request concerning a possible common work area”. In the outside-in case, this request comes from another ICC or from the HSC. In the inside-out case, it comes from one of the SPIRE ICC members or even from the IM himself.

At step 6, PA stands for whoever is doing the work-packages. For instance he could be a software developer.

Related work packages:

Define ICC management structure	This use-case relies heavily on the existence of a clear management structure, with for instance regular progress meetings, clear visibility of each ICC member’s responsibility.
Define ICC information dissemination procedure	Here we need to clearly define our policy for making engineering or calibration

	information known to the other ICCs.
Define and manage ICC work packages	A method for creating new and/or following the evolution of ICC work packages needs to be designed.
Information handler	Including a method to forward information outside the ICC

UC-SDR001: Reduce SPIRE Data using IA

Level: **summary**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **12 Jan 2005**

Brief description:
This use-case describes the steps that a user (see comments for definition) performs to produce calibration/science/engineering quality data using a tailored set of reduction steps or the Standard Product Generator (see notes). This use-case does not describe any subsequent analysis of the data. The nature of IA means that this is a single 'use' of the reduction procedure; IA is not a rigid pipeline.

Phase:
All phases

Actors:
DP (primary actor)

Triggers:
The user wishes to reduce SPIRE data

Preconditions:
User has received the IA (inclusive of manuals, cookbooks etc), a library of reduction steps and the relevant SPIRE 'raw' data.

Minimal post-conditions:
Original SPIRE data are uncorrupted.

Success post-conditions:
User has produced reduced data which are ready to be analysed.

Stakeholders and interests:
ICC Scientist, ICC Software Developer: wants to deliver usable reduced data to the community.

Main Success Scenario:

1. DP: receives SPIRE 'raw' data.
2. DP: imports data into SPIRE-IA (UC-DAS101)
3. DP: examines observation history (UC-DAS104)
4. DP: (for PHT) Subtract off-source data (UC-PHT121)
5. DP: identifies and flag bad data (UC-DAC101)
6. DP: filters data (UC-DAC102)
7. DP: removes effects of instrument crosstalk (UC-DAC103)
8. DP: removes pixel-to-pixel variations (UC-DAC104)

9. DP: transforms between sky and spacecraft coordinates (UC-DAC107)
10. DP: resamples data spatially (UC-PHT108)
11. DP: converts from engineering units to astrophysical units (UC-DAC108)
12. DP: determines colour correction (UC-PHT110)
13. DP: applies colour correction (UC-PHT111)
14. DP: visualises data product (UC-DAS108)

Extensions:

1a1 DP: Queries database for SPIRE data.

3a1 DP: Runs the Standard Product Generator*

*a1 DP: Exports data to popular format (UC-DAS103)
*a2 DP: Runs an external reduction procedure on data
*a3 DP: Import non-IA format data (UC-DAS102)

4a1 DP: for FTS data
4a2 DP: Identify and flag bad data (UC-DAC101)
4a3 DP: Remove effects of instrument crosstalk (UC-DAC103)
4a4 DP: Convert counter position to mechanical position (UC-FTS104)
4a5 DP: Convert mechanical position to OPD (UC-FTS106)
4a6 DP: Phase Correct (UC-FTS107)
4a7 DP: Regrid data to obtain ZPD (UC-FTS108)
4a8 DP: Responsivity correction (UC-FTS109)
4a9 DP: Remove pixel-to-pixel variations (UC-DAC104)
4a10 DP: Apodise (UC-FTS113)
4a11 DP: Conversion from engineering units to astrophysical units (UC-DAC108)
4a12 DP: Fourier Transform scan (UC-FTS114)
4a13 DP: Produce a 3D data cube (UC-FTS117)
4a14 DP: Transform between sky and spacecraft coordinates (UC-DAC107)
4a15 DP: Visualise any data product (UC-DAS108)

References:

UC-DAS101 UC-FTS1 04
UC-PHT121 UC-FTS106
UC-DAC101 UC-FTS113
UC-DAC103 UC-FTS114
UC-DAC104 UC-FTS117
UC-DAC107 UC-FTS107
UC-DAC108 UC-FTS108
UC-DAS-103 UC-FTS109
UC-DAS102 UC-PHT108
UC-DAC102 UC-PHT110
UC-PHT111

Open Issues:

Comments:
MSS Step 1: Delivery can be database query (UCF-489) or CD/tape sent to user.
Users/DP: Astronomer, Calibration Team/Scientist, Instrument Scientist
*Standard Product Generation: a pipeline consisting of the basic reduction steps.
This usecase is an eloboration of UC-FTS001 (Process FTS data) and UC-PHT001 (Reduce PHT data) whose comment sections are pasted here.

IA should be able to process groups of observations as if they were a single observation. As an example it may be more efficient to perform a given processing step once and apply it to a group of observations, rather than performing it for each observation (e.g. guessing the initial conditions in transient correction).
IA should also be able to access other data when they are needed to reduce a given observation.

IA should minimize the number of repetitive processes.

Note that IA should not overrule proprietary rights.

It is assumed that every ICC use-case that is using this use-case contains a validation and verification of the data products. This is the reason why here we only assess the quality of the data products.

Any operation that modifies the data has two modes: creating a new artifact or overwriting the old one.

FTS
User-level use-cases:
These are presented in the order they have appeared to our analysis.

We have to decide how far IA has to go before exporting (format to be defined) to another data processing package to carry out the later reduction/analysis steps.

- **Extraction of Artifact from HCSS => UCF-489 "Retrieve archive artifact".**
  Includes all SPIRE relevant data concerning observation with the FTS.
  Comment: depending on the actor (e.g. CS or AST), the calibration tables and other artifacts are found in different ways (interactive or not). A clear warning message has to be sent if a calibration table is used BUT doesn't correspond to the 'latest/updated/recommended' one.

- **Flag bad and missing data => UC-DAC101**

Self explanatory for dead (or damaged) detector(s). Needs construction of the instrument status history of both the FTS mirror speed (as loss of speed stability, …) and the telemetry defects (as magnetic storm, interruption of the transmission, …) to identify bad portion(s) of the data.

- **Store data (to local store) => UC-ENG003**

In general data should be stored at a point in the pipe-line that one may wish to return to (to run new procedures) without having to go further back. So it is likely that data should be stored after time-consuming or stable processes.

- **Reconstruction of each scan/interferogram => UC-FTS103**

Definitions:

- *Movement*: one direction mirror travel.
- *Interferogram*: dataset concerning one detector and one movement.
- *Scan*: set of the N+M interferograms of one movement. N and M are the number of detectors in the two FTS bands.

- **visualisation of raw data (interactive) => UC-DAS108**

Visualisation routines would be required at some stages during the data processing. I.e., at least, we would like to be able to examine the most raw data products, before any processing has occurred, and we would like to be able to visualise the data in physical units. We need to be able to visualise data by scan and by detector.

- **Electrical cross-talk removal => UC-DAC103**

Self explanatory (but may be difficult to implement)

- **Oth order deglitching => UC-DAC102**

Replace or flag affected points in each interferogram for main glitches (mainly cosmic rays). The signal is by definition varying very quickly: the deglitching could use neighbourhood data and the fact that data are oversampled by a factor 4.

- **Convert position counter to mechanical mirror position => UC-FTS104**

Requires a calibration table. An optical incremental (relative) counter is used to determine the position; it can happen that some steps are missed. In the travel range covered by the absolute sensor (LVDT), its information can be used to correct from missing step(s) of the optical relative counter.

- **Generate array of signal vs. position => UC-FTS105**

Two methods should be considered: either interpolate position to time of the detector sample or interpolate detector signal to time of the position sample. It is foreseen that this step should also be able to restore missing/bad data with the help of the flag information (see UC-DAC101 and UC-FTS104).

- **Convert mechanical position to OPD (optical path difference) for each detector => UC-FTS106**

Self-explanatory. Require also a calibration table, determined at the ground test phase.

- **Phase correct => UC-FTS107**

There may be two phase-corrections: optical and electronic (detector band pass and multiplexer delay).

- **Re-grid data to obtain a ZPD point => UC-FTS108**

Aim at having ZPD (zero path difference) at one measurement.

- **Responsivity correction = UC-FTS109**

Needs change in responsivity to be computed from calibration measurements. There will be long- and short-term variations, according to the responsivity drift of each detector and to the velocity fluctuations of the mirror (frequency response of each detector) respectively. Needs calibration tables.

- **Correct for time-dependent variation in flux => UCFTS110**

Variation in flux may come from the emission of either the HSO telescope or the FTS internal calibrator (Black Body). This may require access to the history database.

- **Correct for position-dependent variation in flux => UC-FTS111**

The variations result from the efficiency in the interference of the two beams. Requires a calibration table (ground test phase).

- **Transform between sky and spacecraft coordinate systems => UC-DAC107**

This is self-explanatory (but may be difficult to implement).
This step need to be done for each movement of the mirrors (pointing drift).

- **1st order deglitching => UC-FTS112**

Replace or flag outliers (median-like method).

- **Apodise (removal of outlying frequency signals) => UC-FTS113**

Optional, to be determined by the user.

- **Fourier Transform individual scans => UC-FTS114**

Produce a spectrum per detector per scan

- **Remove instrument signature => UC-FTS115**

Remove (telescope – calibrator) emission. This step may need to be performed before FT.

- **Remove pixel-to-pixel sensitivity variation (flat-field) => UC-DAC104**

Requires a calibration table.

- **Convert to physical units (flux calibration) => UC-DAC108**

E.g. Jy or W/m^2.cm^{-1} or W/m^2. Requires calibration tables.

- **Produce a spectrum per sky pixel => UC-FTS116**

Average over scans.

- **Produce 3D data cube => UC-FTS117**

3 dimensions are for the two celestial coordinates and the radiation frequency.

- **Select and display map over spectral range => UC-DAS108**

- **Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data) => UC-DAS104**

- **Information and error messaging system => UC-DAS105**
  The level of messages can be adjusted by user.

- **detect and identify lines => UC-FTS118**

Some of the following use cases can also be included as FTS use cases:

- UC-DAC105: Identify and flag data on any criteria
- UC-DAC106: Background subtraction
- UC-PHT108: Resample and combine data spatially and/or temporally (includes statistics)
- UC-PHT110: Determine colour correction
  Rem: The colour correction can be computed from an FTS observation according to the PHT band passes.
- UC-DAS103: Conversion of output data to popular data formats
- UC-PHT113: Detect sources
- UC-DAS106: Interactive Analysis Documentation (includes online, interactive and hardcopy)
- UC-DAS107: Record data reduction history.
- UC-PHT121: Subtract off-source data (demodulate data)
- UC-DAS101: Read and prepare data-frames for IA processing
- UC-DAS102: Import non-IA format data

PHT
User-level use-cases:
These are presented in the order they have appeared to our analysis

- UC-DAC101: Identify and flag bad data.
- UC-DAC102: Filter data on any criteria
- UC-DAS108: Visualize any data product
- UC-DAC104: Remove pixel-to-pixel sensitivity variation (flat-field)
- UC-DAC105: Identify and flag data on any criteria
- UC-DAC106: Background subtraction
- UC-DAC107: Transform between sky and spacecraft coordinate systems
- UC-PHT108: Resample and combine data spatially and/or temporally (includes statistics)
- UC-DAC108: Conversion from engineering units to astrophysical units (flux calibration)
- UC-PHT110: Determine color correction
- UC-PHT111: Apply color correction
- UC-DAS103: Conversion of output data to popular data formats
- UC-PHT113: Detect sources

- UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
- UC-DAS105: Information and error messaging system
- UC-DAS106: Interactive Analysis Documentation (includes online, interactive and hardcopy)
- UC-DAS103: Remove effects of instrument cross-talk
- UC-DAS107: Record data reduction history
- UC-PHT121: Subtract off-source data (demodulate data)
- UC-DAS101: Read and prepare data-frames for IA processing
- UC-DAS102: Import non-IA format data

POF1: All except UC-DAS103 and UC-PHT113
POF2-POF9: All
Dealing with the internal calibrator is done in UC-DAC108
Pointing reconstruction is performed in UC-DAC107. Inside that use-case there will be a lower-level use case dedicated to pointing reconstruction as a stand-alone activity.
Reconstruction of the actual timing of the data is done by the telemetry ingestor and is thus out of the scope of this use-case.
What do we do about special engineering modes?
What about the memory effects?

Related work packages:
See individual usecases

User-level Use-cases

UC-AIV101: Update OBS

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **18 July, 2001**

Brief description:
This use case describes how an on-board software image that is approved is delivered to the HCSS for uploading to the instrument.

Phase:
All phases except post-operations.

Actors:
IH: Information handler
IE: Instrument engineer
CC: Configuration controller
HSC: Herschel Science Center

Triggers:
The MRB approves a change to the OBS as per UC-OBS102 step 7.

Preconditions:
A new image is available for delivery.

Minimal post-conditions:
A new image is not uploaded, there is no change to the current image on-board.

Success post-conditions:
A tested and verified image is uploaded to the instrument.

Stakeholders and interests:
Instrument engineer wants to have the new OBS image uplinked.

Main Success Scenario:
1. CC: Retrieve the OBS image from the local SPIRE database.
2. CC: Prepare delivery documentation.
3. CC: Deliver the OBS image, associated documentation, and an uplink scheduling request to the HSC.
4. HSC: upload the new OBS image.
5. IE: test the new OBS image on the instrument.
6. IH: Notify the instrument users that the change has been made.

Extensions:
4a: HSC refuses to upload new OBS image.

4a1: IE: Analyze HSC problem report.

4b: HSC: If the uplink is not successful then uplink previous OBS image.
4b1: IE: Analyze HSC problem report.

6a: Test on new OBS image fails.
6a1: CC: request uplink of previous OBS image.
6a2: IE: file a problem report.

References:
UC-OBS101

Open Issues:
If the uplink to the instrument fails what is the mechanism for the MOC to inform the ICC?

During ILT and IST, the HSC may not exist or be replaced by another entity.

Comments:
In order to actually test the new OBS image, we might have to extend the communication period with the spacecraft.

Careful co-ordination is required between the update of the OBS and the subsequent downlink data processing. In fact this is more general: updating the OBS has the potential to impact all the ground segment (MIB, CUS…)

It is expected that the MOC will generate the commands to patch the in flight OBS image from the delivered image using the SCOS 2000 OBS management facility. The same mechanism can in principle be used for ILT and for IST.

We have no control on expected or required turnaround time between an ICC delivery and the new image being uploaded.

Work-packages:
- OBS test system
- OBS test plan
- Problem Report/SCR system

UC-AIV102: Generate, validate, and verify scripts and observation requests

Level: **user**
Scope: **SPIRE ICC**
Version: **1.3**
Status: **issue**
Date: **18 June 02**

Brief description:
This use case describes how instrument scripts and observation requests are generated, validated, and verified. By validated we mean that the object is safe for the instrument and by verified that it makes the instrument do what we want it to do. The mechanism for constructing a sequence of commands may vary from ILT to operations.

Phase:
ILT -> Operations

Actors:
IT: Instrument Tester
IE: Instrument Engineer
MRB: Materials Review Board
CC: Configuration controller

Triggers:
A test, engineering or calibration observation is required.

Preconditions:
A system to convert observation requests into observation executions has to exist (CUS).
A system to execute scripts has to exist (Test Control).
S/W tools are available for generating and editing observation requests and scripts (CUS and Test Control).
HCSS is available for storage and retrieval of scripts and observation requests, may not be necessary during early ILT.
A system to validate the list of mnemonics resulting either from the script or from the observation request has to be available. The HCSS provides a system to validate observation executions (described in UC-CUS001).

Minimal post-conditions:
No script or observation request can be generated that could harm the instrument or spacecraft.

Success post-conditions:
A validated script or observation request is placed in the database.

Stakeholders and interests:
CC: needs to ensure that any element entering the system are properly checked

Main Success Scenario:

1. IE: Start and configure Test Control or CUS.
2. IE: Retrieve a script or an observation request from the local database.
3. IE: Modify the script or the observation request.
4. IT: Validate the script or the observation request using validation s/w described in the preconditions.
5. TC/IT: Run the script or the observation request (on instrument simulator UC-ENG101, or on the instrument itself).
6. IE: Inspect the result and produce a verification report.
7. MRB: Decide if the new script or the observation request should be put in the HCSS database
8. CC: Put the new script or the observation request in the HCSS.

Extensions:
2a: Generate a new script or command request.

5a. Test of the new script fails:
    5a1: IE investigate failure report
    5a2: IE go back to 3

References:
UR-AIV-100
UC-CUS001
UC-ENG101

Open Issues:

Comments:
Mnemonics: these are defined in the SCOS2000 database, these are the basic commands we can send. They are contained in the MIB. (unlikely to change). They can have parameters. We (ICC) will never use directly these.
Command sequence: a series of mnemonics build in SCOS2000 and also stored in the MIB. A sort of macro of mnemonics. Here again a command sequence can have parameters. We (ICC) will never use directly these.
Script: available in EGSE/ILT as part of the Test Control. These scripts can combine command sequences and mnemonics, plus some control loops, if/then…Test Control is able to request from the HCSS that an observation request is converted into an observation execution.
Observation execution: this is an instantiation of an observation. It has the form of a collection of mnemonics with their parameters and with a list of relative times between them.
Observation request: resides in the HCSS and is the equivalent to an AOT (AOTs are in fact a subset of the observation request). The CUS is the environment in which these requests (which are scripts) can be defined.

Another possibility after step 5a1 would be to file a problem report if no modification of the script results in a successful test.

When the test script is executed on the simulator (step 5 of the Main Success Scenario), we are using UC-ENG101, but skipping the first two steps, where the writing of the script occurs.

Related work-packages:

- Produce validation software to validate scripts and observation requests
- Produce instrument simulator to verify scripts and observation requests.

UC-AOP101: Plan an observation

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **19 July, 2001**

Brief description:
This use-case describes the steps involved in the preparation of an observation, from the definition of the science objectives to the choice of an observing strategy.
This is in fact an HCSS use case:UCF-484 perform proposal planning. This is the reason why all items are left blank.

Phase:
Call for guaranteed time proposals, onwards.

Actors:

Triggers:

Preconditions:
A time estimator exists.

Minimal post-conditions:

Success post-conditions:

Stakeholders and interests:

Main Success Scenario:

Extensions:

References:

Open Issues:

Comments:

Related work packages:
Main work package: write time estimator.

UC-CAL101: Update Calibration Plan

Level: **user**
Scope: **SPIRE ICC**
Version: **1.3**
Status: **issue**
Date: **12 Jan 05**

Brief description:
This use-case describes the process of creating and updating the Calibration Plan. The Calibration Plan describes the full course of actions necessary to calibrate SPIRE.
In particular, it includes the information required to calibrate the instrument data, and the methods by which that information will be obtained, for example from preparatory laboratory measurements of the instrument, observations performed on other instruments, including Herschel and other observatories, observations during operations, modeling, and archival resources.

Phase:
All phases.

Actors:
CS: Calibration Scientist (with instrument expertise as well)

Triggers:
A Calibration Plan is required to define tests on the ground and in-orbit.
A problem report has been filed and analysis reveals that the Calibration Plan needs to be updated.

Preconditions:
Calibration requirements are known.
A problem reporting system exists.

Minimal post-conditions:
The current plan is unchanged.

Success post-conditions:
A better calibration plan is defined.

Stakeholders and interests:
The Test Scientist needs the plan in order to define the tests to be performed on the ground.
The Astronomer needs the instrument calibrated to perform science observations.

Main Success Scenario:
1. CS: Retrieve the current plan
2. CS: Retrieve the current calibration report

3. CS: Propose changes to the Plan, including assessment of effectiveness, impact, and risk.
4. CS: approves the changes.
5. CS: Make the changes to the Calibration Plan.

Extensions:
1a : CS: Create the first Calibration plan.

4a: Changes to the plan are not approved
4a1. CS: go back to step 3 or exit with no update

References:

Open Issues:
The details of the calibration documentation are TBD. Currently there is an overall calibration requirements document which has fed through to a calibration plan which outlines at the top level all calibration tables required by SPIRE. Note parameter conversions are not considered as calibration here. The calibration plan is then applicable to lower level calibration documentation which will encompass separate plans for testing PV and routine operations phases.
For testing the CQM the tests have already been detailed in a tech note and via detailed procedures. It is expected that the calibration plan for ILT will then link the tests with the tables produced. It is likely a similar procedure will be followed for PV phase. For routine operations, there will be an ongoing plan that is expected to be reviewed and, if necessary, updated weekly.

Comments:
We expect that changes in e.g. data reduction strategy requiring new calibrations will go through the problem report system.

The “calibration plan” referred to in the HCSS Use Case Definitions V1.0 is a Herschel calibration observation plan. This is another “calibration plan” which describes the full course of actions necessary to calibrate SPIRE.

Work-packages:
- Define calibration requirements
- Problem Report/SCR system
- Define calibration plan

UC-CAL102: Schedule Calibration Observation

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7 December 2001**

Brief description:
This describes how the planned calibration observations in a given time period are scheduled.

Phase:
ILT->Operations

Actors:
CS: Calibration Scientist
CC: Configuration controller
IE: Instrument engineer (in extensions)

Triggers:
Periodic: Observations need to be scheduled for this time period.
Due to a problem report we have to schedule specific calibration observations.

Preconditions:
A calibration plan exists.
A scheduling system exists.
The observations have been agreed.

Minimal post-conditions:
No observation which harms the instrument is scheduled.

Success post-conditions:
All agreed observations are scheduled.

Stakeholders and interests:
Calibration team is in charge of implementing the Calibration Plan.

Main Success Scenario:
1. CS: Check the planned observations can be carried out
2. CS: Implement the planned observations using the CUS
3. CS: Check the total time fits in with time allocated
4. CS: Pass observation executions plus any timing constraints to CC
5. CC: Deliver scheduled observations and timing constraints

Extensions:

1a: Observations can not be carried out because previous data e.g. uplink file does not exist
1a1. CS: Exit this use case and return to UC-CAL001 step 1

1b: Observation can not be carried out because object is not visible
1b1. CS: Exit this use case and return to UC-CAL001 step 1

2a: CUS needs updating to implement observation
2a1. IE: Update the MIB if necessary (UC-CUS103)
2a2. IE: Update the CUS (UC-CUS102)
2a3. IE: Implement observation
2a4. CS: Return to step 3

3a Time does not fit within allocation
3a1. CS: Exit this use case and return to UC-CAL001 step 1

References:
UC-AIV102
UC-CAL001
UC-CUS001
UC-CUS102
UC-CUS103

Open Issues:
If a CUS update is required, a decision must be taken whether we are able to do this without testing on the instrument. It is likely that the guideline for this will be that if a change implies a change to the MIB, it should be tested on either the simulator or the flight spare. If the MIB is unchanged, it will not need to be tested as the individual commands would have been tested for instrument safety (see also UC-AIV102, UC-CUS001, UC-CUS102, UC-CUS103)

Comments:
Step 2 does involve possible modification of the most basic way the instrument is operated. Maintaining this possibility is a major aspect of this use-case and of calibration as a whole.

Delivery of the observation executions is made to the CC because we consider the CC as being our interface with the HSC. It is also the way to make the ICC as a whole aware of the scheduling having been performed.

During operations, the scheduling system will have to include a visibility tool

Related work packages:
- MIB updates
- CUS updates
- Observation scheduling
- Configuration controller

UC-CAL103: Scientifically validate a calibration/IA artefact update

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **19 June 2002**

Brief description:
This use-case describe how the ICC, after having decided that a new version of the calibration/IA system is to be released, goes to the process of validating it with respect to scientific criteria negotiated between the ICC and the HSC.
Scientific validation of a calibration update is a purely HSC concept. When a calibration update is made, it is the HSC that will require us to scientifically validate the new system. To be even clearer, this use-case covers those aspects of the scientific validation imposed on the ICC by the HSC for making IA systems available.

Phase:
PV to post-operations

Actors:
CS: Calibration scientist
CC: Configuration controller
IH: Information handler
HSC: Herschel Science Center

Triggers:
The ICC has decided to release a new version of the IA system.

Preconditions:
A new version of the IA system is available and has been approved for release.
The scientific criteria have been negotiated and are known.

Minimal post-conditions:
A report on the validation exercise is produced.

Success post-conditions:
The system is scientifically validated and a report is produced.

Stakeholders and interests:
HSC is the one imposing this exercise on the ICC.

Main Success Scenario:
1. CS: defines the test and test cases for scientific validation
2. CC: delivers list of test cases to HSC
3. HSC: accepts the test cases as complete for scientific validation

4. CS: runs the validation tests
5. CS: writes the scientific validation reports
6. CC: delivers report to HSC
7. IH: delivers report to all interested parties in the ICC

Extensions:
3a: HSC rejects the test cases as incomplete
    3a1. CS: goes back to step 1

4a: Validation tests fail
    4a1. CS: Files a problem report

References:

Open Issues:
Science validation has not yet been considered by the HSC, but it is more than likely that the HSC will impose it on the ICC. Therefore we have written this use-case to take care of the possibility.

Comments:
This use case can be triggered by a software update rather than by a calibration update.

After the extension 3a, the ICC could very well not accept the HSC decision. In this case the normal route is to go back to the scientific criteria negotiation.

Related work packages:
- Negotiate scientific validation criteria
- Perform scientific validation of updates for the HSC
- Information handler

UC-CAL104: Generate calibration report

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **19 June 02**

Brief description:
This use case describes how a calibration report on the properties and status of the instrument is produced and made available. The report is a public document detailing properties and status that are of interest to the other ICCs and the HSC.

Phase:
ILT-> Post Operations

Actors:
CS: Calibration scientist

Triggers:
Periodic

Preconditions:
A Documentation system exists
A Calibration plan for the time period exists

Minimal post-conditions:
A report on calibration activity is archived.

Success post-conditions:
A report on calibration activity is archived, including any observations done, the reasons for carrying out the observations, the assessment and analysis of the data gained and any updates to calibration artefacts made in the time period.

Stakeholders and interests:
HSC: Information content sufficient to feed-back to astronomers
ICC manager: Information content sufficient to alert ICC and plan future work packages within the ICC.
PI: Information content sufficient to alert the SPIRE consortium of potential problems.

Main Success Scenario:
1. CS: Consults the calibration plan for what was planned in the period since the last report.
2. CS: Checks the observations completed in time period since last report.
3. CS: Checks current state of analysis in all calibration areas.
4. CS: Checks the updates of Instrument Calibration made since the last report (UC-CAL105)
5. CS: Checks the delivery status of calibration artifacts.

6. CS: Writes calibration report using create or Update Calibration Document (UC-ICC104)
7. CS: Approve updates

Extensions:

References:
UC-CAL105
UC-ICC104

Open Issues:
Anomalous behaviour should form part of the calibration report but this may have different purposes to the problem reporting system: aspects of instrument behaviour which may or may not lead to a problem report may be under investigation on this time period and will only appear in the calibration report. Other instrument problems e.g. one mode inoperative may not affect data quality as such but will affect how and why observations are scheduled.

At present no use case covers calibration analysis, therefore it is unclear how the anomalous behaviour will be detected and reported.

Comments:
If our calibration requires using observations with HIFI or PACS, these should be included in the calibration plan. Therefore the Main Success Scenario does not explicitly refer to these instruments, even though they may be used in calibrating SPIRE.

Related Work Packages:
- Calibration Plan
- Calibration Scientist (include handling anomalous behaviour)
- Define and create calibration data processing and analysis software.
- Documentation system
- Calibration data-base

UC-CAL105: Update Calibration Artefact

Level: **user**
Scope: **SPIRE ICC**
Version: **1.2**
Status: **issue**
Date: **18 June 02**

Brief description:
This use-case describes how a Calibration Artefact (see definition in the Comments section) is updated.

Phase:
All phases.

Actors:
CS: Calibration scientist
CC: Configuration Control
IH: Information Handler

Triggers:
The implementation of the Calibration Plan has produced new data to be analysed.
A problem report has been filed that implicates the calibration. It has been analysed and a calibration artefact to update has been identified.

Preconditions:
Calibration requirements are known.
A Calibration Database to store calibration artefacts exists.
The scientific goals and expected performance of the instrument are known.

Minimal post-conditions:
The artefact is not updated and any previous version is not erased.

Success post-conditions:
The artefact is updated.

Stakeholders and interests:
The Calibration Scientist wants to get the result of the Calibration Plan he or she has implemented.

Main Success Scenario:
1. CS: Retrieve the relevant data.
2. CS: Use the relevant data to produce the update to the artefact in an ICC sandbox
3. CS: Validate and verify the updated artefact.
4. CS: Produce a report on the update.
5. CS: Place the update in the ICC test environment
6. IH: Inform all interested parties of the update.

Extensions:
3a: updated artefact is not validated or verified.
3a1. CS: file a problem report.

References:

Open Issues:
There has to be a mechanism to make sure that the report that is produced during the update finds its way to the Calibration Report mentioned in UC-CAL104 (Generate Calibration Report).

Comments:
The Calibration Database could contain: Lab test data, “ground-based” observations of known sources, in-flight observations of known sources (SPIRE or other instruments), models of sources, software.

“Calibration Artefact” is expected to include both procedures and updating calibration tables.

See the glossary for the definition of ICC sandbox and test environment.

Related Work Packages:
- Define Calibration requirements
- Define and Create Calibration Database (uplink and downlink)
- Define and Create Calibration data processing and analysis software.
- Define Calibration plan
- Fill Calibration Database
- Maintain Calibration (including software, database and plan)
- Create reporting system.
- Create/maintain sandbox and test environment

UC-CAL106: Simulate science performance

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 02**

Brief description:
This use case describes how an instrument simulator(s) is used to model the behaviour of the instrument in order to produce synthetic data. The data will be used for producing and testing data reduction algorithms for use in QLA/IA, for validating a new observing procedure, or for investigating the consequences of a modification in the instrument properties. The science content of the data output should contain all relevant instrument characteristics (those that impact on the scientific output).

Phase:
ILT->Post-operations

Actors:
SSD: Scientific software developer
IT: Instrument tester
EKS: Expert Knowledge System

Triggers:
- A high-level reduction algorithm or sequence of algorithms needs to be produced or tested, requiring input data.
- The science objectives of an observing mode need to be tested and validated requiring a complete simulation.
- The scientific consequences of a modification in the instrument capabilities need to be investigated.

Preconditions:
Any necessary input data to the simulator(s) is available (source parameters, observing modes).
All technical parameters (e.g. sensitivities, noise sources and amplitudes) of the instruments have estimated/known values.
The formatting of the data produced by the instrument is defined.

Minimal post-conditions:
A properly formatted stream of data is produced.

Success post-conditions:
A stream of data is produced that has exactly the same characteristics in term of formats as real data, so that any system that is able to ingest and treat real data can ingest and treat these simulated data. This data also contains all the known noise contributions and artefacts produced by the real instrument.

Stakeholders and interests:
Problem Analyst: can use the system to investigate a problem report with the instrument.
Astronomer: need to design observing/data reduction strategy.
Helpdesk: can provide guidelines as to the best instrument usage strategy.

Main Success Scenario:
1. IT: Retrieves information describing the sky observed by the instrument (astrophysical units with spectral dependence for FTS)
2. IT: Retrieves information describing how the observation is performed (the observing mode)
3. IT: Prepares input for simulator
4. IT: Executes the observation on the simulator
5. IT & EKS: Assess the scientific quality of the output data
6. EKS: writes report on the data quality
7. IH: Inform all relevant parties of these reports.

Extensions:
4a: Execution of the observation fails:
4a1. IT: writes a problem report

References:

Open Issues:

Comments:
This simulator is essentially a detailed model of the instrument detector behaviour (including signal induced by the instrument operation, e.g. telescope background, stray-light, electronic effects). This distinguishes it from other possible forms for an instrument simulator, e.g. a simulator that models the telecommand logic (for safety checking) or a time estimator tool.

There are potentially four simulators in development:
- Instrument simulator which plugs into MOC s/c simulator (Stockholm)
- Cold parts simulator for the FPU (Saclay)
- Detector Readout and Control Unit Simulator, generates HK and some science TM. Used for testing interface to s/c (Stockholm)
- Simple telemetry simulator for feeding packets into SCOS2000 (ICC)

Any simulator used will need to be maintained at the ICC except the spacecraft simulator which is under the responsibility of the EGSE.

Note that step 5 could be rewritten in two steps: 1-reduce the simulated data, and 2-assess scientific quality of result.

For high-level IA testing and observing mode definition, one only needs the images from the detector. However if one wants to be able to test the low-level algorithms of IA, those that convert the data frames into images from the detector or FTS scans,

the simulator has to be able to produce its output in data frames format (packets would actually also be desirable). We obviously want the second kind of simulator.

This simulator is the one that will be produced and used by the ICC, so it has to provide data in the format expected from the real instrument.

In extension 4a, failure is defined against what we expect the instrument to do. It includes software crashes but also rubbish or invalid science data where valid data are expected.

Related work packages:

Access instrument parameters	This is to make sure that the data stream produced simulates the instrument in its current state.
Access telescope parameters	This is to make sure that the data stream produced simulates the telescope in its current state.
Define telemetry format	To output the data in the correct format
Define the sky	Access models, archives, this may include a ground-based preparatory program
Define interactive analysis	The whole purpose of this use case is to process the data, therefore tools to do that have to exist.
Define quick-look analysis	Used in case only basic diagnostics are needed.
Define nominal instrument test	Since we aim at diagnosing problems or checking performances, the nominal state of the instrument should be defined to serve as a benchmark
Define instrument AOT	We need to know them to produce relevant data, but we may also modify them depending on the result of the Use-Case.
Maintain the science simulator	This is the blood of the use-case.
Make the science simulator	This is the heart of the use-case.
Make the information handler system	For transferring reports
Make the problem report system	Use in case of failure.
Store data for future analysis	Some simulations may be of long lasting interest, or may require too much time to analyse immediately.

UC-CON101: Capture Consortium expert knowledge

Level: **user**
Scope: **SPIRE ICC**
Version: **1.2**
Status: **issue**
Date: **12 Jan 05**

Brief description:
This use case describes how information on all aspects of SPIRE (instrumental and scientific) is obtained from the expertise available in the Consortium. New information is obtained from an expert(s) and stored in some form, or if the information exists from previous requests it is recovered from the store. The status of a new request is logged. Unsolicited information is also dealt with.

Phase:
All phases

Actors:
EKS: Expert knowledge system
ICC-S: ICC scientist, here on a quest for answers (who might even be ICC-HD)
ICC-HD: ICC Helpdesk
IH: Information Handler

Triggers:
Information is requested of ICC that requires knowledge from the Consortium to answer. This request may be internal or external to ICC. Unsolicited information from the Consortium may also be made available to ICC.

Preconditions:
Interfaces between Consortium and ICC need to be in place, e.g. Helpdesk.
A Consortium expertise database exists.
A database for information storage exists.
Interface tools to information database, especially search methods are in place.

Minimal post-conditions:
A request for information is made by ICC-S. The request and any responses are logged.

Success post-conditions:
Requests are answered to the satisfaction of the instigator and the results of any information exchanges are logged.

Stakeholders and interests:
Consortium wants information to be distributed to those that need it.
ICC wants information to be distributed to those that need it.
KN needs the information they seek.
IH needs to track information in the database.
HD needs to ensure queries are adequately dealt with.

Main Success Scenario:

1. ICC-S determines which EKS can answer their question
2. ICC-S contacts EKS for information
3. EKS member returns the required information to ICC-S
4. ICC-S forwards the EKS response to IH
5. IH adds EKS response to ICC database
6. IH logs the transaction

Extensions:

1a. Unsolicited information arrives from EKS
1a1. ICC-HD stores the information in the ICC database
1a2. ICC-HD logs the transfer of information

4a. EKS fails to return required information
4a1. ICC-HD searches other sources and makes further enquiries to fulfil ICC-S request

References:

UC-HSC002

Open Issues:

Comments:

This use-case is very much a ‘wish list’ for an idealised way to handle the collection and distribution of consortium knowledge. The resource needs, in terms of both development and operations to run the system outlined above, could be large, if everything is to be cross-referenced, indexed and fully maintained. Indeed I don’t think such an online active and automated encyclopaedia has ever been produced. It will thus be necessary to examine how the aims of this usecase – to archive and distribute consortium knowledge as automatically and efficiently as possible – can be achieved with more realistic resources. Possibilities include use of LiveLink, the use of a newsgroup or mailing-list-like system, with archiving and searching, or the appointment of an ‘archivist’ as part of the ICC-helpdesk to maintain the knowledge database by hand.

However, critical features of the ‘ideal’ system above that should propagate to the implemented version are:

(1) Existence of an ‘expertise database’ so that the appropriate person can be contacted when information is needed. This can be fairly easily collected within ICC, but determining the expertise of non-consortium members may be difficult.
(2) A way of archiving and searching both queries and answers. As a minimum this could take the form of a flat text archive of messages that is then ‘grepped’ to find keywords of interest. Unsolicited information can then be added to this text archive as well as dialogues between ICC-S and the person with the expertise. In this way the IH role is short-circuited, reducing the resources needed for IH, but also allowing things to fall out of the loop. The involvement of ICC-HD is also reduced to providing

the ‘expertise database’, the logging system (eg. mailing list) and tools for making searches, handling the delivery of unsolicited information, and providing support for ICC-S new to, or having trouble with, the system. In this model of operations the MSS is the same, but all the logging and IH transactions occur automatically through a mailing-list-like system.
(3) In addition, relevant information from this database will need to be wrapped up and incorporated into internal and external documentation from time to time. A system with less indexing and cross-referencing will make this process slower and more complex.

This use-case is extensively based on UC-HSC002, which in turn is based on “UCF-121 [Summary]: Provide user support”. The main distinction is that the body that instigates the query (ICC) will be logging the progress rather than the body that receives the query (the Consortium), since the actions of the Consortium are somewhat beyond the control of the ICC.

It is assumed that any search of the existing knowledge databases has already been performed before the query is instigated.

Some aspects of this use-case are now handled by the SPIRE bulletin board system at DAPSAS London.

Adoption of commercial or open-source 'knowledge base' software is a possibility for this system.

We do not yet have a formalised 'expertise database', but it is beginning to become clear where some centres of expertise lie.

Related Work Packages:

- Provision of ICC help-desk
- Provision of information database
- Provision of expertise database
- Help desk staff
- Provision of internal and external documentation
- Information Handler
- Information mining tools

UC-CON102: Disseminate knowledge

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **12 Jan 05**

Brief description:
This use case describes the formal, but general, process of disseminating information originating within the ICC, or passing along information originating outside, to interested parties within HERSCHEL (consortium, other instruments, etc).

Phase:
All phases

Actors:
IH: Information handler
EKS: Expert knowledge system (in the current context, an ICC member providing knowledge or receiving knowledge from an external source)
ICC-HD: ICC Helpdesk

Triggers:
The ICC generates internally, or receives from an external source information that is deemed to be of interest to an audience outside ICC.

Preconditions:
Documentation system exists
Configuration control system exists
Information Handler exists
ICC-Helpdesk exists

Minimal post-conditions:
ICC-Helpdesk logs the information and decision about dissemination.

Success post-conditions:
ICC-Helpdesk determines that submitted information is of interest to outside parties, identifies these, and sends the information to the appropriate ED.

Stakeholders and interests:
Consortium, ICC, PS - to ensure adequate dissemination of information concerning the mission, but also to avoid information overload
Outside source of information - to receive credit for supplied information

Main Success Scenario:
1. EKS: submits knowledge they believe of use outside ICC to ICC-HD, with appropriate justifications and indications of potential interested parties.
2. ICC-HD: assesses the information for its importance outside ICC

3. ICC-HD: determines which parties are likely to be interested in the information
4. IH: Creates or Updates Documents (UC-ICC104) detailing the information.
5. ICC-HD: sends the information to interested parties

Extensions:
2a. ICC-HD determines that the information is not of interest to external parties
2a1 IH Creates or Updates Document (UC-ICC104) detailing the information and ICC-HD assessment.
2a2 ICC-HD provides feedback to EKS

References:
UC-ICC104 – Create or Update a Document
UC-HSC002 – Support HSC Query

Open Issues:

The process by which ICC-HD assesses the importance of the information is currently unclear. Should it just be ICC-HD or is a broader assessment needed? Determination of interested parties is also a significant task. Access to e.g. List of proposers will be useful for this.

Comments:
This use-case does not deal with externally received information queries. These are dealt with by UC-HSC002.

Since information distributed by ICC will have some official weight, step 2 must be handled with care, and may involve substantial iteration and collaboration in the case of external information providers.

Proper attribution for valuable information coming from an external source will be needed.

This UC is so wide-ranging that the external recipients can range from CON, other instrument teams to funding agencies and the press.

Associated Work Packages:
- Create Documentation system
- Set up Configuration control system
- Establish Help-desk
- Create Information Handler
- Assess usefulness of information to outside parties
- Determine distribution list and method

UC-CUS102: Update the CUS database

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 02**

Brief description:
A new or modified operational functionality is needed for the instrument. As all the functionalities reside in the CUS database, updating the database is the only way to obtain this new or modified functionality.

Phase:
ILT, IST, PV, Operations

Actors:
IT: Instrument tester
IE: Instrument engineer
IH: Information handler
OE: Operations engineer (see comment section)
CC: Configuration Controler

Triggers:
The instrument needs to perform a new or updated function (possibly because a problem has been identified).

Preconditions:
There has to exist a problem analysis mechanism.
The CUS has to exist.
There needs to be a test system available.

Minimal post-conditions:
In case of failure, the original CUS database is unchanged.
A record of the actions performed in the use-case is kept.

Success post-conditions:
A verified and validated new or updated operational mode is available.

Stakeholders and interests:
PI wants the instrument to perform the best observations possible.
PST wants to have a reliable CUS database.
CCB needs to keep track of any modification to the system.

Main Success Scenario:
1. IE documents the changes to be made.
2. OE implements these modifications in a test version of the CUS database.
3. IT tests, verifies and validates the new/updated operational mode.

4. CC: allow the changes to be reflected in the CUS database.
5. OE updates the operational database (UC-ICC102).

Extensions:
2a. Changes cannot be implemented in the CUS database (too complex, require another item to be changed/updated first).
    2a1. OE produces a report explaining the reason for the decision
    2a2. ICC re-analyze problem report

2b. Changes cannot be implemented because a new CUS functionality is needed
    2a1. OE files an SCR on the CUS and exit.

3a. Verification, validation or testing fails
    3a1. IH forward test report to all interested parties
    3a2. ICC re-analyze problem report

4a. CCB does not allow the changes to be reflected in the CUS database
    4a1. ICC re-analyze problem report.

References:
UC-ICC102

Open Issues:
It remains to be decided whether the CUS database is going to go through a delivery procedure, or through an update procedure. In the current version of this use-case, we have decided that it goes through the update procedure.

Comments:
This use case assumes that it comes after a problem report analysis that has concluded that the problem resides in the CUS and that it should be updated. This is why there is no step involving whether or not the CUS should be updated.
Operations Engineer: possibly a new role, a person who knows precisely how to use the ICC system in order to do things, how to define database elements…

Related work packages
- Training in using the CUS
- Problem report/SCR system
- Provide instrument test environment and instrument simulator
- ICC configuration control system
- Information handling
- Create the CUS database

UC-CUS103: Update the MIB

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 02**

Brief description:
This use case describes the steps involved in updating the MIB (Mission Information Base) including testing, validation and preparing the MIB for ingestion. The MIB includes CUS mnemonics (TCs), definitions of all the telemetry and telecommand packets, their parameters and their valid ranges.

Phase:
ILT -> Operations

Actors:
IE: Instrument engineer
OE: Operations engineer
IT: Software tester
IH: Information handler
CC: Configuration controller

Triggers:
After problem analysis, it is decided that the MIB needs to be updated. Items that can be updated include:

- Validity ranges for parameters.
- Mnemonics.
- Telemetry packets definitions.

Preconditions:
A MIB editor is available.
SCOS2000 is available.
The HCSS is available (for retrieval and delivery of the MIB).
An equipment to test the MIB on is available.

Minimal post-conditions:
If the update fails, the operations' MIB is unchanged. The reason for the absence of update is recorded.
If the update succeeds, the new MIB does not adversely affect the instrument.

Success post-conditions:
The new or updated MIB is ready for ingestion.

Stakeholders and interests:
PI wants the instrument to perform the best observations possible.

PST wants to have a reliable CUS database.
CCB needs to keep track of any modification to the system.

Main Success Scenario:

1. IE documents the needed update
2. OE implements the updates in a test MIB
    . Retrieve the MIB from the HCSS database
    . Use the MIB editor to update the MIB access table.
    . Verify the updated MIB
    . Export the updated MIB from the MIB editor into ascii files.
3. ST tests the updated MIB on the instrument simulator.
    . Produce a test report
    . Test the updated MIB with SCOS2000
    . Produce a test report
    . Test the updated MIB in an HCSS sandbox
4. CC decides the new MIB is to be the operations MIB
5. CC delivers the updated MIB to the HCSS

Extensions:

3a. We may wish to test the new MIB on the instrument or instrument spare.

3b. Test, validation or verification fails:
    3b1. IH: distribute test report to all interested parties
    3b2. ICC: re-analyze problem report.

4a. CCB rejects the new MIB:
    4a1. ICC: re-analyze problem report.

References:
UCF-756 "Ingest MIB into HCSS"

Open Issues:
There will actually be several copies of the MIB for each phase of the mission. The configuration control will be handled by the HCSS but the actual scheme to do this is TBD.

Comments:
Use case UCF-756 deals with the ingestion of the MIB into the HCSS.
We assume the present use case results from a problem analysis that has concluded that the MIB needs to be updated.
At the testing stage the IT may have to wait for other changes to happen in other systems of the ICC (e.g. the CUS database). It is assumed that coordination of these changes is done outside of this use-case, possibly in the problem analysis use-case.
The timing of the delivery may have to be worked-out in conjunction with other deliveries. In principle, this should occur at the problem analysis level again.

Related work-packages

- Training in using the MIB editor
- Training in using SCOS2000
- Problem report/SCR system
- Provide instrument test environment and instrument simulator
- ICC configuration control system
- Information handling
- Create the MIB database

UC-ENG101: Simulate instrument behaviour

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **19 June 02**

Brief description:
This use case describes how an instrument simulator(s) is used to model the behavior of the instrument in order to (1) make sure that the command sequence is safe or (2) produce synthetic data. The data can be used for producing and testing QLA/IA, for validating a new observing procedure, or for investigating the consequences of a modification in the instrument properties. Data output should have the right telemetry format, and the house-keeping data should be complete. The simulator has to be able to execute commands that have been flagged as dangerous for the instrument or the satellite, and to produce the resulting output and indicate the consequences for the instrument and satellite.

Phase:
ILT-> Operations

Actors:
SSD: Scientific software developer
IT: Instrument tester
IH: Information handler

Triggers:
- A new script (see definition in UC-AIV102) has been generated and needs to be verified and validated.
- An observing mode needs to be tested and validated requiring a complete simulation (see definition in UC-CUS001).
- The consequences of a modification in the instrument capabilities need to be investigated.
- During tests, one wants to compare the result of a real test sequence, to the result of that same sequence performed on the simulator, for double-checking purposes.

Preconditions:
The MIB is available.
Mnemonics and commands sequences for the instrument have been defined.

Minimal post-conditions:
Any instrument-endangering command sequence would be detected.

Success post-conditions:

The script is verified (i.e. none of its command sequences or combination of command sequences results in the instrument leaving its safety zone).
Commands conflicting with the satellite capabilities are flagged.
A stream of data is produced that has exactly the same characteristics in term of formats as real data, so that any system that is able to ingest and treat real data can ingest and treat these simulated data.

Stakeholders and interests:
Test Scientist: need to know whether new scripts can be implemented on the instrument.
Problem Analyst: can use the system to investigate a problem report with the instrument.

Main Success Scenario:
1. IT: Receives the description of a test to be performed on the instrument simulator.
2. IT: writes the test script.
3. IT: Executes the script on the simulator
4. IT: Validates the technical quality of the output data
5. IT: Writes a validation report on test execution.

Extensions:
1a IT: determines that the description cannot be turned into a test script.
    1a1 IT: writes report containing the reasons of this non-feasibility
    1a2 IH: forwards report to interested parties.

4a the test produces incomplete data (including no data at all in case of a crash)
    4a1 IT: Determines cause(s) of the problem.
    4a2 IT: Files a problem report.

References:
UC-CUS001 – Test and validate observing modes
UC-AIV102 – Generate, validate and verify scripts and observation requests

Open Issues:

Comments:
Since we have to possibility to ingest packets (for instance with the QLA), we, the ICC can live with the packets produced by the instrument simulator. However to be sure that a procedure is safe, we also have to check that the satellite will be able to ingest the packets. Therefore we will probably have to access a satellite simulator for safety checking.
At step 3 of the Main Success Scenario, the simulator can include a satellite simulator as well.

This simulator is specifically a detailed model of the instrument hardware behavior and the telemetry it produces (excluding the science data). This distinguishes it from other possible forms for an instrument simulator, e.g a time estimator tool.

There are potentially four simulators in development:

- Instrument simulator which plugs into MOC s/c simulator (Stockholm)
- Cold parts simulator for the FPU (Saclay)
- Detector Readout and Control Unit Simulator, generates HK and some science TM. Used for testing interface to s/c (Stockholm)
- Simple telemetry simulator for feeding packets into SCOS2000 (ICC)

Any simulator used will need to be maintained at the ICC. This is not true of the spacecraft simulator, which is under the responsibility of the EGSE group.

Related work packages:

Access instrument parameters	This is to make sure that the data stream produced simulates the instrument in its current state.
Access telescope parameters	This is to make sure that the data stream produced simulates the telescope in its current state.
Define/Access telemetry format	Since the idea is to run the data stream through the analysis chain, the format of the telemetry has to be known so that the simulator can output data in this format.
Make Real Time Assessment.	To be able to validate the content of the housekeeping.
Make Quick-Look Analysis	Used in case only basic diagnostics are needed.
Define nominal instrument test	Since we aim at diagnosing problems or checking performances, the nominal state of the instrument should be defined to serve as a benchmark
Define instrument AOT	We need to know them to produce relevant data, but we may also modify them depending on the result of the Use-Case.
Maintain the instrument simulator	This is the blood of the use-case.
Make the instrument simulator	This is the heart of the use-case.
Make the information handler system	For transferring reports
Make the problem report system	Use in case of failure.
Store data for future analysis	Some simulations may be of long lasting interest, or may require too much time to analyze immediately.

UC-HSC101: Train personnel

Level: **user**
Scope: **SPIRE ICC**
Version: **1.2**
Status: **issue**
Date: **12th Jan 2005**

Brief description:
This use case covers the need for ICC actors to receive or give training. Such training could cover, for example, the use of software tools, primarily the Java, Jython and Python languages, in order to efficiently develop ICC software systems, e.g. IA.

Phase:
All phases up to Operations and to a lesser extent during Operations.

Actors:
SSD: Scientific software developer
ST: Software tester
SPA: Software product analyst
IM: ICC manager

Triggers:
A training need is identified.

Preconditions:
Sufficient funds are available in the training budget, if required.

Minimal post-conditions:
The training exercise happens. This is not necessarily a formal course – it could also be learning by experience, by mentor, or by books and articles.

Success post-conditions:
Something is actually learnt and the resulting skills used.

Stakeholders and interests:
The development team: who have deliveries to make.
ESA: the immediate customer for HCSS development.
ICC: the customer for development outside of the HCSS.
Astronomical community: the ultimate customer for the IA.

Main Success Scenario:
1. SSD/ST/SPA: identify training need.
2. IM: Agree appropriate action with SSD/ST/SPA.
3. SSD/ST/SPA: carry out the agreed action.

Extensions:

References:
UR-FSC2.2.1
SIRD-ICCO-05
SIRD-ICCO-080

Open Issues:

Comments:
Some of the areas where training may be needed include:

HCSS systems (CUS)
External system (SCOS2000, IDL, OBS)
Programming language, O/S, database
Languages (French, Spanish, Japanese, English, Welsh)

UC-ICC102: Plan and deliver a new developer release

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **9 August 2001**

Brief description:
This describes the course of actions followed in the ICC to plan and deliver a new developer release of a system. A developer release means a release of a version of the system that will only be available to developer sites, and, on demand, to consortium site. This is different from a use release, which is distributed to any outside users of SPIRE data through the HSC.

Phase:
Ground-segment test phase and onward.

Actors:
CC: configuration controller
SSD: Scientific software developer
ST: Software tester
IH: Information handler

Triggers:
The CCB decides a new release date (e.g. in case of a new users' release).

Preconditions:
A configuration control system is available.
Changes have been made to elements in CC.

Minimal post-conditions:
It is possible to revert to the previous release (developer or user).

Success post-conditions:
A new developers' release is available.
Changes with respect to the previous version are documented.
Software validation has occurred.

Stakeholders and interests:
ICC: wants to benefit from the latest improvements as soon as possible.
ICC CCB wants to keep software development under control.

Main Success Scenario:
1. CC: Issues a time table for next release taking into account the users' release development plan.
2. SSD: Creates or updates software artifact(s) within the ICC (UC-ICC101).
3. CC: Freezes development activities.

4. CC: Pre-releases the developers' release.
5. ST: Validates and verifies the developers' release.
6. CC: Releases the new developers' release.
7. CC: Documents the new developers' release.
8. IH: Delivers that report to all interested parties.
9. CC: Allows development activities to resume.

Extensions:
2a: Software artefact is not created/updated.
2a1. CC: goes back to step 1.

5a: Validation/verification of new developer's release fails.
5a1. ST: documents failed validation/verification.
5a2. IH: delivers report to all interested parties.
5a3. CC: abort the process and revert to previous developers' release
5a4. CC: Allows development activities to resume.

6a: Delivery of the new developers' release fails.
6a1. CC: issues a report on the failed delivery.
6a2. IH: delivers report to all interested parties.
6a3. CC: abort the process and revert to previous developers' release
6a4. CC: Allows development activities to resume.

References:
UC-ICC101: Create or update a new software artifact.
UC-ICC003: Plan and deliver a new users release

Open Issues:
None

Comments:
The formal decision procedure to release includes a possibility for the CCB to make a new release if it is urgently needed.

Related work packages:
- Provide ICC Configuration Control system.
- A system for recording decisions/actions (by ICC, consortium…) is available.

UC-ICC103: Produce a new test environment

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7 December 2001**

Brief description:
This use-case describes how a new test environment is created and delivered to the ICC. The test environment is described in the Definitions document as: a software environment common to all ICC members, where changes to the versions of the system are agreed and monitored. Inside the test environment, it is impossible to modify operational systems (e.g. databases).

Phase:
All phases including possibly post-operations

Actors:
CC: ICC configuration controller
IH: Information handler

Triggers:
It is time to update the test environment (i.e. the current one has become obsolete with respect to current versions of the system elements, or it is that day of the month/year).

Preconditions:
Configuration control for all ICC subsystems exists.

Minimal post-conditions:
The system elements in configuration control are not corrupted, the old test environment can always be brought on-line immediately (i.e. we are never in a situation where no test environment is available).

Success post-conditions:
The new test environment is on-line, users of the system are informed of all the modifications between the old and the new environment.

Stakeholders and interests:
ICC Manager: needs an efficient but controlled way of system development.
Software Developer: needs to be able to benefit from the latest upgrades of the system.

Main Success Scenario:
1. CC: Validates that all elements of the test environment system can be delivered.
2. CC: Assembles a report containing all changes to the test environment.

3. CC: identifies the current versions of the system elements as being part of a released test environment
4. CC: Configure the system so that the selected version of the system elements is now the default one
5. CC: Notifies ICC users of the change
6. IH: distributes delivery report to all interested parties

Extensions:

1a: Not all the elements of the test environment system can be delivered
- 1a1. CC: validates that all outstanding problem reports are being addressed
- 1a2. CC: produces a report explaining the reasons for the cancellation of the delivery
- 1a3. IH: distributes this report to all interested parties

1a1a: One or more problem reports are not addressed
- 1a1a1. CC: assign ICC manpower to outstanding problem
- 1a1a2. CC: Produces a report explaining the reasons for the cancellation of the delivery
- 1a1a3. IH: distributes this report to all interested parties

4a: The system cannot be configured to use the new versions of its elements
- 4a1. CC: issues a problem report
- 4a2. CC: Reverts to previous version of test environment
- 4a3. CC: produces a report explaining the reasons for the cancellation of the delivery
- 4a4. IH: distributes this report to all interested parties

References:

UC-ICC003: Plan and deliver a new user release
UC-ICC101: Create or update a software artefact (within the ICC)
UC-ICC102: plan and deliver a new developer release

Open Issues:

I still need to make sure that this use case is really different from UC-ICC003 and UC-ICC102 (plan and deliver a new user/developer release). It should be as the test environment is a larger system than the one being considered in UC-ICC003/101/102.

Comments:

In the MSS, I have implicitly assumed that the generation and release of a new test environment does not affect the previous test environment, i.e. that ICC member can still work in the old environment while the new one is being delivered. If that is not the case, then an additional step has to be included where the system is frozen and everyone logged out of it while the new environment is being delivered.

Related work packages:

- Define/Create/Manage the configuration control
- Define/Create/Manage the information handling system

- Define test environment

UC-ICC104: Create or update a document

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **12TH Jan 2005**

Brief description:
This use case describes how a formal project document is created or updated. This involves obtaining a document code, creating or retrieving an appropriate template and checking the document in/out of a configuration control system.

Phase:
All phases

Actors:
DOC: Documenter, an ICC member.
IH: Information Handler

Triggers:
There is a need to create or update a document.

Preconditions:
A document configuration control system (DCCS) exists.

Minimal post-conditions:
The integrity of the DCCS is maintained and the original copy of the document is retained.

Success post-conditions:
The document is created or modified.
The created or modified document is entered into the DCCS and is assigned a new version number.

Stakeholders and interests:
DOC: author requires the DCCS to supply the most recent version of the document and prevent simultaneous modification.
CC: Configuration Controller wants to track the modification history and be able to recover any previous state of the documentation.
ICC: requires the ICC documentation to be accurate and functional.

Main Success Scenario:
1. DOC: requests modification of a document
2. CC: Assesses the extent of modification of the document to decide the scope of the review.
3. CC: retrieves the document from the DCCS and locks the document so nobody else can modify it and delivers it to DOC.

4. DOC: Writes or modifies the document and returns it to CC.
5. MRB: reviews the document
6. DOC: Enters the document into the DCCS.
7. IH: Informs all interested parties of the update.

Extensions:
1a: The document is new
    1a1. CC: provides a unique project document reference.
    1a2. CC: Go to 2

3a: The document is already locked (i.e. being modified by another party)
    3a1. CC: informs DOC that the document is being modified and by whom.

5a: The modified document is not acceptable
    5a1. MRB: Informs DCC of reasons for unsuitability
    5a2.CC: Informs DOC of reasons for unsuitability
    5a3. DOC: Go to 4

References:
UR-ICC3.2.1-2
UR-AOP5.5
SIRD-ICCF-190

Open Issues:
It is not clear how dependencies between documents are decided and tracked. (see comments)

Comments:

When the document is locked (Step 2 MSS) other documents with explicit dependencies are also locked.

DOC: Documenter can be Software Developer Designer, ICC-member, ICC manager

Java source code documentation will at least **partly** involve the use of JavaDoc generating HTML files. As well as documents such as technical description and user manuals, the documentation of software will also include the source code and JavaDoc files. *The appropriate Use Case for source code documentation is UC-ICC101.*

The current implementation of a document configuration control system is Livelink. Not all documents though will go into Livelink, there will be an internal ICC repository similar to the ftp we have at the moment.

UC-ICC110: Maintain ICC web page

Level: **user**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **12 Jan 2005**

Brief description:
This use case describes how readily accessible information on the status, functioning, plans, etc of the ICC is created/updated via a web site.

Phase:
Pre-ILT -> post-operation

Actors:
DOC: Documenter
IH: Information handler

Triggers:
New information or functionality is available.

Preconditions:
A web server and any required plug-ins are installed and started.

Minimal post-conditions:
- The security of the site computers and networks is not compromised.
- The web server is still running.

Success post-conditions:
The new information or functionality is available on the Web.

Stakeholders and interests:
The Web master wants to ensure trouble-free web server
IM wants to ensure information distribution over the web
HSC-Helpdesk wants to ensure information distribution over the web
General astronomer wants easy access to up to date information and tools
PI wants to ensure SPIRE has good public profile

Main Success Scenario:
1. DOC: deliver item(s) to IH.
2. IH: place item(s) in the correct place on the server.

Extensions:
1a: IH: configure server to automatically pick up item(s) from DOC.
1a1. IH: exit

2a: IH: update all the files that need to point to the item.

References:
UR-ICC060
UR-OTH4.5-6
UR-PUB3.2.2

Open Issues:
- There are currently a number of ICC related web sites. Should a single site be used? It is still possible that the system could be distributed over multiple sites even if there is only a single point of access (portal).
- Use case needed for issuing usernames and passwords?

Multiple we sites are still being used but we are working towards a portal system unifying the sites.

The SPIRE BBS is functioning as a good way of distributing information within the ICC.

No use case for issuing usernames and passwords still?

Comments:
Both public and non-public areas will be needed.
Password protection needed for non-public areas

Related packages:
- Design and implement web site
- Manage web site

UC-OBS101: Validate and verify OBS

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th November 2001**

Brief description:
This use case describes how a new or updated on-board software (OBS) image is tested to ensure that it:

- does not compromise the safety of the instrument.
- does not adversely affect the performance of the observing and operating modes.
- successfully performs the task that it was intended for.

Phase:
ILT, IST, PV, Operations

Actors:
ST: Software tester
IT: Instrument tester
IH: Information handler

Triggers:
A new OBS image is available for testing.

Preconditions:
The on-board software maintenance facility (OBSMF) exists at the ICC.

Minimal post-conditions:
The new OBS image is not corrupted, overwritten or lost.
Existing OBS images are also not corrupted or lost.
A test report is produced.

Success post-conditions:
The new OBS image is validated and verified, i.e. it passes all the tests outlined in the description above.

Stakeholders and interests:
Instrument engineer wants to ensure that the new OBS image does not damage the instrument.
Instrument scientist wants to ensure that the instrument is being optimally operated.
Project scientist wants to ensure that the S/C and other instruments are not damaged.

Main Success Scenario:

1. IT: Get the new OBS image from the OBSMT (including the OBSMT's own test report).
2. ST: Run a set of software checks to ensure that the image integrity is maintained and that only the expected parts of the image have been updated.
3. IT: Run a set of tests on the instrument simulator to ensure that the OBS behaves in the expected way following the update and that it does not compromise instrument or spacecraft safety.
4. IT: Run a set of tests on the instrument itself to ensure that the OBS behaves in the expected way following the update and that it does not compromise instrument or spacecraft safety.
5. IT: Produce a test report
6. IH: circulate report to all interested parties.

Extensions:

1a: The image cannot be retrieved.
- 1a1: ST: Produce report
- 1a2: IH: Circulate report to all interested parties

2a: The image is corrupt.
- 2a1: ST: Produce report
- 1a2: IH: Circulate report to all interested parties

3a: Test fails
- 3a1: ST/IT: The test is aborted and a report produced.
- 3a2: IH: Circulate report to all interested parties.

References:

UC-OBS102

Open Issues:

It is not clear how the new OBS image is retrieved for this use case. Does the OBSMT put it in a test area of the HCSS, or does it simply make it available to a local ICC database (outside of the HCSS)? If the latter is true then some form of configuration control (internal to the ICC) will be necessary.

The OBSMF should include simulation of the effects on other instruments and the S/C so that the effects of any SPIRE OBS on these can be assessed. It is entirely possible that a SPIRE OBS will function perfectly within SPIRE, but may cause problems for the rest of the S/C.

Comments:

It is expected that the OBSMF will be used by the OBSMT to generate new or updated OBS images. The validation and verification tests described here are distinct from the ones that the OBSMT carries out internally prior to making an image available for uplinking to the instrument.

In ILT, step 3 of the success scenario may be done on the instrument itself.

In the operations phase, step 4 of the main success scenario may be done on the flight spare as well as on the instrument simulator.

During operations step 4 of the MSS involves the HSC as the ICC cannot uplink to the actual instrument.

Sandbox environment is used for MSS steps 1,2 and 3.

Work-packages:

- Set up OBSMF within the ICC
- Provide OBS test system
- Establish Problem report/SCR system
- Provide Instrument simulator
- Provide ICC configuration control system
- Provide Information Handler

UC-OBS102: Generate new OBS

Level: **user**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **18 July 2001**

Brief description:
This use case describes how the SPIRE ICC generates and tests a new OBS image in response to an SCR, approves it, and delivers it to the HCSS for uplink to the instrument.

Phase:
ILT, IST, PV, Operations

Actors:
IH: Information handler
CC: Configuration control
MRB: Material review board
IE: Instrument Engineer
IT: Instrument Tester
OBSMT: On-Board Software Maintenance Team

Triggers:
The ICC receives notification of an On-Board Software Change Request.

Preconditions:
The OBS maintenance facility exists (UR-OBS-110) as part of the ICC.
A problem has been investigated, and a solution, which requires a change of the OBS, has been found, leading to an SCR.

Minimal post-conditions:
A decision has been made by the MRB regarding the SCR.

Success post-conditions:
The OBS is updated.

Stakeholders and interests:
Problem reporter wants to see the problem solved.
ICC scientist wants the optimal instrument performance.

Main Success Scenario:
1. IH: retrieve the SCR on the OBS (use equivalent of UCF-362 Retrieve an SCR).
2. IH: The IH pass the relevant information to the OBSMT.
3. OBSMT: generate a new OBS image.

4. IT: take this new OBS image and test it and produces a report (UC-OBS101).
5. MRB: approve the OBS change.
6. IH: report the outcome of this decision to all interested parties.
7. IE/CC: update the OBS (UC-AIV101).

Extensions:
5a. MRB: reject the change.
5a1. IH: report the outcome of this decision to all interested parties.

References:
UC-OBS101 – Test and validate the OBS
UC-AIV101 – Update the OBS

Open Issues:
The exact location of the OBSMT is TBD. This impacts information exchange in user-level use-cases.
How does the ICC receive a request for a change i.e. do we have an internal SPIRE ICC SPR and SCR procedure?
The mechanism for transferring a change request to all interested parties is still TBD.

Comments:
In this use case, a Software Change Request is both a reason for a change and a description its implementation. By definition, at this stage, it is approved by the ICC. When we only have the description of a problem, it is called a problem report. These definitions will have to be tied in with the mechanisms outlined in the HCSS do deal with problem reports and SCRs.
It is assumed that the interested parties are listed as part of the Problem Report or the SCR, so that they are automatically aware of any change, as well as of the outcome of the PR/SCR.

Work-packages:
- OBS maintenance system
- OBS test system
- Problem Report/SCR system

Sub-function use-cases

UC-AIV103: Access data storage

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 02**

Brief description:
This use case describes how data are retrieved and stored during early ILT. This involves a local AIV database that may not be connected to the local node of the HCSS.

Phase:
ILT

Actors:
IT: Instrument tester
QLA: Quick look analysis software

Triggers:
- An instrument test is to be performed.
- Data from an instrument test is to be played back.

Preconditions:
- A local AIV database is available and accessible.
- The local AIV database can ingest telemetry.
- At least a minimal QLA is available.
- The EGSE packet router is available.

Minimal post-conditions:
- No database is corrupted.
- No telemetry/telecommand data is lost.

Success post-conditions:
The data is successfully read or written.

Stakeholders and interests:
IS, IE, TS, IT, AIV manager, EGSE manager, DM.

Main Success Scenario:
1. IT/TS: Starts a test or starts replay of a test.
2. QLA: Accesses the storage system.
3. TM ingestor/QLA: Stores (test) or retrieves (replay) the data.
4. DM: Replicate the data in the main database ie not the one in the AIV setup.

Extensions:

1a. IT/TS: Extracts data based on database query (i.e. more flexible than on a test basis).

References:

Open Issues:

Comments:
Step 4 is not always performed. It is done at well-defined periods, and is essentially an “offline” operation. This step is only necessary if a separate, isolated, AIV database is used.

If software has been changed during the performance of tests, this will have to be reintegrated within the main ICC system.

Database replication is simpler if the AIV database is connected to the network. This could be during periods when tests are not taking place.

The reason this Use Case exists is an AIV requirement for network isolation during tests.

Software to ingest and replay telemetry is part of the HCSS.

Related Work Packages

Infrastructure	Hardware procurement System management Database management
QLA	Requirements definition Analysis and prototyping Implementation Maintenance

UC-AOP102: Estimate observation time

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 2002**

Brief description:
This use case describes how the Time Estimator tool is used to get a prediction of the total observing time required for a given observation. The use case includes the steps involved in obtaining an observation as input to the Time Estimator, and what is done with the result (e.g. the observation proposal is flagged as 'valid' with respect to time constraints).
This is in fact an HCSS use case, but it does not exist yet.

Phase:
Call for guaranteed time proposals, onwards.

Actors:
CS: Calibration Scientist

Triggers:
A science or calibration goal has been set, an observing/operational mode has been selected and the time requested to perform this observation needs to be estimated.
An observer has been allocated a given time to perform an observation, and one needs to check whether the selected set-up does not exceed the allocated time.

Preconditions:
A time estimator exists.

Minimal post-conditions:
An estimate of the time requested to perform a given sequence is produced. An upper limit is acceptable but not a lower limit.

Success post-conditions:
The time requested to reach the goal is computed with less than 10% error. Achieved signal to noise ratio on the target are computed. Efficiency of the observation (ratio of on-source to total observing time) is computed.

Stakeholders and interests:
Proposal Handler: can check whether a proposal fits in its allocated time
Astronomer: will use the tool to define the feasibility of a science goal.
FOTAC: will probably request that some standard observations be budgeted so that they know whether a proposal is realistic in its time request.

Main Success Scenario:
1. CS: Defines the target parameters

2. CS: Defines the goal to reach (S/N, mapped area…)
3. CS: Obtain the time requested to reach the goal.

Extensions:
none

References:
UR-AOP3.1.1-7, UR-AOP4.1-4.3

Open Issues:
- Who is the correct actor?
- Do we separate this use case in two according to the two possible levels (i.e. costing an observing sequence, or estimating how much time is needed to reach a given goal)?

Comments:
This use case's scope is the ICC so only covers the use of the time estimator within the ICC and not, for example, by a version distributed to the Astronomer. In that case, the only actor really worth considering here is the calibration scientist.
Note that this use case has two levels: (1) one can use a time estimator to compute how much time a given observing sequence will request, and (2) One can use a time estimator to estimate how much observing time is needed to reach a given goal.
This can be resolved in the following way: for a given observing configuration (a set of AOT parameters) and a given source configuration (point/extended, brightness level, background level), the tool would provide the requested time and achieved S/N. This mode of operation will change the Main Success Scenario.

Related work packages:
Main work package: write time estimator.

UC-CUS101: View observation schedule

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 02**

Brief description:
This use case describes how the observation schedules for operational days generated by the CUS, in the HCSS system, are viewed at the ICC.

Phase:
Operations

Actors:
CS: Calibration scientist
IE: Instrument engineer

Triggers:
The need to see what has been scheduled on a particular day.
One is searching for certain type of observations in the schedules

Preconditions:
A schedule exists in the HCSS and is accessible by the ICC.

Minimal post-conditions:
The schedule is not changed or locked by the viewing software.

Success post-conditions:
The schedule is viewed and the actor is at peace.

Stakeholders and interests:
CS & IE: need to make sure the observation they submit are correctly scheduled.

Main Success Scenario:
1. CS & IE: start the access software.
2. CS & IE: Search the list of schedules on a list of user-specified criteria.
3. CS & IE: display the resulting schedule(s).

Extensions:

References:

Open Issues:
What do we do if we don't like what we see and we need to get the day re-planned? This should be addressed in the ground segment Interface Requirement Document, because it cannot be resolved only inside the ICC.

One also needs to ensure that the ICC will be able to retrieve Herschel observations spotted with this viewing facility, even if they don't belong to the SPIRE consortium, when they can be used for calibration purposes. Again this is an interface issue.

Comments:
The mission timeline is a time-keyed sequence of telecommands sent to the satellite. A schedule is a timed sequence of schedulable units, a unit being either an observation or a slew.

The software for this purpose is expected to be accessed via a web browser.

The main reason for this use-case is that we will need to inspect the observation schedule: the ICC will prepare observations (e.g. calibrations) but it is the responsibility of the HSC to produce the observation schedules. We will have to be able to inspect these schedules to know when our calibration observations will be done. We will also have to inspect these schedules to identify general observations that could be used for calibration purposes.

What we want to view is a list where observations are easily recognisable (i.e. no a list of all the individual commands sent to the instrument), with sufficient time information to locate the observation within the operational day.

The user-specified criteria are:
- the operational day
- the type of observation
- the instrument
- the object
- the observer
- …

Related work-packages:
- Create and maintain the viewing software (this is a common work-package).
- Inspect the schedules with the viewing software.

UC-DAC101: Identify and flag bad data

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **Issue**
Date: **18 June 02**

Brief description:
Some of the photometer data will not be suitable for science and/or will spoil the data products. This may be due to problems with or characteristics of the instrument or the conditions under which the data were taken. Such data are flagged as bad by the S/W using HK information and/or the data themselves using criteria, which can be specified by the user.

Phase:
ILT -> post-operation

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The user wants to identify and flag bad data.

Preconditions:
The photometer and HK data exist and have been retrieved from the DB.

Minimal post-conditions:
The IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
Data that meet the criteria are identified and flagged.
Data reduction history is updated.

Stakeholders and interests:
AST wants to identify all bad data in their observation.
IE wants to remove different types of bad data to investigate the data parameter space as part of instrument testing.
ICC Scientist needs to know what proportion of the data are being flagged as bad for determining the efficiency of the instrument.

Main Success Scenario:
1. DP: Interacts with IA
   1.1 DP: Examines observation and data reduction history (UC-DAS104)
   1.2 DP: Determines the criteria on which the data should be flagged – these

might be the default criteria (TBD) or user-specified (UC-DAC105).
1.3 DP: Runs “Identify and flag bad data” S/W specifying data and criteria.
1.4 IA: Compares specified data against specified criteria and flags bad data.
1.5 IA: Outputs flags (either incorporated with the data or as a separate output)
1.6 IA: Reports on quantities (relative and absolute) of data flagged against each criterion (UC-DAS5, UC-DAS107)

2. DP: Inspects results.

Extensions:

1a DP: Bad data already flagged and/or removed, stops process.

2a DP: Results are unacceptable.
2a1 DP: Returns to step 1.2.

References:

UC-DAC105
UC-DAS104
UC-DAS105
UC-DAS107
UC-CAL105

Open Issues:

None.

Comments:

This is a specific case of UC-DAC105.
The use-case for defining default bad data criteria is UC-CAL105.

Related work packages:

Write “Flag and identify bad data” software artefact
Define default bad data criteria.
Write history software.
Create documentation/help system

UC-DAC102: Filter data on any criteria

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **Issue**
Date: **18 June 02**

Brief description:
This use case describes how certain data are excluded according to specified criteria. Data which are not excluded remain unchanged.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
A subset of the data is required.

Preconditions:
Photometer data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The photometer data are filtered according to the specified criteria.
Data reduction history is updated.

Stakeholders and interests:
AST wants to filter data from their observation.
IE wants to filter data to investigate the data parameter space as part of instrument testing.

Main Success Scenario:

1. DP: Interacts with IA
   1.1 DP: Examines data observation and reduction history (UC-DAS104)
   1.2 DP: Decides criteria on which to filter data (see comments).
   1.3 DP: Runs “Filter data” S/W specifying data and criteria.
   1.4 IA: Compares specified data against specified criteria and filters out matching data.
   1.5 IA: Outputs filtered data.

1.6 IA: Reports on quantities (relative and absolute) of data filtered against each criterion (UC-DAS105, UC-DAS107)

2. DP: Inspects results.

Extensions:

2a DP: Results are unacceptable.

2a1 DP: Returns to step 1.2.

References:

UC-DAC101
UC-DAC105
UC-DAS104
UC-DAS105
UC-DAS107

Open Issues:

Need input on filter criteria from someone with knowledge of ILT.

Comments:

The criteria that we may want to flag data on should be the same as the criteria we want to filter data on. A list of criteria that we may want to filter on are:

- Flagged bad data (UC-DAC101)
- Flagged data (UC-DAC105)
- Statistical filtering, e.g. sigma clipping, median filtering
- Time slice of the data
- Bolometer number
- Sky position
- Any more?

Related work packages:

Write software to filter on specified criteria (see comments)
Write the history software
Create the documentation/help system

UC-DAC103: Remove effects of instrument cross-talk

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **18 June 02**

Brief description:
This use case describes how the effects of instrument cross-talk are removed.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The effects of cross-talk need to be removed from the data.

Preconditions:
Photometer data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The effects of cross talk are removed.

Stakeholders and interests:
AST/IE/IT/CS need to know how much cross talk is taking place and remove its effects.

Main Success Scenario:

1. DP: Interacts with IA
1.1. DP: Runs “Remove effects of instrumental cross-talk” S/W.
1.2. A: Determines the effects of cross-talk, and reports this to the user
1.3. IA: Effects of instrument cross-talk are removed.
2. DP: Inspects results.

Extensions:
1.1a. IA: Effects of cross-talk already removed. Stops process.

References:

Open Issues:
The level of cross-talk is unknown. This usecase and workpackages may disappear.

Comments:

Related work packages:

- Write software to remove effects of instrumental cross-talk
- Create documentation/help system
- Design algorithm(s) to characterize and remove effects of instrumental cross-talk

UC-DAC104: Remove pixel-to-pixel sensitivity variations (flat-field)

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6 September 2001**

Brief description:
The data have to be corrected for pixel to pixel sensitivity variations, using a flat-field artifact.

Phase:
ILT -> post-operation

Actors:
DP: Data Processor
IA: Interactive Analysis

Triggers:
The user wants to flat-field the data.

Preconditions:
Photometer data exist and have been extracted from the database.
Flat-field artifact* exists and is available.
IA exists and includes the flat-fielding software.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
Pixel to pixel sensitivity variation is corrected.
Data reduction history is updated.

Stakeholders and interests:
AST/IE/IT/CS want to correct pixel to pixel sensitivity variation.

Main Success Scenario:
1. DP: interacts with IA.
   1.1 DP: runs "Flat Field correction".
   1.2 IA: flat-fields the data.
   1.3 IA: modifies the data reduction history (UC-DAS107).

Extensions:

1.1a: IA: determines that data are already flat-fielded and then stops the process (UC-DAS105).

References:
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
Any operation that modifies the data has two modes: creating a new artifact or overwriting the old one.
*The form of the flat-field artifact has to be specified.

Related work packages:
- Write the flat-fielding software.
- Write the history software.
- Create the documentation/help system.

UC-DAC105: Identify and flag data on any criteria

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6 September 2001**

Brief description:
This use case describes how some data are identified and flagged with logical/mathematical expression using temporal, spatial and intensity coordinates (e.g. for sky background identification).

Phase:
ILT -> post-operation

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
The user wants to identify and flag a subset of the data.

Preconditions:
Photometer data exist and have been extracted from the DB.
IA exists and includes the flag data software.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The relevant data are flagged.
Data reduction history is updated.

Stakeholders and interests:
AST/IE/IT/CS want to flag a subset of the data for further process.

Main Success Scenario:

1. DP: interacts with IA.
   - 1.1 DP: chooses a way to identify the subset (may include parameters and visualization: UC-DAS108).
   - 1.2 IA: determines the data that satisfy the selection criteria.
   - 1.3 IA: flags these data.
   - 1.4 IA: reports the quantity of data flagged (relative and absolute). (UC-DAS105, UC-DAS107)

Extensions:

References:
UC-DAS108: Visualize any data product
UC-DAC105: Information and error messaging system.
UC-DAC107: Record data reduction history.

Open Issues:

Comments:
Examples of selection criteria:

- User-specified bolometer numbers.
- Outside or inside a user-defined region (spatial and/or temporal).
- Below a user-specified intensity level.
- Fitting trends to the data (e.g. gradients).
- Statistical definition (e.g. 2 sigma around median).
- Logical/mathematical expression using temporal, spatial and intensity coordinates.

The data is not necessarily an image.
Any operation that modifies the data has two modes: creating a new artifact or overwriting the old one.

Related work packages:
Write the data subset identification/flag software.
Write the history software.
Create the documentation/help system.

UC-DAC106: Subtract a background

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6th December 2001**

Brief description:
This usecase describes the removal of background signal (spacecraft, telescope, 'sky') from science and calibration data by user specified criteria. The removal of background signal can be performed interactively or automatically as part of a pipeline.

Phase:
ILT – Post-operations

Actors:
DP, IA

Triggers:
Data has been reduced to a degree where the subtraction of background signal is required.

Preconditions:
Photometer data exist and have been extracted from the DB.
IA exists and includes the flag data software.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The correctly identified background signal is removed from the science or calibration signal.
Data reduction history is updated.

Stakeholders and interests:
AST/IE/IT/CS want to remove the background signal data to obtain the true source flux.

Main Success Scenario:
1. DP: selects a method for background subtraction
2. DP: interacts with IA.
   2.1 IA: determines the background
   2.2 IA: reports the quantity of signal. (UC-DAS105, UC-DAS107)
   2.3 IA: removes the background signal

2.4 IA: modifies data reduction history (UC-DAS107)

Extensions:
1a method requires the identification and flagging of background data
1a1 DP: flags data for use of background determination (UC-DAC105)
1a2 DP: goto step 2

References:
UC-DAC105: Identify and flag data on any criteria.
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
Examples of selection criteria:
- User-specified bolometer numbers.
- Outside or inside a user-defined region (spatial and/or temporal).
- Fitting trends to the data (e.g. gradients).
- Logical/mathematical expression using temporal, spatial and intensity coordinates.

This usecase is different from UC-PHT121 (Subtract off-source data), which deals with the demodulation of the signal.

Related work packages:
- Write IA
- Write background subtraction algorithms
- Define observing modes
- Write data reduction history recording system

UC-DAC107: Transform between sky and spacecraft coordinate systems

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6th December 2001**

Brief Description:
This usecase describes how data coordinates are transformed from those relating to the spacecraft to those relating to the sky, e.g., right ascension and declination. This is pointing reconstruction.

Phase:
ILT -> Post Operations

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
The user wants to know where SPIRE was pointing on the sky and/or wants the data in a format that can be re-sampled into a map (UC-PHT108).

Preconditions:
The photometer and attitude data exist, have been retrieved from the database and have been processed into a format suitable for IA (UC-DAS101).

Minimal post-conditions:
If the process fails, a clear and relevant error message is produced (UC-DAS105) and the original data are not modified.

Success post-conditions:
Required coordinates are appended to the data.
Data reduction history is modified (UC-DAS107).

Stakeholders and Interests:
IE/AST wants data coordinates in astronomically useful format.

Main Success Scenario:
1. DP: interacts with IA
   1.1 DP: Examines observation and data reduction history (UC-DAS104.)
   1.2 DP: Decides which coordinate system is required.
   1.3 DP: Runs S/W specifying required coordinate system.
   1.4 IA: Appends coordinates and description of coordinate system to data.
   1.5 IA: Modifies data reduction history (UC-DAS107)

Extensions:

1.2a: Required coordinate system is not supported.

1.2a1 DP: either issues a SCR from within the ICC or if external contacts the HSC helpdesk.

References:

UC-PHT108
UC-DAS104
UC-DAS105
UC-DAS107
UC-DAS101

Open Issues:

The conversion from pixel matrix coordinates to sky coordinates can be made to a number of reference system but then the subsequent processes would have to be able to understand all these systems.

Comments:

Pointing reconstruction is converting from the pixel matrix coordinates to astronomical coordinates.

Related work packages:

- Write data reduction history s/w
- Write coordinate conversion s/w artifact
- Write documentation/help system
- Write s/w to “prepare data frames for IA processing”

UC-DAC108: Convert from engineering units to astrophysical units.

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6 December, 2001**

Brief description:
This usecase describes the transformation of signal from engineering units (e.g. volts) into astrophysical units (e.g. Jy, Jy/beam)

Phase:
All phases

Actors:
DP, IA, CS

Triggers:
DP wants to convert data into astrophysical units.

Preconditions:
Photometer data exist and have been extracted from the DB.
IA exists.
Relevant calibration artifacts have been generated and are available to IA.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The data are correctly calibrated using the requested calibration artifacts.
Data reduction history is updated.

Stakeholders and interests:
AST, CS require a scientifically valid data-product.

Main Success Scenario:
1. DP: interacts with IA.
   1.1 DP: chooses the relevant calibration artifact
   1.2 IA: imports calibration artifact
   1.3 IA: applies the calibration artifact to the data
   1.4 IA: modifies data reduction history (UC-DAS107)

Extensions:
1.1a: IA provides the default calibration artifact.

References:
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
'Calibration artifact' covers all the information required by all arrays to be correctly calibrated.

Related work packages:
- Write software that enables IA to multiply the data by a given factor.
- Generate calibration tables (artifacts) for use by IA.

UC-DAS101: Read and Prepare Data-frames for IA processing

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief description:
This usecase describes the stages that are required to transform data-frame(s) into an IA object. Individual observations (calibration, science, pointing etc) will consist of many data-frames and these will require consolidation into objects within IA. This will be the first step of the reduction process in IA.

Phase:
All phases

Actors:
DP, CS, IA.

Triggers:
Data-frames require conversion into an object to be reduced and analysed by IA.

Preconditions:
IA has access to the database.

Minimal post-conditions:
IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data-frames are not modified.

Success post-conditions:
The data-frames have been read into IA and are ready for reduction and analysis
Data reduction history is updated.

Stakeholders and interests:
AST/CS

Main Success Scenario:
1. DP: interacts with IA.
   1.1. DP: chooses the observation
   1.2. IA: retrieves data-frames associated with the observation
   1.3. IA: runs conversion software on selected data-frames
   1.4. IA: reports the status of the new object (UC-DAS105, UC-DAS107)

Extensions:

References:
UC-DAS105, UC-DAS107

Open Issues:

Comments:
For the PHT the absolute timeline will have been reconstructed.

For the FTS the absolute timeline will not have been reconstructed within the data-frame.

Related work packages:

- Write database interface
- Define data-frames
- Define IA raw data object
- Define what pre-processing needs to take place.
- Write data-frame to IA-raw data object software.

UC-DAS102: Import non-IA format data.

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief description:
This use-case describes importing data into IA (or QLA).

Phase:
All phases

Actors:
DP, IA.

Triggers:
DP requires data to be imported from an external analysis/presentation package.

Preconditions:
IA exists.
IA supports the import data format.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The data are imported into the IA.

Stakeholders and interests:
AST, CS

Main Success Scenario:
1. DP: interacts with IA.
   1.1. DP: selects data to import into IA
   1.2. IA: imports the data into IA.

Extensions:
1.1a: The IA cannot tell what format the import data are in.
   1.1a1. IA: prompts DP for import format.

1.2a: The import format is not supported.
   1.2a1. IA: informs user. (UC-DAS105)

References:

UC-DAS105

Open Issues:

Comments:
Wish list for possible import formats:

XDF, FITS, ASCII, TIF, JPEG (may experience compression problems), GIF, PS, EPS, SVG (supported by JAI probably will become standard format), XML, PDF, PICT.

The data reduction history may be changed during the export/import.
The actual data values may be changed if they are exported and imported (rounding, precision problems)

Related work packages:

- Write import software.

UC-DAS103: Convert output data into popular data formats

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief description:
This use-case describes the conversion of objects within IA (or QLA) to popular data formats for importing into other astronomical analysis packages. The conversion can be performed at any stage of the data reduction after initial SPIRE-specific reduction steps.

Phase:
All phases

Actors:
DP, IA.

Triggers:
DP requires data to be exported to an external analysis/presentation package.

Preconditions:
Photometer data exist and have been extracted from the DB.
IA exists and includes the flag data software.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The data are converted into the requested data format.

Stakeholders and interests:
AST, CS

Main Success Scenario:
1. DP: interacts with IA.
   1.1. DP: chooses the data object and the required output data format
   1.2. IA: converts the object into the requested data format.
   1.3. IA: original data object is retained.

Extensions:

1.1a: format is incompatible with the data object
1.1a1. IA: provides a clear and relevant error message. (UC-DAS105)
1.1a2. IA: process stops

References:
UC-DAS105

Open Issues:

Comments:
There exists an HCSS usecase/technical note on the scope of the popular data formats being considered for support within IA. This usecase will probably be redundant if the IA is to be developed commonly by the three instruments.
The formats being considered (taken from Minutes of IA-commonality infrastructure meeting 31/10/01):
XDF, FITS, ASCII, TIF, JPEG (may experience compression problems), GIF, PS, EPS, SVG (supported by JAI probably will become standard format), XML, PDF

Not all output formats will be able to be imported back into SPIRE-IA.

Related work packages:
- Write conversion software.

UC-DAS104: Examine observation and data reduction history

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief Description:
This usecase describes how the user is able to obtain information relating to the observation itself (integration time, source name, observing mode, house keeping, spacecraft etc) and to its data reduction history, including the adopted software and calibration artifact versions.

Phase:
ILT -> Post Operations

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
The user needs to know what steps in the data reduction pipeline have already been applied to a given dataset, or wants to know details of the observation or status of the spacecraft during the observation.

Preconditions:
The data exist.

Minimal post-conditions:
IA doesn't crash.
If the process fails, a clear and relevant error message is produced (UC-DAS105).

Success post-conditions:
The observation and data reduction history are printed to the screen and/or to a file.

Stakeholders and Interests:
AST/IE/CS/IS requires information on observation and data reduction history

Main Success Scenario:

1. DP: Interacts with IA.
   1.1. DP: Specifies information required.
   1.2. IA: Writes history information to screen and provides option to write to file.

Extensions:

References:
UC-DAS105

Open Issues:

Comments:

Related work packages:

- Write documentation/help system
- Write s/w to examine observation and data reduction history

UC-DAS105: Display error and information messages

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December, 2001**

Brief description:
This use-case describes the flow of information between the IA and the user.

Phase:
All phases

Actors:
DP, IA

Triggers:
IA/QLA is in use.

Preconditions:
IA/QLA exists.

Minimal post-conditions:
The IA doesn't crash.

Success post-conditions:
The correct message is dispatched to the correct target.

Stakeholders and interests:
AST: wants to have a user-friendly interface
SM: wants to be able to trace and reproduce errors.

Main Success Scenario:

1. IA: event occurs that requires a message to be sent
2. IA: choose message(s) and target(s) for message(s)
3. IA: dispatch message to targets

Extensions:
3a: Message requires response
3a1 IA: prompts DP for response
3a2 DP: responds

References:

Open Issues:

Comments:
The level of messaging or prompting can be set by the DP or set to the default level. The behaviour depends on the type of message e.g. dialogue box, logging, scrolling screen.
System configurations can be saved and loaded by the user.

The messaging system can be configured to suggest possible sources of the error and directs the DP to the relevant documentation.

Related work packages:

- Write messaging system
- Write configuration management system (user defaults, messaging verbosity)
- Write documentation

UC-DAS106: Access interactive analysis documentation

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief Description:
This usecase describes how a user accesses documentation through IA. The level of documentation is specified by the user and may be invoked from IA.

Phase:
ILT -> Post Operations

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
The user needs help during an IA session.

Preconditions:
The documentation has been written into the IA.

Minimal post-conditions:
IA doesn't crash.

Success post-conditions:
The user is provided with the required information.

Stakeholders and Interests:
AST/IE requires information to efficiently and correctly reduce data

Main Success Scenario:
1. DP: Interacts with IA.
   1.1. DP: Requests IA documentation
   1.2. IA: Writes documentation to a specified output device.

Extensions:

References:

Open Issues:

Comments:

Related work packages:

- Provide hardcopy/online documentation
- Write s/w for IA documentation.

UC-DAS107: Record data reduction history

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief Description:
This usecase describes how data reduction history is written into the output files produced by the IA. The information will include version numbers of the IA and calibration artifacts, and the individual tasks with associated parameters that have acted on the data.

Phase:
ILT -> Post Operations

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
The IA acts on input data file and produces a new output file.

Preconditions:
IA exists.
Data exist and have been retrieved from the database.

Minimal post-conditions:
IA doesn't crash.
False history information is not written into the IA output file

Success post-conditions:
All relevant history information is written into the IA output file.

Stakeholders and Interests:
AST/IE needs to know the extent to which a given observation has been processed and version of software artifacts used.

Main Success Scenario:
1. DP: Interacts with IA.
   1.1. DP: Runs IA task
   1.2. IA: Writes relevant data reduction history into output file(s)

Extensions:

References:

Open Issues:

Comments:

Related work packages:

- Write s/w for recording data reduction history.

UC-DAS108: Visualise any data product

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **Issue**
Date: **6 September 2001**

Brief description:
This use case describes how any photometer data are visualized.
Visualization can involve a two-way exchange of information, e.g. cursor selection.
The display output is not restricted to screen; hardcopy output is also valid.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
Data need to be visualized.

Preconditions:
Photometer data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The data are visualized in the specified manner.

Stakeholders and interests:
AST/IE/IT/CS want to visualize data.

Main Success Scenario:
1. DP: Decides which regions of the data parameter space are to be visualized.
2. DP: Interacts with IA
   2.1. DP: Runs “Data visualization” S/W specifying data and criteria.
   2.2. IA: Displays data as requested.
3. DP: Inspects results.

Extensions:
3a. DP: Goes back to 1 to change way data is viewed.

References:
UC-DAC101
UC-DAC105
UC-DAS104
UC-DAS105
UC-DAS107

Open Issues:
What is the scope of the IA for data visualization? Is it not easier to convert the data to an external format and then view it with a familiar package, e.g. IDL?

Comments:
A list of types of things that we might want to visualize are:

- Results of processing steps, e.g. flagged bad data, filtered data
- Image with specified colour map.
- Contour/surface plots.
- X-Y plots, e.g. bolometer vs. bolometer noise
- Time-series.
- Overlay plotting
- 3D data.
- Magnification window.
- Panning
- Multiple displays
- Printing

It is not envisaged that all of this will be possible solely through SPIRE IA; the use of external visualization packages is recommended where possible to prevent unnecessary recoding of existing functionality.
This provides the functionality for visualizing the data and not any interactive processing, although this will be used as part of interactive processing, e.g. background selection.
The user can use the result of the display to select part of the data for further display and/or processing. This can be a non-interactive process (UC-DAC105).

Related work packages:

- Write software to visualize data (see comments)
- Create documentation/help system
- Collect requirements on visualization tools

UC-ENG102: Store analysis data

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7 December 2001**

Brief description:
This use case describes how analysis data from instrument engineering tests, either performed on the ground or in flight, are made persistent for future reference. It is assumed that, within the context of this use case, persistence will be provided by a database.
By analysis data we mean here not only the data produced by the analysis. One should also store the test data or a reference to them, the test procedure(s) used or a reference to it (them), the software used to analyze the test data or a reference to it, the test report, etc…

Phase:
ILT, IST, PV, Operations

Actors:
IE: Instrument engineer
IT: Instrument tester

Triggers:
A test has been performed and the analysis data is to be kept.

Preconditions:
Connection to suitable storage is available, e.g. a local database.

Minimal post-conditions:
User version of the data is not corrupted or permanently lost, a message describing the fate of the storage attempt is issued.

Success post-conditions:
- Data is stored persistently.
- It is possible to locate and retrieve the data.

Stakeholders and interests:
Calibration Scientist: May use the data from tests when producing calibration parameters for the instrument.

Main Success Scenario:
1. IT: Select data to be made persistent
2. IT: Request from the storage system the information needed to store the data

3. IT: Make the data persistent in the storage system.

Extensions:
3a: The data cannot be made persistent because of a problem in the data
    3a1. IT: updates the data to get rid of the problem
    3a2. IT: starts again at step 1 of MSS

3b: The data cannot be made persistent because of a problem in the storage system
    3b1. IT: Files a problem report

References:
UR-AIV3.2.3

Open Issues:
ILT may require separate data storage from the local node of the HCSS that is isolated from any network. If so then UR-AIV3.2.3 requires that the interface to this data storage is the same as that for the local node of the HCSS. Options include:

- ILT uses another copy of the database and replication is used to periodically copy it to the main ICC database.
- Data are stored in a simple custom database and periodically ingested into the main ICC database.

The use-case is very general because we expect that different kinds of data will need different procedures to be made persistent.

Comments:
The implementation of this use case is also a part of QLA.
Along with the analysis data itself, related material should also be stored: test description; any ‘quick and dirty’ analysis software used; test report, etc.
Alternatively a reference to these elements (which should already be stored somewhere in the system) could be stored with the analysis data.
Note that in some cases we may not wish to store the analysis data.

Related work packages
Create/maintain a persistent storage system
Create/maintain problem report system
Manage databases
Define analysis data format (part of the IA/QLA framework)

UC-FTS 103: Reconstruct of each scan/interferogram

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **13 Jan. 05**

Brief description:
This use-case describes how the scans and interferograms are reconstructed by grouping the corresponding movement data (see comments for definitions).

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The FTS data are acquired movement by movement. The scans and interferograms contain the scientific information and have to be reconstructed to apply the subsequent reduction steps.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The scans and interferograms are correctly reconstructed from the data set.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Reconstruction of scans and interferograms” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Retrieves the corresponding movement data.
   1.4. IA: Generates the desired data from the movement data.
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run
    1.2a1: advises user (UC-DAS105)
    1.2a2: continue

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS107: Record data reduction history.
UC-DAS105: Information and error messaging system

Open Issues:

Comments:
Definitions:
- *Movement*: one direction mirror travel.
- *Interferogram*: dataset concerning one detector and one movement.
- *Scan*: set of the N+M interferograms of one movement. N and M are the number of detectors in the two FTS bands.

Related work packages:
- Write software to reconstruct the scans and interferograms
- Create documentation/help system
- Write data reduction system

UC-FTS 104: Convert position counter to mechanical mirror position

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **Thursday, January 13, 2005**

Brief description:
This use-case describes how mirror position is transform from counter number to mechanical position.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The physical positions of the mirrors are needed to apply the subsequent reduction steps.

Preconditions:
FTS data exist and have been extracted from the DB.
Calibration table exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
All the mirror positions corresponding to the set of counter data are generated.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Convert counter data times to mirror position” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Retrieves the needed calibration table (UCF-489).
   1.4. IA: convert the counter data times to the mirror positions.
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run
    1.2a1: advises user (UC-DAS105)
    1.2a2: continue
1.2b: The interferograms are not available.
    1.2b1: IA: ask user to run the Reconstruction of each scan/interferogram S/W (UC-FTS103).
1.4a: Some counter data times are missing and LVDT data are available.
    1.4a1: IA: use the LVDT data to determine the missing values of the mirror position set
1.4b: Some counter data times are missing and LVDT data are not available
    1.4b1: IA: Identifies and interpolate the missing values.

References:
UC-FTS101: user-level use-case for the FTS processing.
UCF-489: Extraction of Artifact from HCSS
UC-FTS103: Reconstruction of each scan/interferogram
UC-DAC101: Flag bad and missing data
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system
UC-FTS119: Record data reduction history.

Open Issues:

Comments:
An optical incremental (relative) counter is used to determine the position; it can happen that some steps are missed. In the travel range covered by the absolute sensor (LVDT), its information can be used to correct from missing step(s) of the optical relative counter. This step is applied to every interferogram (UC-FTS103).

Requires a calibration table.

As the data are over-sampled, the missing data can be reconstructed from other adjacent data (UC-FTS105).

Related work packages:
- Write software to convert the counter data to the mirror position.
- Create documentation/help system
- Write data reduction system

UC-FTS 105: Generate array of signal vs. position

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
This use-case describes how the mirror positions obtained after UC-FTS104 and the detector signal are associated to build an array.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
To perform accurate Fourrier transformation on data, detectors signal and mirror position have to be associated in a regular array.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The array of signal vs. position is correctly generated.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “generate array of signal vs. position” S/W.
   1.2. IA: Asks user to specify the grid on which the interpolation will be done (either position or time).
   1.3. IA: Examine observation and data reduction history (UC-DAS104)
   1.4. IA: Interpolate the signal array on the regular position grid (position interpolation) or the position array on the regular time grid (time interpolation).
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.3a: This tool as already been run with the same method for step 1.2
1.3a1: advises user (UC-DAS105)
1.3a2: continue
1.3b: The mechanical positions of the mirror are not known
1.3b1: IA: ask user to run “Convert position counter to mechanical mirror position” S/W (UC-FTS104).

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-FTS104: Convert position counter to mechanical mirror position.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system
UC-FTS119: Record data reduction history.

Open Issues:

Comments:
It is foreseen that this use-case should also be able to restore missing/bad data with the help of the flag information (see UC-DAC101 and UC-FTS104).

Related work packages:
- Write software to generate array of signal vs. position
- Create documentation/help system
- Write data reduction system

UC-FTS106: Convert mechanical position to OPD (optical path difference) for each detector

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
The mechanical position of the mirror is converted to the optical path difference, using a calibration table.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The OPD corresponding to each detector signal is needed to compute the Fourrier transform.

Preconditions:
FTS data exist and have been extracted from the DB.
Calibration table exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
A correct OPD is associated to each detector signal.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Convert mechanical position to OPD” S/W.
   1.2. IA: Retrieves the corresponding calibration table (UCF-489).
   1.3. IA: Examine observation and data reduction history (UC-DAS104)
   1.4. IA: Computes the OPD for each mechanical position of the input data.
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:

1.3a: This tool as already been run with the same calibration table
- 1.3a1: IA: advises user (UC-DAS105)
- 1.3a2: continue

1.3b: This tool as already been run with another calibration table
- 1.3b1: IA: asks user to confirm the action.

1.3c: array of signal vs. position not determined.
- 1.3c1: IA: Asks user to run “generate array of signal vs. position” S/W (UC-FTS105).

References:

UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system
UC-DAS107: Record data reduction history.
UCF-489: Extraction of Artifact from HCSS

Open Issues:

Comments:

Require a calibration table, determined at the ground test phase.

Related work packages:

- Write software to convert mechanical position to OPD
- Create documentation/help system
- Write data reduction system

UC-FTS 107: Phase correct

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
The Fourrier transform must be applied to a set of data which is phase shift corrected.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
All the data corresponding to one scan have to be in phase before to apply the Fourrier transformation.

Preconditions:
FTS data exist and have been extracted from the DB.
Calibration table exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The phase is correctly corrected.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Phase correction” S/W.
   1.2. IA: Examine observation and data reduction history (UCDAS104)
   1.3. IA: Retrieves the corresponding calibration tables (UCF-489)
   1.4. IA: Apply the phase correction to the input data
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:

1.2a: This tool as already been run with the same calibration tables.
    1.2a1: advises user (UC-DAS105)
    1.2a2: continue
1.2b: The OPD of the data are not available.
    1.2b1: IA: ask user to run “Convert mechanical position to OPD” S/W (UC-FTS106)

References:
UC-FTS101: user-level use-case for the FTS processing.
UCF-489: Extraction of Artifact from HCSS
UC-FTS106: Convert mechanical position to OPD (optical path difference) for each detector.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
They may be two phase-corrections: optical and electronic (detector band pass and multiplexer delay).

Related work packages:
- Write software to do the phase correction.
- Create documentation/help system
- Write data reduction system

UC-FTS108: Re-grid data to obtain a ZPD point

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **13 January 2005**

Brief description:
One measurement of each interferogram must correspond to the ZPD (zero path difference). If no point fulfils this constrain, the interferogram is re-gridded.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The Fourrier transform result is calibrated by the value of the ZPD point.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
Each interferogram contains a ZPD point correctly obtained from the input data.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Re-grid data” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Determines of a re-grid is needed
   1.4. IA: Performs the re-grid
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run

1.2a1: advises user (UC-DAS105)
1.2a2: continue
1.2b: The data are not phase corrected.
1.2b1: IA: asks user to run “phase correction” S/W (UC-FTS107)

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-FTS107: Phase correct
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:

Related work packages:
- Write software to re-grid the data
- Create documentation/help system
- Write data reduction system

UC-FTS109: Correct responsivity

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
The responsivity of each detector is evolving with time. The data need to be corrected from this effect.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The user has data, which need to be corrected from responsivity variations.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
Data are correctly corrected from responsivity.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Correct responsivity” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Retrieves the appropriate calibration tables (UCF-489).
   1.4. IA: Corrects from responsivity
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run

1.2a1: advises user (UC-DAS105)
1.2a2: exit
1.2b: The data have not been re-grid and no point contains ZPD measurement.
1.2b1: IA: asks user to run UC-FTS108: Re-grid data

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS107: Record data reduction history.
UC-DAS105: Information and error managing system
UCF-489: Extraction of Artifact from HCSS

Open Issues:

Comments:
Needs change in responsivity to be computed from calibration measurements. There will be long- and short-term variations, according to the responsivity drift of each detector and to the velocity fluctuations of the mirror (frequency response of each detector) respectively.

Needs calibration tables.

Related work packages:
- Write software to perform responsivity correction
- Create documentation/help system
- Write data reduction system

UC-FTS110: Correct for time-dependent variation in flux

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
The telescope or the internal calibrator may have a time-dependent variation in flux, which needs to be corrected before the calibration is done.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
Flux calibration is needed and must be independent of any variation of the telescope or internal calibrator emission.

Preconditions:
FTS data exist and have been extracted from the DB.
History of the telescope and internal calibrator emission is available.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The emission of both the telescope and the internal calibrator are correctly estimated.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Determine HSO and Internal Calibrator emission” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Compute the HSO and Internal Calibrator emission
   1.4. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:

1.2a: This tool as already been run
    1.2a1: advises user (UC-DAS105)
    1.2a2: continue
1.2b: The data have not been corrected for the responsivity of each detector.
    1.2b1: IA: asks user to run UC-FTS109: Responsivity correction

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:
This step of the reduction will certainly need access to part of data from the previous and following observation => how do we solve the problems of the data proprietary right?
Do we need a use-case to recover the HSO and Calibrator properties (as temperature)?

Comments:
Variation in flux may come from the emission of either the HSO telescope or the FTS internal calibrator (Black Body).

Related work packages:
- Write software to estimate the telescope and internal calibrator emission.
- Create documentation/help system
- Write data reduction system

UC-FTS111: Correct for position-dependent variation in flux

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
This use-case describes how the position-dependent correction is applied.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The efficiency of the interference of the two beams is not uniform over the spatial range covered by detectors. Corrections have to be applied.

Preconditions:
FTS data exist and have been extracted from the DB.
Correction table exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The correct correction has been applied.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Correct for position-dependent variation” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Load the appropriate correction table (UCF-489).
   1.4. IA: Applies the correction.
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run
1.2a1: advises user (UC-DAS105)
1.2a2: continue
1.2b: Data have not been corrected for time-dependent variation in flux
1.2b1: IA: asks user to perform UC-FTS110: Correct for time-dependent variation in flux

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS107: Record data reduction history.
UCF-489: Extraction of Artifact from HCSS

Open Issues:

Comments:
The variations result from the efficiency in the interference of the two beams. Requires a calibration table (ground test phase).

Related work packages:
- Write software to correct for position-dependent variation
- Create documentation/help system
- Write data reduction system

UC-FTS112: Deglitch FTS data to a 1st order

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
This use-case describes how outliers are replaced or flagged.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The data contain outliers point, which have to be removed.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The outliers are flagged or removed.
The meaningful data are not changed,
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “flag or remove outliers (1st order degliching)” S/W.
   1.2. DP: Determines if the outliers will be removed or flagged.
   1.3. DP: Choose a method to determine the outliers.
   1.4. IA: Examine observation and data reduction history (UC-DAS104)
   1.5. IA: Determines the outliers
   1.6. IA: Flags or removes the outliers.
   1.7. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:

1.4a: Data have not been corrected for time-dependent variation in flux
1.4a1: IA: asks user to perform UC-FTS110: Correct for time-dependent variation in flux

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
The method can be a median-like method or any user-defined method.

Related work packages:
- Write software to flag or remove outliers
- Create documentation/help system
- Write data reduction system

UC-FTS113: Apodise (remove outlying frequency signals)

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.3**
Status: **draft**
Date: **3 January 2002**

Brief description:
This use-case describes how outlying frequency signal is removed.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
Part of the signal doesn't contains scientific information, it must be removed before executing the Fourrier transform

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The data have correctly been modified.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs "Apodise" S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Removes the outlying frequency signals.
   1.4. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run
   1.2a1: advises user

1.2a2: exit

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
This step in the reduction process is optional.

Related work packages:
- Write software to apodise
- Create documentation/help system
- Write data reduction system

UC-FTS114: Fourier Transform individual scans

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
This use-case describes how the Fourrier transform is applied to scans.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
Data have to be transformed from intensity vs. space into intensity vs. frequency (spectrum).

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
A correct spectrum per detector per scan is obtained.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Fourrier transform” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Apply the Fourrier transform to the input data, for each scan and for each detector.
   1.4. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run

1.2a1: advises user (UC-DAS105)
1.2a2: continue
1.2b: Data have not been processed enough to perform this step
1.2b1: IA: asks user to perform the required processing steps.

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:

Related work packages:
- Write software to Fourrier transform a scan.
- Create documentation/help system
- Write data reduction system

UC-FTS115: Remove instrument signature

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
This use-case describes how the instrument signature is removed.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The observed data contains both the scientific observation of some sky point, and the emission from the telescope and the calibrator. This part of the signal needs to be removed.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The undesirable part of the signal has been removed and the wanted signal is not altered.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Remove Instrument Signature” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Compute the instrument signature.
   1.4. IA: Remove the instrument signature.
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run
    1.2a1: advises user (UC-DAS105)
    1.2a2: continue

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:
This step may need to be performed before Fourrier Transform.

Comments:

Related work packages:
- Write software to remove Instrument Signature
- Create documentation/help system
- Write data reduction system

UC-FTS116: Produce a spectrum per sky pixel

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.3**
Status: **draft**
Date: **3 January 2002**

Brief description:
This use-case describes how averaging over scans produces a spectrum per sky pixel.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
All the scans covering the same sky pixel must be grouped together to produce a unique spectrum.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
A correct spectrum is produced for each sky pixel.
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Average over scans” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Determines all the scans covering the same sky point.
   1.4. IA: Average the previously determined scans.
   1.5. IA: Modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:

1.2a: This tool as already been run
    1.2a1: advises user
    1.2a2: exit
1.2b: Data have not been Fourrier Transformed.
    1.2b1: IA: asks user to perform UC-FTS114: Fourier Transform individual scans

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS107: Record data reduction history.

Open Issues:

Comments:

Related work packages:
- Write software to average over scans.
- Create documentation/help system
- Write data reduction system

UC-FTS117: Produce 3D data cube

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **0.4**
Status: **draft**
Date: **Thursday, 13 January 2005**

Brief description:
This use-case describes how the two celestial coordinates and the radiation frequency 3D cube is produced.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The final scientifically useful data is a spectral image, which have to be reconstructed from all the spectrum corresponding to an observation.

Preconditions:
FTS data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
Data reduction history is modified (UC-DAS107).

Stakeholders and interests:
AST/CS want to apply all the reduction steps to the data.

Main Success Scenario:
1. DP: Interacts with IA
   1.1. DP: Runs “Produce spectral image” S/W.
   1.2. IA: Examine observation and data reduction history (UC-DAS104)
   1.3. IA: Retrieves all the spectra corresponding to an observation.
   1.4. IA: Computes a 3D array.
   1.5. IA: modify the data reduction history (UC-DAS107)
2. DP: Inspects results.

Extensions:
1.2a: This tool as already been run

1.2a1: advises user (UC-DAS105)
1.2a2: continue

References:
UC-FTS101: user-level use-case for the FTS processing.
UC-DAS104: Examine observation and data reduction history (including software and calibration file versions and relevant house-keeping data)
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
The output coordinates can be chosen by user or can be transformed using UC-DAC107: Transform between sky and spacecraft coordinate systems

Related work packages:
- Write software to compute the 3D array
- Create documentation/help system
- Write data reduction system

UC-FTS118: Detect and identify lines

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **13 January 2005**

Brief description:
This use-case describes how lines are detected in a FTS spectrum. For some of the detected lines, an identification is proposed. The velocity of the instrument and the redshift of the source are taken into account.

Phase:
ILT -> Operations

Actors:
DP: Data processor

Triggers:
Emission and absorption lines observed by the FTS have to be detected and identified (e.g. to determine a redshift). Line parameters (e.g. width, profile) have to be extracted.

Preconditions:
FTS data is available (in the form of wavelength-calibrated spectra)
IA exists and includes the spectral analysis software.
Line database is available (from HCSS database or from user database).

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The lines in the FTS spectrum fulfilling user-sprecified criteria are detected and identified, their parameters are measured.

Stakeholders and interests:
CS needs to calibrate the FTS.
AST needs to obtain scientifically validated data (redshift, line identifications and parameters).
IS: needs to make sure the instrument is achieving its scientific goals.

Main Success Scenario:
1. DP: Interacts with IA.
    1.1. DP: Runs "Line Detection" SW.

1.2. DP: Identifies which regions of the data will be used (UC-DAC102, UC-DAS108, UC-DAC105).
1.3. DP: Selects a line database.
1.4. DP: Selects line detection criteria.
1.5. IA: Apply instrument velocity lambda correction.
1.6. IA: Detect lines.
1.7. IA: Obtain redshift.
1.8. IA: Apply source velocity lambda correction (redshift).
1.9. IA: Identify lines.
1.10. IA: Report line list (lambda, identification, parameters).
2. DP: Inspects results.

Extensions:
1.7a DP: run the redshift determination software.

References:
UC-FTS101: user-level use-case for the FTS processing
UC-DAC102 Filter data on any criteria
UC-DAS108 Visualise any data product
UC-DAC105 Identify and flag data

Open Issues:

Comments:
Due to the way the FTS produces spectra, any real spectrum is actually wavelength-calibrated.
User-specified criteria to detect lines could be

- Significance with respect to a noise level
- By selecting manually regions of the spectra
- By searching specifically for a set of lines (correlation analysis)
- …

The reason to extension 1.7 is that there are two main success scenario: 1) one knows the redshift and one needs to identify the detected lines, 2) one doesn't know the redshift and through correlation of the detected lines with position one search the redshift.
Because of the redshift, the Line database must include lines with wavelengths lower than 200 μm (down to 20μm).

This is a desirable item of the SPIRE IA. We may instead use an already existing spectral package or participate in a common development for a spectral analysis package.

Related work packages:
Write software to:

- detect lines according criteria (intensity, shape).
- correct for velocity (lambda shift).
- redshift determination.

- create/update the user line database (subset of a larger database on HCSS server)
- define the interface to the line database
- Create documentation s/w
- Create documentation/help system

UC-ICC101: Create or update a software artefact(s) (within the ICC)

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **8 August 2001**

Brief description:
This use case describes the steps involved in creating a new software artifact or updating an existing one. It covers the areas of setting up a sandbox environment, checking artifacts in/out of a configuration control system, validation and verification within the test environment, and documentation. It does not cover the release of an artifact.

Phase:
All phases

Actors:
SSD: Scientific software developer
ST: Software tester

Triggers:
There is a need to write or modify some software (manifested e.g. through an SPR or an SCR).

Preconditions:
A configuration control system (CCS) is available.
A sandbox is available.
A software test environment is available.

Minimal post-conditions:
The integrity of the CCS is maintained.

Success post-conditions:
- The software is created or modified.
- The software is entered into the CCS.
- Any applicable documentation is created or modified (**uses** UC-ICC104)

Stakeholders and interests:
- SSD: responsible for software creation or update.
- ST: responsible for ensuring the software is validated.
- CC: responsible for ensuring only validated software is entered into the system.

Main Success Scenario:

1. SSD: prepare a sandbox.
2. SSD: Checkout the artifact from the CCS into sandbox.
3. SSD: Write or modify the unit test.
4. SSD: Write or modify the software.
5. SSD: Test the software within the sandbox.
6. SSD: Write or modify the documentation.
7. ST: prepare the test environment.
8. ST: Test and validate the software within the test environment.
9. ST: Enter the artifact into the CCS.

Extensions:

2a Not necessary for a new item.

8a ST: If the test fails, give test results to the SSD.
8a1 SSD: go back to step 1

9a In some case, approval from the ICC or some configuration control board will be necessary.

References:

UR-ICC100 (UR-ICC-110-150)
SIRD-ICCF-180
SIRD-ICCF-185
UCF-361 “Analyse an SCR
UCF 371 “Implement an SCR”
UCF-421 “Submit an SCR”
The release of a software artifact into the HCSS is covered by
UCF-371 Implement an SCR
UCF-372 Update/Create a software artefact
UCF-373 Supply an SCR implementation

Open Issues:

How does the OODBMS fit into this? The system consists not only of software, but also of data.
The process that updates the test environment following a change in the CCS is not part of this use-case, we need a use-case for it.
If this is triggered by something outside of the formal SPR/SCR structure, are we sure that the documentation produced is adequate to understand the changes.

Comments:

The CCS is expected to be CVS.

Note that sandbox refers to our definition of the sandbox: a software environment completely isolated from the ICC system in the sense that elements of the ICC system can get in, but not out. The sandbox is a personal environment. It can contain any piece of software, including or exclusively the test environment. This is where developments will occur. We are not using the term development environment because it is used in the HCSS with a different meaning.

A unit test is a software element used to test another software element.

S/W for the HCSS must adhere to its standards. Documentation for the HCSS must adhere to its standards.

In the test environment (step 8) there is a test plan which may include scientific validation.

For step 9, In some case, approval from the ICC or some configuration control board will be necessary.

The release of a software artifact into the HCSS is covered by
UCF-371 Implement an SCR
UCF-372 Update/Create a software artefact
UCF-373 Supply an SCR implementation

Associated work-packages

- Provide ICC Configuration Control system
- Define Science verification test plan
- Provide and maintain ICC Sandbox environment
- Provide and maintain ICC test environment

UC-ICC105: Place an item into configuration control

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **4 October 2001**

Brief description:
An item has been created or updated, and it has to be placed under configuration control to ensure both that it fits in the system, that a safe future development can occur, and allows to revert to a previous version of the system.

Phase:
All phases

Actors:
CC: Configuration controller
IH: Information handler

Triggers:
A new system item has been created or updated and needs to be stored in the configuration control system to ensure a safe development of the system.

Preconditions:
A configuration control system is available.
A definition of privileges for all ICC members is available.
A definition of responsibilities for all ICC members is available.

Minimal post-conditions:
Elements currently under CC are left in a consistent state.
In case of failure, the element in the CC system is not updated/created, a reason for the failure to update/create is determined and provided to the originator of the element.

Success post-conditions:
The item is placed under configuration control, the originator of the item is informed of the success.

Stakeholders and interests:
CC: wants to keep control of the system development.
ICC manager: needs to ensure that ICC development is under control.
The author of the item creation/update is an obvious stakeholder.

Main Success Scenario:
1. CC receives item to be placed under configuration control
2. CC obtains from item author item's status (new or update)
3. CC checks author is allowed to create/update elements of the CC system

4. CC checks that item is not already reserved by another system developer
5. CC register item in the configuration control system
6. CC/IH notifies item author of the success

Extensions:
3a: CC finds author is not allowed to create/update elements of the CC system
3a1. CC: notifies ICC manager of the conflict and exits use-case

4a: CC finds that item is already reserved by another system developer
4a1.CC: notifies item author and software manager of the conflict and exits use-case

5a: CC cannot register item in the configuration control system
5a1. CC: files a problem report

References:
UCF-032 Maintain version/configuration control environment

Open Issues:
What or where is the configuration control system is probably out of the scope of this use-case (i.e. are we using our own system, a subset of the HSC system operated only by us, a subset of the HSC system operated jointly by us and the HSC?).

Comments:
The author of the item is not specified because the item can be anything worth putting under CC and not necessarily a piece of code. For instance, numbers resulting from calibration measurements (i.e. calibration tables), documents, memoranda resulting from expert advice on particular aspects of the instrument could also be placed under configuration control.

Related work packages:
- ICC configuration control system
- Information handling
- Problem report system
- Organize privileges and responsibilities within the ICC (manage the ICC).

UC-ICC107: Retrieve artefact from the database

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **4 October 2001**

Brief description:
In this use-case, an ICC member connects to the database to retrieve an artefact (whole software, piece of code, data, a table…) and receives it in his personal environment.
This is in fact an HCSS use case: UCF-489 Retrieve archive artefact. This is the reason why all items are left blank.

Phase:

Actors:

Triggers:

Preconditions:

Minimal post-conditions:

Success post-conditions:

Stakeholders and interests:

Main Success Scenario:

Extensions:

References:

Open Issues:

Comments:

Related work packages:

UC-ICC108: Access database from outside the ICC

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.1**
Status: **issue**
Date: **19 June 02**

Brief description:
This use case describes how an actor external to the ICC Operations Centre at RAL accesses the HCSS database system in order to store or retrieve artefacts.

Phase:
All phases

Actors:
EU: External User
DB: Database

Triggers:
An actor external to RAL requires database access.

Preconditions:
- The database is available.
- Network services are available.
- The user has authenticated his or her self.
- A list of user permissions has been constructed.
- Other preconditions depend on the exact scenario implemented (see below)

Minimal post-conditions:
- The security of the site computers and networks is not compromised.
- The integrity of the database is maintained.

Success post-conditions:
The database is successfully queried or updated.

Stakeholders and interests:
Database manager: Has an interest in seeing that data in the database are accessible and that use of the database does not corrupt data.
Site Security Managers: Have responsibility for ensuring that the security of their sites is not compromised.

Main Success Scenario:
1. EU: Issue a command to interact with the database.
2. DB: validates that the user has the privilege to perform the requested operation

3. DB: Return the selected data, or store the given data

Extensions:
2a: The user does not have the privilege to perform the requested operation
2a1. DB: Report a privilege violation failure to EU
2a2. DB: Log the failure

3a: Data cannot be returned or stored
3a1. DB: Report an error to EU
3a2. DB: Log the error

References:

Open Issues:
DAPSAS centers can potentially access data in a number of ways:
1. From their own local database and use replication to get data from RAL.
2. Copying data in a file-based format, using ftp or equivalent.
3. Logging in at RAL and running applications remotely.
4. Using local clients to transparently access data over the network from RAL.

Currently Option (4) is favoured.

Should the DAPSAS centers have identical privileges to RAL?
If not, what privileges should the DAPSAS centers have over normal consortium members?

In the Operations and Archive phases, it is possible that all data is served directly from the HSC.

Comments:
From the actors' point of view, this Use Case is an extremely simple one. All the details are in the implementation.

This Use Case also applies perfectly well to ICC actors at RAL. Privileges could be granted on the basis of roles, so (for example) only certain staff would be allowed to modify the CUS scripts.

This Use Case has security implications that must be considered.

Network connections could be implemented either with a normal socket-based TCP/IP connection, or using HTTP via a Web server. The HCSS will provide support for this.

Related Work Packages

Infrastructure	System management Database management
Access Control	Requirements definition

(I assume this is common…)	Analysis and prototyping Implementation Maintenance

UC-ICC109: Request ICC system access privilege

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date **November 5th 2001**

Brief description:
This use case describes how a new or existing user of ICC systems makes a request for privileges to access these systems.

Phase:
All phases

Actors:
EU: External User
SM: System manager e.g. database or computer system manager

Triggers:
A new user wants access to ICC systems or wants privileges changed.

Preconditions:
Access control is implemented.

Minimal post-conditions:
System security is not compromised.

Success post-conditions:
The user has access to the desired ICC systems with the desired privileges.

Stakeholders and interests:
SPIRE PI: Unauthorised users do not have access to SPIRE science data.
Site Security Managers: Have responsibility for ensuring that the security of their sites is not compromised.

Main Success Scenario:
1. EU: Requests system access or changed privileges from SM.
2. SM: Grants access or privileges.
3. SM: Informs user that the change has been made.
4. SM: Logs the access or privilege change.

Extensions:
3a SM: Informs user that the request could not be granted.

References:

Open Issues:

Comments:

Decision on privileges or access may be made by someone other than the SM. Certain types of request e.g. data access with normal privileges may not require interaction with SM, and a registration facility may be adequate.

Certain modes of access, such as being able to login directly to a machine at RAL, are likely to be restricted to the DAPSAS centres.

Related Work Packages

Infrastructure	System management Database management Configuration Control Information Dissemination (includes web site)
Access Control (I assume this is common…)	Requirements definition Analysis and prototyping Implementation Maintenance

UC-ICC111: Reach ICC personnel out of normal hours

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6 November 2001**

Brief description:
This use case describes the action performed when abnormal behaviour of the instrument is reported out of normal office hours. The MOC and SOC may require a rapid response to a query about the instrument.

Phase:
Commissioning, PV, Operations

Actors:
PR: Problem reporter
IM: ICC Manager
IE: Instrument engineer(s)

Triggers:
Anomalous instrument behaviour is observed. In Commissioning and PV phase this is likely to be initiated by ICC@MOC; in operations it will probably be MOC or HSC.

Preconditions:
- The procedure for dealing with instrument anomalies has been agreed and written down.
- Overtime payments and on-call allowance are agreed and budgeted for.

Minimal post-conditions:
PR reports problem to IM.

Success post-conditions:
The problem is reported to the appropriate person(s) and the problem is dealt with.

Stakeholders and interests:
HSC wants the mission to keep functioning properly
ICC wants to keep instrument functioning properly
IM and IE don't want to be woken up for anything unimportant

Main Success Scenario:
1. PR reports problem to IM or delegated deputy.
2. IM reports problem to the appropriate person(s) i.e. IE.
3. The IE analyses the problem and takes appropriate action.

Extensions:

References:
UR-ICC3.4.2

Open Issues:
It is clear that this Use Case is a special case and **extends** some other Use Cases for reporting problems within office hours.

Comments:
The PR may be a variety of people depending on operations phase. During Commissioning and PV it may be ICC@MOC, during operations it may be the normal nighttime instrument operator.

Related work packages:
- Set up ICC@MOC
- Contingency

UC-PHT108: Resample and combine data spatially and/or temporally

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6 December 2001**

Brief description:
This use case describes how data products such as light-curves and images are produced. The data product could have 1 dimension (e.g. a light-curve), 2 dimensions (e.g. an image) or more (e.g. a stack of phase folded images). The data will have to be re-sampled onto some spatial and/or temporal grid to produce the data product. There is a variety of coordinate grids that the user may want to use: eg sky position, location of bolometer in the array, distance from some astronomical source, time. The user may also want to compile statistical information about the data product for example a variance array.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The user wants to produce a data product by binning and/or re-sampling the data.

Preconditions:
Photometer data exist and have been extracted from the DB.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The required data product is produced

Stakeholders and interests:
AST/IE/IT/CS want to produce images, light-curves, etc from their data.

Main Success Scenario:
1. DP: Decides what sort of data product is required.
2. DP: Interacts with IA

2.1. DP: Runs “Resample and combine data” S/W specifying dimensions of the required product.
2.2. IA: Provides sensible defaults for the dimensions of the data product
2.3. DP: Specifies dimensions and other details of output data product or accepts defaults.
2.4. IA: Re-samples data to produce data product.
2.5. IA: If specified, produces variance array to match data product and reports mean statistical quantities.
2.6. IA: Modifies data reduction history (UC-DAS107)
3. DP: Inspects results.

Extensions:
2.1a DP: Data do not contain the required information (eg time) required to produce the data product. Process stops

References:
UC-DAS108
UC-DAS107

Open Issues:

Comments:
A list of coordinate systems we might want to resample the data to are:
1. general sky coordinate systems
2. Time, spacecraft or barycentric
3. A specific sky coordinate system read from an image taken at a different wavelength
4. Bolometer position
5. Phase of something periodic
6. Distance from a particular astronomical source

The re-sampling can occur at any step of the reduction process
As mentioned in UC-DAS108 visualization may require re-sampling data in which case DP is IA.

Related work packages:
- Write software to Re-sample and combine the data. Collect requirements on re-sampling.
- Create documentation/help system
- Write data reduction history system.

UC-PHT 110: Determine colour correction

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6th December 2001**

Brief description:
As a photometer measures the flux over a broad spectral bandpass, the relationship between the flux at the effective wavelength of the filter, and the amount of flux recorded by the bolometer, will be a function of the intrinsic spectrum of the target. A colour correction is required to determine the monochromatic flux of an object or to transform the measurement to the photometric system of another instrument. Default spectra for common types of astronomical source should be provided so that users do not have to go to the trouble of obtaining such spectra. If the user has colours for sources, for example from the SPIRE photometers, but does not know what their spectra really should be, then some empirical colour correction will be required.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The user wants to determine the monochromatic flux from the photometer measurement, or to transform the photometer measurement to a different photometric system.

Preconditions:
Photometer data exist and have been extracted from the DB. The filter response curves for SPIRE are available, and if the user wishes to convert to a different photometric system, the response curve for that system is also available.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The appropriate colour correction is produced.

Stakeholders and interests:
AST/CS want to be able to interpret SPIRE photometry.

Main Success Scenario:

1. DP: Decides what monochromatic wavelength, or what photometric system to convert to.
2. DP: Interacts with IA
   2.1. DP: Runs “Determine colour correction” S/W and provides a template spectrum, uses one of the defaults, or provides colours for the sources.
   2.2. IA: Reports on validity of colour correction
   2.3. IA: Computes colour correction
3. DP: Inspects results.

Extensions:

2.2a. IA reports the colour correction is meaningless i.e. template spectra, and/or SPIRE response curve do not cover the wavelength range of the target photometric system or requested wavelength.

2.2a1: Process stops

References:
UC-PHT111

Open Issues:
What sort, and how many default spectra should the IA contain? Someone has to calibrate the empirical colour corrections, and/or decide what algorithms might be useful for this purpose.

Comments:
The task must deal with source lists or single sources and if the colour corrections are being determined from the colours of the sources the task will have to compute a colour correction individually for each source in the list and store it in a fashion that can be interpreted by a person and by the ‘Apply colour correction’ task (UC-PHT111)

Related work packages:

- Write software to produce colour correction
- Create documentation/help system
- Compile default spectra and response curves for other photometric systems.
- Determine the empirical colour corrections.

UC-PHT 111: Apply colour correction

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief description:
As a photometer measures the flux over a broad spectral bandpass, the relationship between the flux at the effective wavelength of the filter, and the amount of flux recorded by the bolometer, will be a function of the intrinsic spectrum of the target. A colour correction is required to determine the monochromatic flux of an object or to transform the measurement to the photometric system of another instrument.
This colour correction can be supplied by the user, or calculated by the 'Determine colour correction' task.

Phase:
ILT -> post-operations

Actors:
DP: Data processor
IA: Interactive analysis

Triggers:
The user wants to apply a colour correction.

Preconditions:
Photometer data exist and have been extracted from the DB. The colour correction has already been determined.

Minimal post-conditions:
IA does not crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
The desired colour corrections are applied.

Stakeholders and interests:
AST/CS want to be able to interpret SPIRE photometry.

Main Success Scenario:
1. DP: Decides what colour corrections to apply.
2. DP: Interacts with IA
   2.1. DP: Runs "Apply colour correction" S/W and provides the colour correction(s).
   2.2. IA: Applies colour correction(s)
   2.3. IA: Modifies data reduction history (UC-DAS107)

3. DP: Inspects results.

Extensions:

References:
UC-PHT110
UC-DAS107

Open Issues:

Comments:
The colour corrections may be specified by the user as a number or set of numbers, or as a table of values. The task should deal with source lists and accept a table of colour corrections appropriate for each individual source or a global colour correction.

Related work packages:
- Write software to apply colour correction
- Create documentation/help system
- Write data reduction system

UC-PHT113: Detect sources

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **7th December 2001**

Brief Description:
This usecase describes how sources are identified and extracted from an image. There may be a number of algorithms available to the user.

Phase:
ILT -> Post Operations

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
The user has an image and wants to identify detected sources and candidate detected sources and their parameters (positions, fluxes, signal-to-noise, extent etc)

Preconditions:
The photometer data exist, have been retrieved from the database and have been processed into an image.

Minimal post-conditions:
If the process fails, a clear and relevant error message is produced (UC-DAS105) and the original data are not modified.

Success post-conditions:
If sources are identified then graphical and tabular output is produced, otherwise a null result is returned.
Data reduction history is modified (UC-DAS107).

Stakeholders and Interests:
IS/AST wants statistically sound method of source detection.

Main Success Scenario:
1. DP: consults documentation and selects desired/appropriate algorithm.
2. DP: interacts with IA
   2.1. DP: Runs “detect sources” algorithm
   2.2. DP: Inputs required parameters
   2.3. IA: Applies requested algorithm with supplied parameters
   2.4. IA: Interacts with graphical interface marking identified sources and produces tabulated source list with relevant physical quantities, and tagged with input parameters (UC-DAS107)

Extensions:

References:
UC-DAS105
UC-DAS107

Open Issues:

Comments:
Step 1 – ‘desired’ could be the DP’s own algorithm.

Related work packages:
- Write source detection s/w artifact(s)
- Write documentation/help system

UC-PHT121: Subtract off-source data (demodulate data)

Level: **sub-function**
Scope: **SPIRE ICC**
Version: **1.0**
Status: **issue**
Date: **6 September 2001**

Brief description:
This use-case describes how negative (off-source) beam is subtracted from positive (on-source) beam for data taken in chopping modes.

Phase:
ILT -> post-operation

Actors:
DP: Data Processor
IA: Interactive analysis

Triggers:
Chopped data need to be demodulated.

Preconditions:
Photometer data exist and have been extracted from the DB.
IA exists and includes the demodulating software.

Minimal post-conditions:
The IA doesn't crash.
If the process fails, a clear and relevant error message is produced and the original data are not modified.

Success post-conditions:
Chopped data are demodulated.
Data reduction history is updated.

Stakeholders and interests:
AST/IE/IT/CS want to demodulate data taken in chopping modes.

Main Success Scenario:
1. DP: interacts with IA.
   1.1. DP: runs "Demodulation".
   1.2. IA: demodulates the data.
   1.3. IA: modifies the data reduction history (UC-DAS107).

Extensions:
1.1a. IA: determines that data are already demodulated and then stops the process (UC-DAS105).

References:
UC-DAS105: Information and error messaging system.
UC-DAS107: Record data reduction history.

Open Issues:

Comments:
Any operation that modifies the data has two modes: creating a new artifact or overwriting the old one.
Synchronization of chopping data is done when IA gets the data from the database.
May be several methods to demodulate data.

Related work packages:
Write the demodulating software.
Write the history software.
Create the documentation/help system.

Appendix: ILT and QLA Use Cases

The following set of ILT and QLA use cases show how the HCSS Use Cases for ILT (see HCSS use cases for ILT technical note, Issue 1.0, 8th Feb 2002, FSCDT/TN-023) relate to the SPIRE ICC use cases for ILT and QLA. As the high level HCSS ILT use cases were written jointly by the three ICCs and the HSC it is therefore not necessary to re-write them for the SPIRE ICC.

The HCSS use cases are identified by the prefix “UCF”, while the SPIRE ICC ones are prefixed with “UC”-. All the minor changes that have been made to the HCSS use cases here are shown in **bold face**.

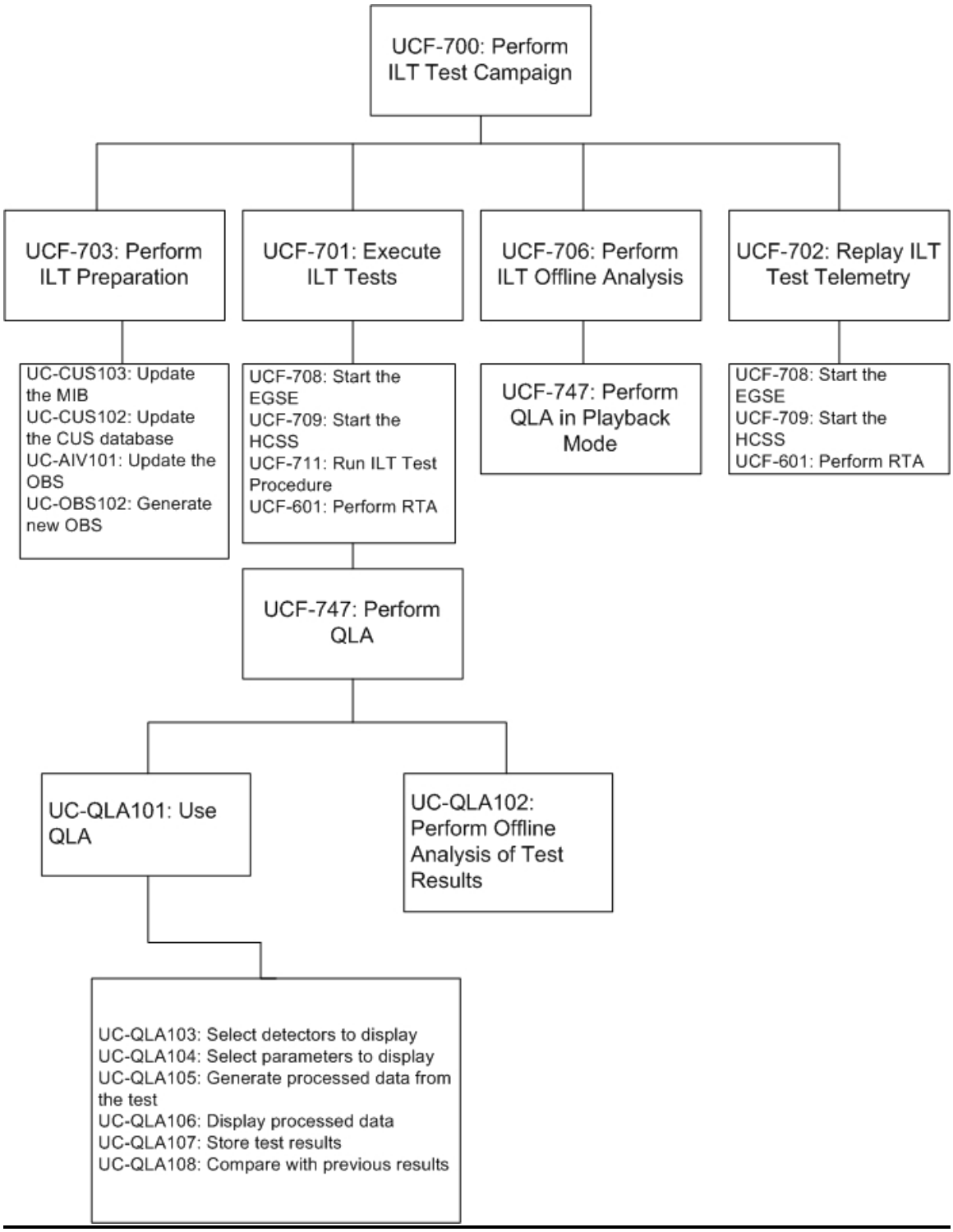


Figure 1: SPIRE ICC and HCSS use cases for ILT and QLA

UCF-700 [Summary]: Perform ILT test campaign

Level: **summary**
Scope: **EGSE-ILT and HCSS**
Version: **1.0**
Status: **issue**

Brief description:
For each instrument model the ILT will be split into test campaigns. This summary level use-case describes the preparation for a test campaign, the execution of the tests identified in the test campaign and the subsequent analysis of the results.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IE: Instrument engineer (primary actor)
IT: Instrument tester (secondary actor)
ICC: Instrument control centre personnel (secondary actor)

Triggers:
Testing of an instrument is to be performed. Normal work during ILT.

Preconditions:
None.

Minimal postconditions:
None.

Success postconditions:
The test campaign is completed successfully.

Stakeholders and interests:
TBW.

Main success scenario:
1 IE: Perform ILT preparation [UCF-703]
2 ICC: Perform test campaign readiness review (procedural)
3 IT: Execute ILT tests [UCF-701]
4 IE: Perform ILT offline analysis [UCF-706]

Step 4 can be performed in parallel to step 3 (see comments)
5 ICC: Perform formal post test campaign review (procedural)

Extensions:
2a: ICC: Fail test campaign readiness review
3a: IT: Replay ILT test telemetry [UCF-702]
4a: ICC: Fail formal post test campaign review

Frequency of occurrence:
Often during ILT

References:
None.

Open issues:
• None.

Comments:
• There can be several tests associated with a campaign. Offline analysis of the results of a
previous test can be performed while a new test is being run.

UCF-703 [Summary]: Perform ILT test preparation

Level: **summary**
Scope: **HCSS and SPIRE ICC/EGSE-ILT**
Version: **2.1**
Status: **issue**

Brief description:
This summary level use case describes how the instrument engineer actor prepares for ILT testing. The actor must ensure that the necessary building blocks and observing modes are defined within the HCSS archive. Using the EGSE-ILT system the instrument engineer uses these observing mode definitions to create test procedures templates. These test procedure templates are also stored in the HCSS archive using an HCSS API. A MIB is ingested into the HCSS archive for use in the creation of building blocks and observing modes. Existing building blocks and observing modes must be validated against a new MIB if the new MIB contains modified command definitions. Additionally, the use case identifies how a MIB is imported into the HCSS (using a MIB version control/ storage mechanism) and how a required MIB is subsequently exported out of the HCSS when it is required by an external system (the EGSE-ILT system).

Now includes extensions to MSS for the SPIRE ICC.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IE: Instrument engineer (primary actor)

Triggers:
Testing of an instrument is to be performed [UCF-700]

Preconditions:
None.

Minimal postconditions:
None.

Success postconditions:
The HCSS archive contains everything necessary to support the ILT tests [UCF-700].

Stakeholders and interests:
TBW.

Main success scenario:
1 IE: Ingest MIB into HCSS [UCF-756]
2 IE: Import MIB (as a BLOB) into HCSS [UCF-761]
3 IE: Revalidate building blocks/ observing modes [UCF-704]
4 IE: Define building block or observing mode [UCF-752]
5 IE: Define test procedure template (EGSE-ILT) [UCF-757]
6 IE: Export MIB (as a BLOB) out of HCSS [UCF-762]
Steps 1 to 6 can be optional, performed in parallel, taken in different orders or repeated.

Extensions:
1a IE: Update the MIB (SPIRE ICC use case UC-CUS103)

3a IE: Update the CUS database (SPIRE ICC use case UC-CUS102)

4a IE: Update the CUS database (SPIRE ICC use case UC-CUS102)

4b IE: Update the OBS (SPIRE ICC use case UC-AIV101)

4c IE: Generate new OBS (SPIRE ICC use case UC-OBS102)

Frequency of occurrence:
Often during ILT

References:
TBW

Open issues:
• None.

Comments:
• Note: Test control can use test procedure templates retrieved from the HCSS for subsequent execution by SCOS 2000. It also permits the sending of individual telecommand mnemonics to SCOS 2000. This use case relates specifically to the use of the test control interface between test control and the HCSS.
• The steps shown in the main success scenario can be optional, can be performed in parallel, can be taken in different orders, and can be repeated. For example:
• Several MIBs may need to be held in the database.
• A MIB can be exported at the same time as building blocks and observing modes are being defined.
• A previously stored MIB can be exported while a new MIB is being imported/ingested.

• A MIB could be exported before building blocks/ observing modes are defined.
• If a new MIB does not contain modified command definitions then it will not be necessary to perform building block/ observing mode revalidation.
• Export MIB (as a BLOB) out of HCSS for use in the EGSE environment. RTA analyses the housekeeping telemetry packets. It must have access to the MIB which was used in the generation of the telecommands which are to be sent to the instrument.

UCF-701 [Summary]: Execute ILT tests

Level: **summary**
Scope: **EGSE-ILT and HCSS**
Version: **2.0**
Status: **issue**

Brief description:
This use case describes the steps that will be involved in performing a series of tests on an instrument. This includes the starting of the systems involved in the tests, the running of the tests, and the monitoring and analysis of the results from these tests.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IE: Instrument engineer (primary actor)
IT: Instrument tester (secondary actor)

Triggers:
ILT testing of the instrument is to be performed [UCF-700].

Preconditions:
ILT test preparation has been performed [UCF-703].

Minimal postconditions:
None.

Success postconditions:
The tests identified for the test campaign are executed and the resulting telemetry analysed.

Stakeholders and interests:
TBW.

Main success scenario:
1 IE: Start the EGSE [UCF-708]
2 IE: Start the HCSS [UCF-709]
3 IE: Setup hardware in test configuration (EGSE-ILT)
4 IT: Execute test

4.1 IT: Run ILT test procedure (EGSE-ILT) [UCF-711]
4.2 Router: Receive telemetry packet and forward to registered clients (EGSE-ILT)
4.3 Telemetry ingestion: Receive telemetry and process it (HCSS) [UCF-758]

4.4 RTA: Receive telemetry and process it (EGSE-ILT) [UCF-601]
4.5 QLA: Receive telemetry and process it (HCSS) [UCF-747]
Repeat steps 4.2 to 4.5 until test completes
5 IE: Perform test completion process
5.1 IE: Set hardware to nominal mode
5.2 IE: Mirror HCSS archive to an offline archive
Repeat steps 3 to 5 until all tests in the campaign are completed
6 HCSS: Ingest TC history into HCSS [UCF-759]
7 HCSS: Ingest out of limits history into HCSS [UCF-763]
Steps 6 and 7 are performed at regular intervals throughout the test campaign (see comments).
8 IE: Shutdown the HCSS and EGSE
9 IE: Shutdown the instrument and EGSE hardware

Extensions:
None.

Frequency of occurrence:
Often during ILT

References:
HCSS-UR-3.1-0840 (partial)

Open issues:
• None.

Comments:
• Setting the system to a nominal mode will involve, for example, leaving the instrument in the cryostat but not actively being used.
• Mirroring the HCSS archive to an offline archive ensures that offline analysis can take place in parallel to the actual testing [see UCF-700].
• Automatic analysis of the test results (telemetry) is not required for v 0.1 of the HCSS [**HCSS Open Issues, Issue 2.1, 23 May 2002, FIRST/FSC/DOC/0159**: Open issue 2.12.5].
• Running QLA/IA requires that the proper algorithms and procedure are present in the HCSS [**HCSS Open Issues, Issue 2.1, 23 May 2002, FIRST/FSC/DOC/0159**: Open issue 2.12.5].
• TC history ingestion and OOL history ingestion are continuously running processes which will checks for updates to the TC history and the OOL history at regular intervals.

UCF-702 [Summary]: Replay ILT test telemetry

Level: **summary**
Scope: **EGSE-ILT and HCSS**
Version: **2.0**
Status: **issue**

Brief description:
This summary level use case describes the steps that will be performed for instrument telemetry playback (replay) during ILT.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IT: Instrument tester (primary actor)
EGSE-ILT: The electrical ground segment equipment system for ILT.

Triggers:
It is necessary to replay ILT telemetry [UCF-700].

Preconditions:
Test preparation has been performed [UCF-703].

Minimal postconditions:
None.

Success postconditions:
The telemetry playback is successful.

Stakeholders and interests:
TBW.

Main success scenario:
1 IT: Start EGSE [UCF-708]
2 IT: Start HCSS [UCF-709]
3 IT: Start playback selector and identify playback telemetry (HCSS) [UCF-705]
4 Router: Register playback selector (EGSE-ILT)
5 Playback selector: Retrieve TM from database and forward to router (HCSS) [UCF-704]
6 Router: Receive telemetry packet and forward to registered clients (EGSE-ILT)

7 RTA: Receive telemetry and process it (EGSE-ILT) [UCF-601]
Repeat steps 5 to 7 until telemetry playback is complete

Extensions:
None.

Frequency of occurrence:
Often during ILT

References:
TBD

Open issues:
• None.

Comments:
• None.

UCF-706 [Summary]: ILT offline analysis

Level: **summary**
Scope: **EGSE-ILT and HCSS**
Version: **1.0**
Status: **issue**

Brief description:
This use case describes the steps that will be performed during offline analysis.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IT: Instrument tester (primary actor)

Triggers:
It is required to perform offline analysis of instrument test data [UCF-700].

Preconditions:
The required test data is in the archive [UCF-701].

Minimal postconditions:
None.

Success postconditions:
The instrument test data is successfully analysed.

Stakeholders and interests:
TBW.

Main success scenario:
1 IE: Start telemetry and data frame server (optional)
2 IE: Run QLA in playback mode [UCF-747]
3 IE: Run interactive analysis

Extensions:
None.

Frequency of occurrence:
Often during ILT

References:
TBD

Open issues:
• None.

Comments:
• None.

UCF-708 [Summary]: Start the EGSE

Level: **summary**
Scope: **EGSE-ILT**
Version: **1.0**
Status: **issue**

Brief description:
This use case details the steps that will be involved in starting the EGSE processes prior to the testing of an instrument, or the playback of instrument test telemetry, during ILT.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IE: Instrument engineer (primary actor)
HCSS: Herschel common science system (secondary actor)

Triggers:
Testing of the instrument is to be performed [UCF-701].
Playback of instrument data is to be performed [UCF-702].

Preconditions:
Test preparation has been performed [UCF-703].

Minimal postconditions:
None.

Success postconditions:
The EGSE processes are successfully started.

Stakeholders and interests:
TBW.

Main success scenario:
1 IT: Start router (EGSE-ILT)
2 IT: Start telemetry gateway and configure for test (EGSE-ILT)
3 Telemetry gateway: Register with router (EGSE-ILT)
4 IT: Start RTA [UCF-601]

5 IT: Start test control (EGSE-ILT) (optional) (see comments)

Extensions:
None.

Frequency of occurrence:
Often during ILT

References:
HCSS-UR-3.1-0840 (partial)

Open issues:
• None.

Comments:
• For the purposes of this use case it has been shown that the instrument tester starts the processes. However, it is possible that some of the EGSE processes will be started automatically as part of a boot process.
• If telemetry playback is to be performed [UCF-702] then it will not be necessary to start test control. The telemetry will be coming directly from the HCSS archive.

UCF-709 [Summary]: Start the HCSS

Level: **summary**
Scope: **HCSS**
Version: **1.0**
Status: **issue**

Brief description:
This use case details the steps that will be involved in starting the HCSS processes prior to the testing of an instrument during ILT or the playback of telemetry from the HCSS archive.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IT: Instrument tester (primary actor)
EGSE: Electrical ground segment equipment (secondary actor)

Triggers:
Testing of the instrument is to be performed [UCF-701].
Playback of instrument data is to be performed [UCF-702]

Preconditions:
Test preparation has been performed [UCF-703].
The EGSE systems have been started [UCF-708].

Minimal postconditions:
None.

Success postconditions:
The HCSS processes are successfully started.

Stakeholders and interests:
TBW.

Main success scenario:
1 IT: Start telemetry ingestion and configure for test (HCSS) [UCF-758]
2 Telemetry ingestion: Register with router (EGSE-ILT)
3 IT: Start QLA in NRT mode (HCSS) [UCF-747]
4 Router: Register QLA (EGSE-ILT)

5 IT: Start TC history ingestion process [UCF-759]
6 IT: Start OOL history ingestion process [UCF-763]

Extensions:
1-6a: IT: Start telemetry playback selector [UCF-705] (see comments)

Frequency of occurrence:
Often during ILT

References:
HCSS-UR-3.1-0840 (partial)

Open issues:
• None.

Comments:
• For the purposes of this use case it has been shown that the instrument tester starts the processes. However, it is possible that some of the HCSS processes will be started automatically as part of a boot process.
• If telemetry playback is being performed [UCF-702] then it will only be necessary to start the telemetry playback selector. This will be used to send the telemetry from the HCSS archive to the EGSE router.

UCF-711 [User]: Run ILT test procedure

Level: **user**
Scope: **EGSE-ILT**
Version: **2.0**
Status: **issue**

Brief description:
This use case describes how the EGSE-ILT will interact with the HCSS to enable an instrument tester to execute a test procedure.
The HCSS will participate in this use case as a secondary actor. Only those interactions that will involve the HCSS are detailed in this use case. The detail of the interactions of the instrument tester with the EGSE-ILT are not specified.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IT: Instrument tester (primary actor)
HCSS: Herschel common science system (secondary actor)

Triggers:
Testing of an instrument is to be performed [UCF-701].

Preconditions:
Test preparation has been performed [UCF-703].
EGSE systems have been started [UCF-708].
HCSS systems have been started [UCF-709].

Minimal postconditions:
None.

Success postconditions:
The test procedure is successfully executed.

Stakeholders and interests:
TBW.

Main success scenario:
1 IT: Select instrument model
2 EGSE: Store instrument model

3 IT: Request list of test procedures
4 EGSE: Request list of test procedures for this instrument model
5 HCSS: Supply list of test procedures
6 IT: Select test procedure
7 EGSE: Request test procedure
8 HCSS: Supply test procedure
9 IT: Supply test procedure parameters
10 EGSE: Start test procedure execution
11 HCSS: Store test procedure execution time, test procedure parameters, instrument model
12 EGSE: Start logging (or pass upcoming log records immediately to HCSS)
IT/EGSE interaction without HCSS participation to execute the test procedure
13 EGSE: Request observing mode execution to be instantiated and executed
13.1 EGSE: Supply observing mode mnemonic and associated parameters
13.2 HCSS: Instantiate observing mode into observation execution
13.3 HCSS: Generate command sequence with absolute timing
13.4 EGSE: Fix absolute timing according current time
13.5 EGSE: Report back corrected absolute time for command sequence
13.6 HCSS: Fix absolute timing of building block tree for this observation
13.7 EGSE: Execute command sequence
Repeat step 13 for all observing modes encountered during test procedure execution
14 EGSE: Detects end of test procedure execution
14.1 EGSE: Pass log entries which are not transmitted yet to HCSS
14.2 HCSS: Accept log records and store them
14.3 EGSE: Add log entry with detailed information about end of test procedure
14.4 HCSS: Accept log record and store it
14.5 EGSE: Report reason for end of test procedure
14.6 HCSS: Flag testProcedureExecution with reason for end of test procedure
15 EGSE: Request saving of log
16 HCSS: Make log persistent

Extensions:
3a IT: Already knows test procedure
Continue with step 6
11-15a link to EGSE removed and no reconnection within time-out limit
HCSS: Add log record
HCSS: Save log
HCSS: Flag testProcedureExecution
12a EGSE aborts test
EGSE: Add log record
Continue with step 14
13a EGSE/IT aborts test
If there are no commands resulting from an observation request pending
it is safe to continue with step 14.
otherwise
the instrument must be commanded to „reset“ the OBSID and BBID in the TM packets to a default value (see open issues)
flag observationExecution

Continue with step 14
13b EGSE/IT aborts observation
If there are no commands resulting from an observation request pending
it is safe to continue with step 13.
otherwise
the instrument must be commanded to „reset“ the OBSID and BBID in the TM packets to a default value (see open issues)
flag observationExecution
Continue with step 13
13.2a Generation of observation execution fails
EGSE: Report error back to IT
EGSE: Add log record
IT may choose to continue test procedure or step to 13a
13.3a Command sequence generation fails
EGSE: Report error back to IT
EGSE: Add log record
IT may choose to continue test procedure or step to 13a
13.6a Fixing times of building block tree fails
EGSE: Report error back to IT
EGSE: Add log record
IT may choose to continue test procedure or step to 13a
Building block data inside the databases of the HCSS has to be fixed offline

Frequency of occurrence:
Often during ILT and IST

References:
HCSS-UR-3.1-0840 (partial)
HCSS-UR-3.1-1290
HCSS-UR-3.1-1340
FGS-IR-4.1-10
FGS-IR-4.1-20
FGS-IR-4.1-30

Open issues:
• An extension use case is needed to describe the case where the EGSE-ILT does not follow the usual sequence of events; for example, EGSE-ILT dies halfway through a test procedure and does not generate a log [**HCSS Open Issues, Issue 2.1, 23 May 2002, FIRST/FSC/DOC/0159**: Open issue 2.12.2].
• The current scenario does not allow nested procedure templates. Test procedure templates may use nested constructs as the scripting language allows, but the current model requires each procedure to be a self-contained script [**HCSS Open Issues, Issue 2.1, 23 May 2002, FIRST/FSC/DOC/0159**: Open issue 2.12.2].
• There must be a way to terminate an ongoing observation in a clean way without starting a new one. This is needed for ILT and IST but also for operations, e.g. before a time window starts which is reserved for spacecraft operations. One approach can be a defined „idle“ observation ID, which is set whenever the instrument is not executing an observation.

• HCSS must store which test procedure and which version of it was used.

Comments:
• Note: Test control can use test procedure templates retrieved from the HCSS for subsequent execution by SCOS 2000. It also permits the sending of individual telecommand mnemonics to SCOS 2000. This use case relates specifically to the use of the test control interface betweeen test control and the HCSS.

UCF-601 [User]: Perform RTA

Level: **user**
Scope: **HCSS and RTA**
Version: **2.0**
Status: **issue**

Brief description:
This use case describes the steps that will be involved in the running of RTA and its interactions with the HCSS.

Phase:
Instrument level testing: Y
Integrated system testing: Y
Ground segment testing: Y
LEOP: N
Commissioning: Y
Calibration/ PV: Y
Science demonstration: Y
Routine operations: Y
Post operations: N

Actors:
IT: Instrument tester (primary actor)
IE: Instrument engineer (primary actor)
CS: Calibration scientist (primary actor)
HCSS: Herschel common science system (secondary actor)
The generic term USR (for user) is used in the “Main success scenario” and “Extensions” sections below. When reading replace USR by IT, IE or CS as appropriate.

Triggers:
• An instrument test is to be performed (ILT/ check out) [UCF-701].
• Instrument housekeeping data needs to be re-analysed to investigate (anomalous) instrument
behaviour [UCF-702].

Preconditions:
• The relevant MIB has been made available to RTA [UCF-703] [UCF-762]
• The telemetry gateway has been started and configured [UCF-708]

Minimal postconditions:
None.

Success postconditions:
• RTA processes the telemetry packet data.
• The RTA process products are made available to the HCSS.

Stakeholders and interests:

ICC: Wants to use RTA as one of many means to analyse instrument behaviour

Main success scenario:
1 USR: Start RTA client
2 USR: Configure RTA client
3 RTA: Establish link with telemetry gateway
Telemetry packet data stream arrives via telemetry gateway
4 RTA: Receive and analyse telemetry packet data stream, generate products
5 RTA: Make products available to HCSS ingestion processes

Extensions:
3a RTA: Unable to establish link with telemetry gateway

Frequency of occurrence:
Several (many?) times per day during ILT and checkout
A few times per day during early operations
A few times per week during later operations

References:
HCSS-UR-3.1-0840 (partial)
HCSS-UR-3.1-0990
HCSS-UR-3.1-0991
HCSS-UR-3.1-0992
HCSS-UR-3.1-0993
HCSS-UR-3.1-0994
FGS-IR-3.8-10
FGS-IR-3.8-20
FGS-IR-3.8-30
FGS-IR-4.3-10
FGS-IR-4.3-20
FGS-IR-4.3-25
FGS-IR-4.3-26
FGS-IR-4.3-30
FGS-IR-4.3-35
FGS-IR-4.3-40 (Not compliant)
FGS-IR-4.3-50 (Not compliant)
FGS-IR-4.3-60 (Not compliant)
FGS-IR-4.4-10 (Not compliant)
FGS-IR-4.4-20 (Not compliant)
FGS-IR-4.4-30 (Not compliant)
FGS-IR-4.4-40
FGS-IR-4.4-50

Open issues:
• Is saving of results/logs compulsory, and how are RTA results linked into the system when the same data might be analyzed many times by independant RTA sessions, possibly with different configurations, and possibly running simultaneously [**HCSS Open Issues, Issue 2.1, 23 May 2002, FIRST/FSC/DOC/0159**: Open issue 2.10.9]?

Comments:

• The telemetry gateway specifies which telemetry packets are wanted (from the telemetry router). It forwards these packets to the RTA system (see UCF-701, UCF-702).

• If RTA is to be used for analysis of near real-time (uncosolidated) telemetry outside of the context of MOC, the link between MOC and ICC is not provided by HCSS.

• If RTA is running at MOC and it is required to ingest RTA results (logs, events, etc.) into HCSS, a link has to be provided.

• Is there a need for an 'automatic' running of RTA (eg. in a quality control process at the ICCs)? Running RTA automatically can be covered by proper procedures surrounding the current use case. Nothing in the use case makes it special.

• The interface between the HCSS and RTA is described in a set of ICDs (ICD-7, ICD-8 and ICD-12) [**HCSS Open Issues, Issue 2.1, 23 May 2002, FIRST/FSC/DOC/0159**: Open issue 2.7.3].

• The products currently identified for ingestion into the HCSS [UCF-701] are the telecommand history and the out of limits history.

UCF-747 [User]: Perform QLA

Level: **user**
Scope: **HCSS**
Version: **2.0**
Status: **issue**

Brief description:
This use case describes the running of QLA in either playback mode or NRT mode and its interactions with the HCSS archive. It does not identify the specific details of QLA processing. A distinction is made below between HCSS QLA and HCSS access. HCSS access is an API used by QLA in order to access the data to be analysed.

Phase:
Instrument level testing: Y
Integrated system testing: Y
Ground segment testing: Y
LEOP: Y
Commissioning: Y
Calibration/ PV: Y
Science demonstration: Y
Routine operations: Y
Post operations: N

Actors:
IT: Instrument Tester (primary actor)
IE: Instrument Engineer (primary actor)
CS: Calibration Scientist (primary actor)
The generic term USR (for user) is used in the “Main success scenario” and “Extensions” sections below. When reading replace USR by IT, IE or CS as appropriate.

Triggers:
ILT: Instrument testing is being performed [UCF-701].
ILT: ILT offline analysis is to be performed [UCF-706].
IST to routine operations: Instrument science data needs to be re-analysed to investigate
(anomalous) instrument behaviour.

Preconditions:
ILT: Test preparation has been performed for NRT [UCF-703].
ILT: The required playback test data is in the archive [UCF-701].

Minimal postconditions:
None.

Success postconditions:
QLA processes the data.

QLA process products, if any, are made persistent in HCSS (TBC).

Stakeholders and interests:
ICC: Want to use QLA as one of many means to analyse instrument behaviour

Main success scenario:
1 USR: Start QLA
2 USR: Configure QLA for playback
2.1 USR: Select playback mode
2.2 USR: Select database from which data is to be retrieved
2.3 USR: Select telemetry packets or data frame data
2.4 USR: Specify data selection criteria (see comments)
2.5 USR: Select data playback rate
3 USR: Request connection
4 HCSS access: Establish connection with identified data source (see comments)
5 HCSS access: Get data from database and pass to QLA
6 USR: Interact with QLA to analyse data **(SPIRE ICC use case UC-QLA101)**
7 HCSS QLA: Perform requested analysis steps **(SPIRE ICC use case UC-QLA101)**
Steps 5 to 7 can be performed in parallel
Repeat steps 2 to 7 until analysis complete

Extensions:
2a: USR: Configure QLA for NRT connection
2a1: USR: Select NRT mode
2a2: USR: Select telemetry packets or data frames
6a: USR: Request automatic analysis
7a: HCSS QLA: Perform automatic analysis on the data
8: HCSS QLA: Automatically generate QLA report and store in archive (TBC)
8a: USR: Create QLA report and request storage in the archive (TBC).

Frequency of occurrence:
For ILT the processing is performed at least once for every observation execution
A few times per day during early operations
A few times per week during later operations

References:
HCSS-UR-3.1-0840 (partial)
HCSS-UR-3.1-0990
HCSS-UR-3.1-0991
HCSS-UR-3.1-0992
HCSS-UR-3.1-0993
HCSS-UR-3.1-0994

Open issues:
• None.

Comments:
• In playback mode it will be possible to request the retrieval of data based on:

• Observation identifier
• Time period
• Connection to data source:
• In NRT mode this will be the EGSE-ILT router for the ILT mission phase. TBC for the subsequent mission phases.
• In playback mode this will be the identified database.
• Details of any QLA reporting and its subsequent storage in the archive are still TBC.
PACS currently state that QLA will produce a printed summary report.

UCF-706 [Summary]: ILT offline analysis

Level: **summary**
Scope: **EGSE-ILT and HCSS**
Version: **1.0**
Status: **issue**

Brief description:
This use case describes the steps that will be performed during offline analysis.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IT: Instrument tester (primary actor)

Triggers:
It is required to perform offline analysis of instrument test data [UCF-700].

Preconditions:
The required test data is in the archive [UCF-701].

Minimal postconditions:
None.

Success postconditions:
The instrument test data is successfully analysed.

Stakeholders and interests:
TBW.

Main success scenario:
1 IE: Start telemetry and data frame server (optional)
2 IE: Run QLA in playback mode [UCF-747]
3 IE: Run interactive analysis

Extensions:
None.

Frequency of occurrence:
Often during ILT

References:
TBD

Open issues:
• None.

Comments:
• None.

UCF-702 [Summary]: Replay ILT test telemetry

Level: **summary**
Scope: **EGSE-ILT and HCSS**
Version: **2.0**
Status: **issue**

Brief description:
This summary level use case describes the steps that will be performed for instrument telemetry playback (replay) during ILT.

Phase:
Instrument level testing: Y
Integrated system testing: N
Ground segment testing: N
LEOP: N
Commissioning: N
Calibration/ PV: N
Science demonstration: N
Routine operations: N
Post operations: N

Actors:
IT: Instrument tester (primary actor)
EGSE-ILT: The electrical ground segment equipment system for ILT.

Triggers:
It is necessary to replay ILT telemetry [UCF-700].

Preconditions:
Test preparation has been performed [UCF-703].

Minimal postconditions:
None.

Success postconditions:
The telemetry playback is successful.

Stakeholders and interests:
TBW.

Main success scenario:
1 IT: Start EGSE [UCF-708]
2 IT: Start HCSS [UCF-709]
3 IT: Start playback selector and identify playback telemetry (HCSS) [UCF-705]
4 Router: Register playback selector (EGSE-ILT)
5 Playback selector: Retrieve TM from database and forward to router (HCSS) [UCF-704]
6 Router: Receive telemetry packet and forward to registered clients (EGSE-ILT)

7 RTA: Receive telemetry and process it (EGSE-ILT) [UCF-601]
Repeat steps 5 to 7 until telemetry playback is complete

Extensions:
None.

Frequency of occurrence:
Often during ILT

References:
TBD

Open issues:
• None.

Comments:
• None.

UC-QLA101: Use QLA

Level: **user**
Scope: **SPIRE ICC**
Version: **0.2**
Status: **draft**
Date: **17 June, 2002**

Brief description:
This is the top ICC use case for using QLA and corresponds to steps 6 and 7 of UCF-747, 'Interact with QLA to analyse data' and 'Perform requested analysis steps'.

Phase:
ILT-OPS

Actors:
TS: Test Scientist

Triggers:
Test data needs to be displayed and analysed.

Preconditions:
QLA has been started and configured (UCF 747 steps 1-5)

Minimal post-conditions:
The database is not corrupted.

Success post-conditions:
The test data is successfully analysed

Stakeholders and interests:

Main Success Scenario:

1. TS: Select detector(s) to display **(UC-QLA103**)
2. TS: Select parameters to display **(UC-QLA104)**
3. TS: Select displays while the test is running
4. TS: De-select displays during the test.
5. TS: Change speed of data stream (only in playback mode)
6. QLA: Generate processed data from the test **(UC-QLA105)**.
7. QLA: Display processed data **(UC-QLA106)**
8. QLA/TS: Store test results **(UC-QLA107)**
9. TS: Compare with previous results**.(UC-QLA108)**

Extensions:

References:
ILT-PERF-OPI – Telescope Pupil Ilumination
ILT-PERF-OPB – Balancing of Ports
ILT-PERF-OSL – Pixel Spatial Information Using the Laser
ILT-PERF-OSB – Pixel Spatial Information Using the External Black Body
ILT-PERF-OPB – Point Spread Function Using the BSM
ILT-PERF-DFT – Operating Temperature Range
ILT-PERF-DAN – Amplifier Noise
ILT-PERF-DND – Detector Noise with Bias (DC Bias)
ILT-PERF-DST - Detector Noise With Sink Temperature (DC Bias)
ILT-PERF-DOT - Bolometer Operating Temperature Characterisation
ILT-PERF-DNA - Detector Noise (AC Bias)
ILT-PERF-DDB - Blanked Load Curves (DC Bias)
ILT-PERF-DAB - Blanked Load Curves (AC Bias)
ILT-PERF-DDL - Optical Load Curves (DC Bias)
ILT-PERF-DAL - Optical Load Curves (AC bias)
ILT-PERF-DMD - Mechanism Microphonics (DC Bias)
ILT-PERF-DMA - Mechanism Microphonics (AC Bias)
ILT-PERF-DDM - Microphonic Susceptibility (DC Bias)
ILT-PERF-DAM - Microphonic Susceptibility (AC Bias)
ILT-PERF-DRL - Detector Relative Response vs Input Power Using the Laser
ILT-PERF-DRB - Detector Relative Response vs Input Power Using the External Black Body
ILT-PERF-DSR - Spectral Response
ILT-PERF-CPC - Photometer Calibrator Characterisation
ILT-PERF-CSC - Spectrometer Calibrator Characterisation
ILT-PERF-CSR - Room temperature nulling
ILT-PERF-CST – Spectrometer Calibrators Performance with Time
ILT-PERF-SZP - ZPD Position
ILT-PERF-SFC - Fringe Contrast and Spectral Response While Scanning
ILT-PERF-SFL - Fringe Contrast and Spectral Response Step and Look
ILT-PERF-SMC - Mirror Carriage Characterisation While Scanning
ILT-PERF-SML- Mirror Carriage Characterisation Step and Look
ILT-PERF-BSM - BSM Characterisation
ILT-PERF-BCT- BSM Chop Throw
ILT-PERF-OBL - Out of Band Radiation Using a Set of Laser Lines
ILT-PERF-OBS - Out of Band Radiation Using the SMEC
ILT-PERF-OBE - Out of Band Radiation Using an Edge Filter

Open Issues:

Comments:
The MSS here has the TS as the actor; however we could also envisage these steps being done by a test script.

The test start is likely to be indicated in an event packet. This event will trigger QLA to start test specific support.

Before the test start there is no requirement for QLA to remember what data has come in. Between the test start and test end, all the data must be available to QLA for final processing and output.

The references are based on SPIRE-RAL-DOC-001123 draft 0.4, these may change with later drafts.

Related work packages:
GHS13X5000 Summary WP for QLA
GHS13X5100 QLA for AVM
GHS13X5200 QLA for CQM
GHS13X5300 QLA for PFM
GHS13X5400 QLA for operations
GHS13X5130 QLA framework
GHS13X5110 QLA requirements
GHS13X5120 QLA system level analysis and design

UC-QLA102: Perform offline analysis of test results

Level: **user**
Scope: **ICC**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:
This use case is a step after UC-QLA001 and is not covered by UCF-747

Phase:
ILT-OPS

Actors:
TS: Test Scientist

Triggers:

Preconditions:

Minimal post-conditions:
The database is not corrupted.

Success post-conditions:

Stakeholders and interests:
Instrument Engineer
Calibration Scientist

Main Success Scenario:

1. TS: requests data
2. QLA: checks authorisation
3. QLA: searches database
4. QLA: displays search results
5. TS: selects data required
6. TS: selects output format
7. QLA: writes data to disk
8. TS: analyses data
9. TS: starts database interface tool
10. TS: supplies details of data to import
11. QLA: imports data

Extensions:

2a Authorisation fails
2a1: Notify user of reason for failure

References:

Open Issues:

Comments:
Could branch into the analysis needed for each test type.
QLA can download data over the network.

Related work packages:
GHS13X5140 QLA User Interface

UC-QLA103: Select and display detectors

Level: **user**
Scope: **QLA**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:
QLA user or script selects and displays detectors

Phase:
ILT-OPS

Actors:
TS: Test Scientist

Triggers:
A QLA user wishes to select detectors for display

Preconditions:
QLA is running

Minimal post-conditions:
QLA does not crash

Success post-conditions:
Detectors are selected

Stakeholders and interests:

Main Success Scenario:

1. TS: Starts detector selector process
2. TS: Selects how the detector data will be displayed
3. TS: Selects array
4. TS: Selects detectors
5. TS: Presses 'finished' button
6. QLA: Stores information about what has been selected and how it will be displayed
7. QLA: Displays selected parameters

Extensions:

1a. QLA Script tells detector selector process what detectors are required and how they should be displayed.
1a1. QLA: Main success scenario steps 6 and 7 are executed.

5a TS: Quits process

5b TS: Selects another array, main success scenario proceeds from step 3.

References:

Open Issues:

Comments:
This use case refers to a choice of timeline plot, scrolling list or an RTA-like display. There will be a default image display that will appear when QLA is started i.e. as part of the start-up GUIs plus possible other image displays to support specific tests e.g. peakup.

Related work packages:
GHS13X5170 Detector selection and display
GHS13X5170 Parameter display, packet display and time series plotting

UC-QLA104: Select and display parameters

Level: **user**
Scope: **QLA**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:
QLA user or script selects and displays parameters

Phase:
ILT-OPS

Actors:
TS: Test Scientist

Triggers:
A parameter needs to be monitored

Preconditions:
QLA is running

Minimal post-conditions:

Success post-conditions:
Parameter and display types are successfully selected.

Stakeholders and interests:

Main Success Scenario:

1. TS: Starts parameter selector process
2. TS: Selects how the parameter data will be displayed
3. TS: Selects parameters to display
4. TS: Presses 'finished' button
5. QLA: Stores information about what has been selected and how it will be displayed
6. QLA: Displays selected parameters

Extensions:

1a. QLA Script tells parameter selector process what parameters are required and how they should be displayed.
1a1. QLA: Main success scenario steps 5 and 6 are executed.

References:

Open Issues:

Comments:

Related work packages:
GHS13X5170 Parameter display, packet display and time series plotting

UC-QLA105: Generate processed data for the test

Level: **user**
Scope: **QLA**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:
QLA generates processed data for the specific test. This covers UCF-747 step 7 'Perform requested analysis steps'.

Phase:
ILT-OPS

Actors:

Triggers:
An event packet arrives with information that the test has completed.

Preconditions:
A test specific support process is running.

Minimal post-conditions:

Success post-conditions:
The data are processed correctly.

Stakeholders and interests:
Test Scientist

Main Success Scenario:

1. QLA: Checks which test support process is active
2. QLA: Checks which data needs to be compiled for that display type.
3. QLA: Checks all data for the test period is present
4. QLA: Checks all data has been processed to the correct level
5. QLA: Compiles data into appropriate data cube

Extensions:

References:

Open Issues:

Comments:
Here it is assumed that a test-specific support process will be responsible for processing and compiling a dataset. The test specific support processes are

naturally defined by the test procedures (referenced in UC-QLA001) and will be specified in detail elsewhere.
Note the word ‘analysis’ in step 7 of UCF-747 is the wrong use of the word, next time we comment on UCF-747 we should request that it gets changes to ‘process’.

Related work packages:
GHS13X5200 QLA for CQM

UC-QLA106: Display processed data

Level: **user**
Scope: **QLA**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:
Display processed data

Phase:
ILT-OPS

Actors:

Triggers:
A QLA process has data ready for display.

Preconditions:

Minimal post-conditions:
QLA does not crash.

Success post-conditions:
The processed data is successfully displayed

Stakeholders and interests:
Test Scientist

Main Success Scenario:

1. QLA: Checks which test is being done or which displays are active (see comments).
2. QLA: Check all required data is present
3. QLA: Display current/final results
4. QLA: If test has ended prompts user to ask if data is to be stored

Extensions:

2a: QLA: If it is not present retrieve it from memory/database wherever!
2a1: QLA: Apply appropriate processing.

References:

Open Issues:

Comments:

Related work packages:
GHS13X5200 QLA for CQM

UC-QLA107: Store test results

Level: **user**
Scope: **QLA**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:

Phase:
ILT-OPS

Actors:

Triggers:
An script or user interaction following a QLA prompt

Preconditions:

Minimal post-conditions:
The database is not corrupted.

Success post-conditions:

Stakeholders and interests:
Test Scientist
Calibration Scientist
Instrument Engineer

Main Success Scenario:

1. QLA: prompts the user to enter storage requirements
2. QLA: stores the data into the database
3. QLA: outputs data into a file
4. QLA: informs user of the successful storage

Extensions:

1a. A script step could determine whether and how the reduced data is stored.

References:

Open Issues:

Comments:

Related work packages:

GHS13X5140 QLA User Interface

UC-QLA108: Compare with previous results

Level: **user**
Scope: **QLA**
Version: **0.3**
Status: **draft**
Date: **04 July, 2002**

Brief description:
QLA user compares current test results with previous results retrieved from the database by QLA.

Phase:
ILT-OPS

Actors:
TS: Test Scientist

Triggers:
A script step or user interaction

Preconditions:
The data is available

Minimal post-conditions:
The database is not corrupted.

Success post-conditions:

Stakeholders and interests:

Main Success Scenario:

1. QLA starts database selection process
2. TS: selects data
3. TS: selects how data will be displayed
4. QLA: retrieves data
5. QLA: displays data

Extensions:

References:

Open Issues:

Comments:
Related work packages:
GHS13X5140 QLA User Interface