FIRST

DPU/ICU On Board Software

Product Assurance Plan

Document Ref.:

Issue: Draft 3

Prepared by:	Anna Maria Di Giorgio	Date:	09/10/2000
Checked by:	Riccardo Cerulli-Irelli		
Approved by:	Renato Orfei		

Document Status Sheet:

Document Title: DPU/ICU On Board Software PA Plan			
Issue	**Revision**	**Date**	**Reason for Change**
Draft 1		12 Jan 2000	First draft
Draft 2		08 May 2000	Second draft
Draft 3		06 Oct 2000	

Document Change Record:

No change record is included for the changes in the draft versions.

Document Title: DPU/ICU On Board Software PA Plan	
Document Reference Number:	
Document Issue/Revision Number: Draft 2	
Section	**Reason For Change**

1 Introduction

This document specifies the product assurance organisation and program plan for the DPU-ICU On Board Software for the three instruments onboard the ESA FIRST/PLANCK mission.
The definition of on board software adopted in this document includes various items related to the generation of an executable program:
- Source code
- Executable code
- Data files
- Documentation

All the tools and the interfaces' simulators developed for the testing activities shall be considered as an integral part of the on board software as well.

1.1 Purpose of the document

This plan intends to define the functions and procedures used to establish the software quality assurance for all the DPU/ICU on board software development life-cycle. It is written to ensure that the software will be developed according to the ESA standards.
As suggested in the Guide to Applying the ESA software Engineering Standards to Small Software Projects (ESA BSSC (96) 2, Issue 1), this document will include the Software Project Management Plan, the Software Configuration Management Plan and the Software Quality Assurance Plan.

1.2 Acronyms and Abbreviations

1.2.1 Acronyms

AD	Architectural Design
ASCII	American Standard Code for Information Interchange
ATP	Acceptance Test Plan
AVM	Avionic Model
CASE	Computer Aided Software Engineering
CDMS	Command and Data Management System
CNR	Consiglio Nazionale delle Ricerche
CPU	Control Processing Unit
DDD	Detailed Design Document
DPU	Digital Processing Unit
EEPROM	Electrically Erasable Programmable Read Only Memory
EGSE	Electronic Ground Support Equipment
ESA	European Space Agency
FIRST	Far InfraRed and Submillimeter Telescope
HIFI	Heterodyne Instrument for FIRST
HK	HouseKeeping
HW	HardWare
IBDR	Instrument Baseline Design Review
ICD	Interface Control Document

ICDR	Instrument Critical Design Review
ICU	Instrument Control Unit
IHDR	Instrument Hardware Design Review
IFSI	Istituto di Fisica dello Spazio Interplanetario
ISVR	Instrument Science Verification Review
NCR	Non Conformance report
OBS	On-Board Software
PACS	Photoconductor Array Camera and Spectrometer
PA	Product Assurance
PDR	Preliminary Design Review
QA	Quality Assurance
RID	Review Item Discrepancy
SCCS	Source Code Control System
SCR	Software Change Request
SPIRE	Spectral and Photometric Imaging REceiver
SPR	Software Problem Report
SPU	Signal Processing Unit
SR	Software Requirement
SSD	Software Specification Document
SVVP	Software Verification and Validation Plan
SW	SoftWare
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
UR	User Requirement
URD	UR Document
WP	Work Package

1.3 References

1.3.1 Applicable Documents

Document Reference	Name	Number/version/date
AD1	PA Requirements for FIRST/Planck Scientific Instruments	PT-RQ-04410 Issue 1 September 1997
AD2	Guide to Applying the ESA Software Engineering Standards to Small Software projects	ESA BSSC (96) 2 Issue 1
AD3	Product Assurance Plan for the FIRST HIFI Instrument	SRON-U/HIFI/PL/1999-008 Issue 2, 18-02-2000
AD4	SPIRE PA Plan	Draft 1, 05-02-1998
AD5	PACS Project PA Plan	Draft, 26-11-1999
AD6	Product Assurance Plan for the FIRST DPU-ICU Subsystem	Draft2, 02-02-2000
AD7	FIRST DPU/ICU Subsystem Development Plan	Issue 1, 13-06-2000

1.3.2 Reference Documents

Document Reference	Name	Number/version
RD1	The C (ANSI C) Programming language. (Kernighan, B.W., Ritchie, D.M.)	1989, Prentice Hall PTR
RD2	Strategies for real-Time System Specification (D. J. Hatley, I. A. Pirbhai)	1988, Dorset House Publishing
RD3	HIFI Non Conformance Control Procedure	SRON-U/HIFI/PR/1999-007 Issue 1 16-05-2000
RD4	FIRST/Planck Instrument Reviews	SCI-PT/FIN-06692 Issue 2.0 26-03-2000

1.4 Overview of the document

TBW

2 Management

The IFSI management structure is shown in the organisation chart given in Figure 2-1: it is the same for the three Consortia. The IFSI local Project manager is responsible for both the DPU/ICU HW and SW development. As far as the OBS is concerned, his tasks are:
- to define the development schedule necessary to meet the Consortia milestones;
- to manage and monitor the activities of the development team;
- to manage technical and schedule risks.

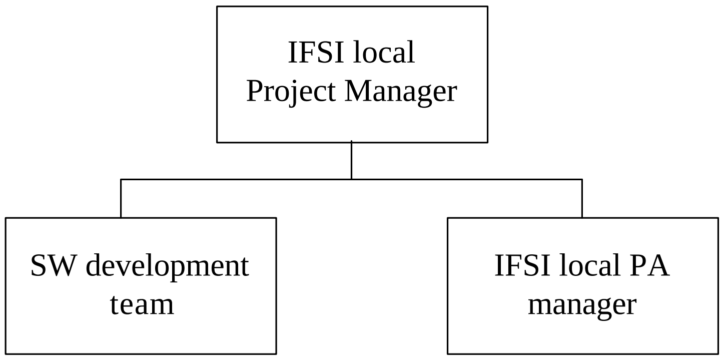

Figure 2-1

The SW development team is responsible for providing all the software products (from AVM release to Flight model release) and documentation. The same team will carry out all the IFSI internal testing activities (see sections 4.3) and will assist all the instrument integration activities involving software use and testing.
The PA manager is responsible for the implementation of the PA activities related to the project. He is located at IFSI and reports directly to the FIRST Instruments PA Project managers.
The PA manager has in charge the overall configuration management of the DPU/ICU OBS ensuring that the software configuration status list is prepared and maintained.

2.1 Software life cycle

The project life cycle represents all the activities related to the design, development, operations and maintenance of the software. It is therefore structured as a model which organises the project activities into phases and defines what activity occurs in which phase.
Since the products of a software development project will be delivered in a timely manner, it is possible to provide a planning of the SW development phases with an indication, for each of them, of all the activities to be performed and all the items to be delivered.
According to ESA Standards for Small Software Projects (ESA BSSC (96) 2, Issue 1), the life cycle of the DPU/ICU OBS development is assumed to be divided into the following phases (see Figure 2-2):
- user requirements definition
- software requirements definition and architectural design

- software detailed design and coding
- software testing and delivery

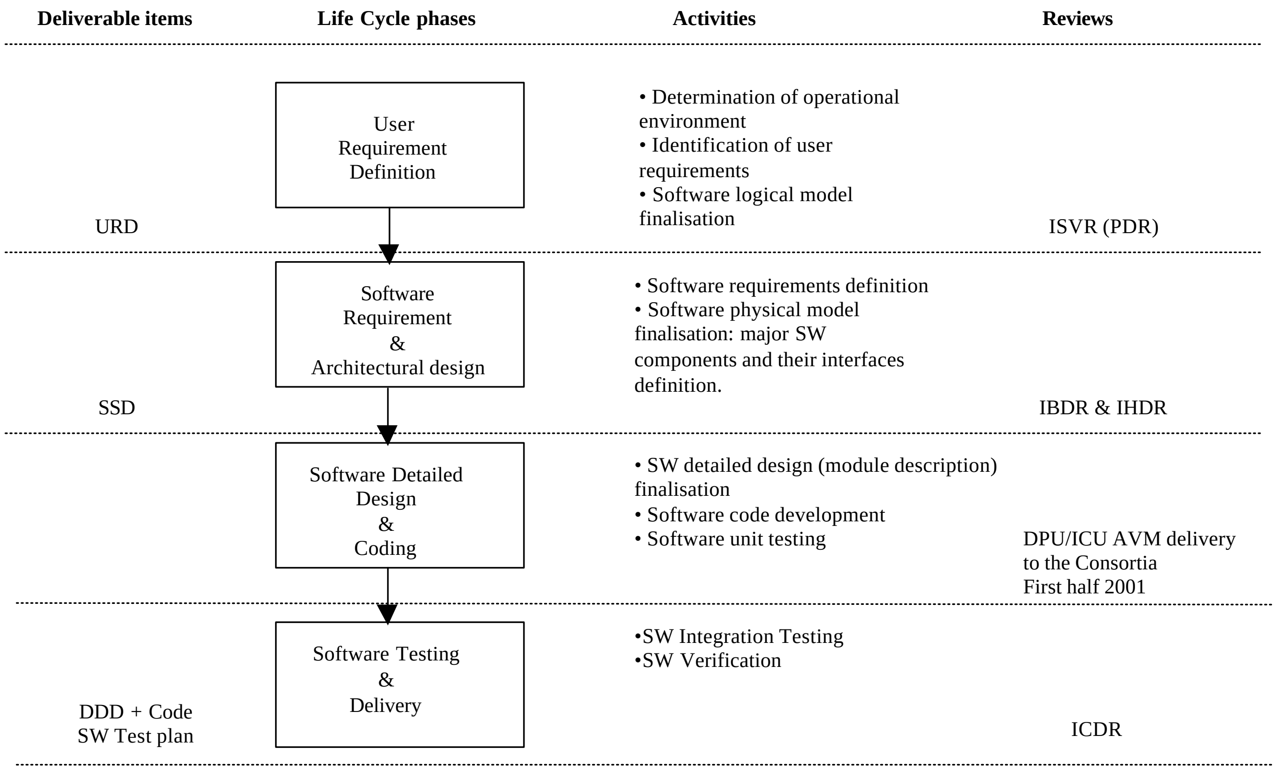
Deliverable items
Life Cycle phases
Activities
Reviews
User Requirement Definition
• Determination of operational environment
• Identification of user requirements
• Software logical model finalisation
URD
ISVR (PDR)
Software Requirement & Architectural design
• Software requirements definition
• Software physical model finalisation: major SW components and their interfaces definition.
SSD
IBDR & IHDR
Software Detailed Design & Coding
• SW detailed design (module description) finalisation
• Software code development
• Software unit testing
DPU/ICU AVM delivery to the Consortia First half 2001
Software Testing & Delivery
•SW Integration Testing
•SW Verification
DDD + Code
SW Test plan
ICDR

Figure 2-2 Life cycle of the DPU/ICU OBS development

2.2 Tasks and activities.

The present PA Plan is referred to all DPU/ICU OBS development activities described in the development plan reported in Appendix C. In the following the activities belonging to each development phase are briefly described.

2.2.1 User requirements definition phase

The user requirements shall be defined in close co-operation with the three Instruments' teams and the URD document shall be written. An implementation-independent model of what is needed by the user (SW logical model) shall be provided.
The PA activities in this phase will be to ensure the final integrity and completeness of the URD and to check its compliance with the applicable documents and with the methods and standards defined in the applicable documents.

2.2.2 SR and AD phase

The SR shall be defined and the SW Architectural Design shall be finalised. The SW Specification Document will be provided (including the SW architecture and describing the method adopted for the design, see AD2) and the traceability of the AD components versus the SR will be checked. The traceability of the SR versus the UR will be checked as well: for each item of the SW requirements list, the reference UR will be indicated.
The PA activities in this phase will be to ensure the final integrity and completeness of the SSD, to check its compliance with the applicable documents and to ensure the full mapping of the requirements on the design components.

2.2.3 Detailed Design and coding phase

The SW code shall be developed according to the standard described in section 4.2. All the unit tests shall be carried out, to check the single module interfaces. The DDD document will be provided containing mainly a commented copy of the source code.
The PA activity in this phase will be to perform all the necessary software configuration management tasks: all deliverable code will be identified in a configuration item list. No code inspection activities will be done by the IFSI personnel: the code reviewing will be performed by the Consortia in dedicated OBS reviews, see section 5.

2.2.4 Software Testing and Delivery phase

All the OBS integration tests will be performed according to IFSI internal verification plans. The System testing activities will be carried out according to the verification plans, see section 4.3.3 . When all the IFSI internal testing phases are passed, the OBS will be delivered to the Consortia for the acceptance testing phase. At this stage the software will be ready to be programmed into PROMs and installed on the final target system (i.e. EEPROMs).
The PA activities in this phase will be to perform all the necessary software configuration management activities (create and maintain the history logs for software changes, check the completeness of changes, ensure the traceability of all the modifications) and to participate in all the test phases (creating and maintaining the logs for the test reports, the SPRs and the NCRs).

2.2.5 OBS Maintenance phase

Two main deliveries are foreseen during the lifetime of the project: the AVM delivery and the PFM delivery. The bulk of the DPU/ICU OBS will be delivered at the AVM review. The PFM package will be mainly an upgrade of the AVM package, with the modifications of the code requested by changes in the OBS specifications and with new modules included only if the corresponding specification were not available at the AVM delivery.
Therefore, it can be said that the OBS maintenance starts after the AVM delivery.
The maintenance activities between the AVM review and the PFM delivery will be performed by IFSI using the IFSI development environment to provide the new images of the upgraded code. No packetisation of the new images will be performed: it is expected that this will be one of the services provided by the SCOS 2000 package.
All the configuration management/document upgrade/testing activities on the new code will be done in this phase by IFSI. This approach is considered as applicable to all the Subsystem tests phases.

After the PFM delivery, OBS Maintenance and the development of the OBS Maintenance Facility will be part of the IFSI contribution to the ICCs.
A first draft of the DPU/ICU On Board Software User Manual will be provided by IFSI to the Consortia at the AVM delivery. The final version of the Manual will be part of the PFM delivery packet.

2.3 Product assurance relationships

Given the organisation reported in Figure 2-1, in case of disagreement between the PA manager and the SW development team, the problem will be reported to the Project manager for a decision.

No IFSI internal quality inspection audits are foreseen. External audits can be arranged, see Section 5.

3 Documentation

The following documents for the DPU/ICU OBS will de delivered:

- DPU/ICU OBS User requirement Document: common to the three consortia, with a section for each instrument dedicated to the non common user requirements;
- DPU/ICU OBS Software Specification Document: one per each Instrument;
- DPU/ICU OBS Detailed Design Document: one per each Instrument;
- DPU/ICU OBS Product Assurance Plan (this document): common to the three consortia
- DPU/ICU OBS Software Verification and Validation Plan: one per each Instrument.

The reference codes of the documents shall be in accordance with the document management indications provided by the three instruments (see section 9).
Non deliverable documents will be written for IFSI internal use, e.g. technical notes. These documents will not be under configuration control.

4 Standards, Practices, conventions and metrics

4.1 Documentation standards

The structure and contents of all documents will be compliant with the indications in AD2.
Each document will contain header pages showing approval and status of the document.
The final format of the documents will be MSWord 97 and , if desired, Acrobat PDF.
After a preliminary circulation in a draft form, for gathering all the main comments, the documents will be delivered to the consortia according to the three following procedures:

1. SPIRE: the documents will always be delivered to the ESA DMS domain responsible, who will put them in the DMS.

2. PACS: the documents will be put in the PACS internal web dedicated page.

3. HIFI: the document will be put in the HIFI internal web dedicated page.

4.2 Design and coding standards

Design standards: The Structured Analysis method will be adopted, with the Hatley-Phirbai Real-Time extensions for architectural designing, see RD2. The adopted method and formalism will be described in the SSD.

Coding standards: The software will be written in ANSI C (see RD1), avoiding to use non-standard features. If any non standard features are needed for technical reasons this will be documented. In the final documentation ALL use of non-ANSI features will be documented.

IFSI internal conventions will be:
- Max number of code lines in one file (module): 1000.
- Max number of code lines in a function: 50.
- For each source file a dedicated header file will be provided including constants and variable declarations, type definitions, function declarations, defines etc., for the usage inside the corresponding module. A separate include file containing all the prototypes of the functions that will be needed publicly will be provided, as well as an additional global header file with all the definitions and declarations that will be made accessible from all the modules.
- All private definitions and declarations shall be put at the beginning of the module/function.
- The GOTO construct shall not be used
- Avoid to use dynamic memory allocation: every use shall be documented and in any case, the dynamically allocated memory blocks must be explicitly deallocated after use.
- Avoid the use of global variables (this use can be justified for software speed purposes, in these cases it shall be clearly explained).
- Every system call will be checked for an error return and the system error text will be included in the error message log. The overall error handling strategy will be described in the SSD (AD section).
- Pointer variables to structures must be explicitly deallocated after use.

- Each module shall start with a header, showing the title, a brief description, creation date and last update, listing and describing the functions in the module. In the following, some examples of headers are reported :

    1. Example of module header:

```
/****************************************************************************
* File name:
* Version.Revision:
*
* Purpose:
*
* Public Functions:
*
* Private Functions:
*
* Description:
*
* Creation date & author: dd-mm-yyyy                XXX
* Version, Update date & author:     nn.mm          dd-mm-yyyy          XXX
****************************************************************************/
```

2. Example of function header

```
/****************************************************************************
* Function name:
*
*  Purpose:
*
* Description:
*
* Syntax:
*
* Input:
*
* Output:
*
* Return:
*
* Files Accessed:
*
* Version, Creation date & author:    nn.mm          dd-mm-yyyy          XXX
*
* Version, Update date & author:      nn.mm          dd-mm-yyyy          XXX
* Update changes:
*
****************************************************************************/
```

3. Example of include file header

```
/*****************************************************************************
* File name:
*
* Public Functions:
*
* Private Functions:
*
* Description:
*
* Short Description:
*
* Creation date & author: dd-mm-yyyy              XXX
* Version, Update date & author:    nn.mm         dd-mm-yyyy          XXX
*****************************************************************************/
```

A naming convention to be adopted in the OBS code will be included in the design specifications.

4.2.1 Change logs and version control

A SW system undergoes many changes in its lifetime. Revision control can be viewed as *change management*: it allows the developers a degree of control over what changes are made to the system, and allows them to find information about previous changes to the system. In other words, it avoids users making changes to collide with each other.
All the DPU/ICU OBS software deliverables will be under revision control as well as all the provided documentation, after the first Issue.
The software changes can be due either to non conformances or to software problems detected during tests. The procedures for rising the problems and changing the software accordingly are described in section 7.

The version control tool that will be used by IFSI is the **Cyclic Software CVS (Concurrent Version System)** freeware. The tool works by keeping a single copy of all the source code (and of all the other deliverables) and putting it into a database of the change history of the developed system: CVS maintains a history of all changes made to each directory tree it manages. Using this history, it will be possible to recreate past states of the tree, knowing when, why and by whom a given change was made.
CVS can be set up as a client/server system: the version history is stored on a single central server and the client machines will have a copy of all the files that the developers are working on.
Finally, the selected tool supports parallel development, allowing more than one developer to work on the same sources at the same time and managing the concurrent changes.
A detailed report on the history of the project can be obtained easily and included in the package documentation provided with each delivery of the code.

4.3 Testing standards

In the following, the definitions in AD2 and AD5 are adopted for the different testing phases.

4.3.1 Unit tests

A 'unit' of software is here intended to be a module/file: the inputs to unit testing are the successfully compiled modules from the coding process. These are assembled during unit testing to make the largest units, i.e. the components of architectural design. Tests will be performed to check both the internal logic of the software and the overall software functionality.
IFSI will not provide any unit test documentation (test plan, test input and output data, test reporting forms).

4.3.2 Integration tests

With Integration testing it is here intended the process of testing the developed software against the architectural design: the already unit tested components are integrated to make the system. The tests will be designed to verify the data and control flows across the interfaces between the AD components. Performance tests will be included as well.
Integration tests can be performed on an emulation system.
The integration tests plan will be provided as a draft before the IFSI on site integration. The first issue of the document will be essentially composed of a complete set of the tests reports and will be delivered with the software.

4.3.3 Acceptance tests

The acceptance testing activities are referred to the process of testing the system against the software (and user) requirements. The input to these tests is the successfully integrated SW.
The tests will be designed to verify each functional requirement and each performance requirement to the system. A set of tests will be dedicated to the system interface testing, using the ICDs as the baseline of the acceptance. For these purposes, simulators of the interfaces of the DPU/ICU OBS with the spacecraft and the other instruments' subsystems will be developed and used during instrument integration. These simulators will be part of the DPU/ICU OBS maintenance facility, as well as all the analysing tools developed by IFSI and necessary to put in a readable form the data flows to be controlled and to examine the function and performance of the OBS.
The acceptance tests will be performed at the three consortia PIs premises, on the real systems (avionic models/simulators). It is assumed that the EGSEs will be available for the testing activities.

The SVVP will be provided as a separate document, with to be circulated as a first draft at the end of the SR & AD phases. The first Issue of the SVVP is intended to be ready at the beginning of the AVM acceptance tests (see AD7, section 6.8).

5 Reviews and audits

No internal reviews and audits for pure quality assurance purposes will be performed. During the overall software design and development life cycle, the IFSI SW development team will have periodic internal technical reviews for the purposes of reporting the work package status/problems to both the PM and the PA manager.
Since the code inspection activity is considered as essential by the Consortia, some dedicated "external" audits will be organised with members of the Consortia, in order to review the code before the official delivery. These reviews will be arranged either at IFSI premises or via internet exchange of the relevant documentation. The quality activity will be performed by the Consortia.
The proposed reviews are (see AD7, section 9, for information about the Reviews scheduling):

- OBS Intermediate Review;
- OBS 2nd Review.

Two reviews are foreseen for each DPU/ICU acceptance testing (see AD6, sect. 8.4.3):

1. A Test Readiness Review, before the acceptance tests activity start up. This review will have the purpose of checking the following issues:
   - Is the DPU/ICU (HW and SW) ready for testing?
   - Is the test program and its related facilities (test plans and procedures) ready to perform the test?

   The review will be held at IFSI premises. The first formal review is the TRR of the AVM.
2. A Delivery Review, at the end of the acceptance tests. A delivery is a transfer of responsibilities between two parties, therefore this review will have the purpose of checking the following issues:
   - Has the DPU/ICU passed successfully the planned test program?
   - Are there unresolved problems preventing further integration?

   The review will be held at IFSI premises.

The Test Readiness Review Board (TRRB) and the Delivery Review Board (DRB) will include the Software Review Board IFSI representatives (see section 7).

GO-ahead for prom Burn-in will be given after an acceptance review between IFSI and the Consortia.

For what concerns the documentation, both the PM and the PA manager will review all the documents before the official delivery (even in the draft form). The documentation review cycle is described in section 7. The Software Specification inspection activities (Software Requirements review, Architectural Design review) are considered as part of the Documentation life cycle.

In addition to the dedicated reviews, the DPU/ICU OBS software status will be part of the DPU/ICU hardware reviews foreseen in AD7.

6 Tests

See section 4.3.

7 Problem reporting and corrective actions

- **Documents.**
  All deliverable documents have to run through a review cycle, which is shown in Figure 7-1. The documents are first delivered from IFSI to the Consortia, and then are included into a delivery pack to ESA, in correspondence of the main reviews. Therefore, in principle, there should be two different reviews: one with the Consortia and the second with ESA. The first review will be included into the periodic FIRST OBS meetings organised between IFSI and the Three Consortia. The OBS draft documents and the software design technical notes will be circulated at least one month before the meetings. The comments from the Consortia will be gathered at least one week before the meetings. The comments and the consequent modifications will be discussed at the meetings. This iterative approach will lead to the first issue of the various documents, approved by the Consortia, to be submitted to ESA.
  The formal review cycle of the submitted documentation through RIDs will be applied starting from the first submission of the document to ESA.
  The considered formal ESA reviews are (RD4):

Review	OBS documents to be delivered
ISVR	DPU/ICU OBS URD, PA plan
IBDR,IHDR	DPU/ICU OBS SSD
ICDR	DPU/ICU OBS DDD,SVVP

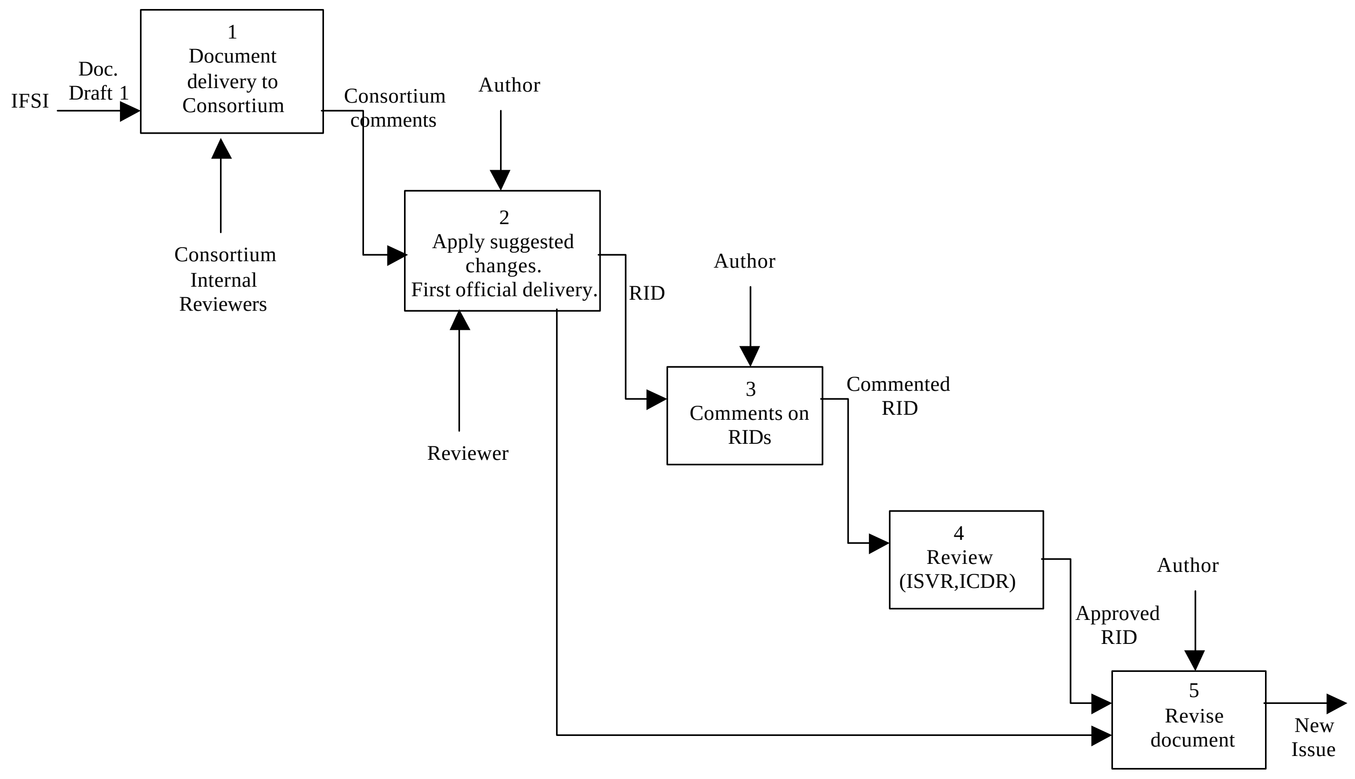

Figure 7-1 Review cycle for documents

- **Code.**
  A Software Review Board (SRB) shall be constituted, including a representative of the Software Development Team (and the PM and PA managers). All problems detected during tests shall be reported to the SRB, using the SW problem reporting form (SPR) included in this document, appendix A. The form will contain a field to indicate the criticality level of the related problem: major problems will be marked as critical. All the helpful material for the problem investigation will be attached to the form (e.g. intermediate steps results, test actions log files etc). The critical SPR's will be reported in the Monthly Progress Report of the DPU/ICU OBS PM to the Instruments PMs.
  The PA manager will provide a history log for all SPR's.

In order to handle software (and hardware) non conformances to the requirements (a non conformance is a deviation which can not be reworked to the original specification and/or configuration) a NCR control system will be adopted, described in AD6, section 8.5, similar to the one in use in the HIFI Consortium (RD3).
In case a NCR affects the DPU/ICU interfaces with subsystems, the instruments are invited to participate to the Material Review Board (MRB, see AD6, section 8.5.3), whose composition depends on the category of the NCR involved.

8 Tools techniques and methods

A CASE tool will be used for the design phases: a top-down approach will be adopted, both in the definition of the logical model (independent on the HW architecture and based only on the user requirements listed in the URD) and in the architectural design.
The **AxiomSys** CASE tool, based on the Yourdon DeMarco structured analysis method has been selected for these purposes. Given its real time extensions (based on the Hatley-Pirbhai method, see RD2), this tool will allow the production of state transition diagrams and timing diagrams.
The references for the tool are:
AxiomSys System Analysis C.A.S.E. tool
Structured Technology group, Inc. (http://www.stgcase.com)

The tool will be used under a Windows NT environment, in a client server configuration.
It will be used only for the software design. No automatic code generation is foreseen.
The maintenance of the Software design, in order to keep the AD aligned with the code will be performed only if a software non-conformance is detected, and the modification of the SSD shall be indicated in the NCR as one of the actions to be taken.

9 Configuration Management

The responsible for the configuration management is the PA manager (see section 2).

For referencing the documents, even in the case of common documents, IFSI will adopt different documentation identification rules for the three consortia, following the indications contained in:

1. SPIRE: SPIRE Document Management Plan, SPIRE-RAL-DOC-000032, Issue 1.0, Feb 99
2. HIFI: HIFI Project Management Plan (SRON-U/HIFI/PL/1999-001)

3. PACS: PACS Documentation and Identification Procedure, PACS-ME-GP-001, Issue 1, 22-03-2000

9.1 Identification for code

The following file extensions will be used:

Extension	**File type**
EXE	executable code, linked code
OBJ	Relocatable object codes
C	C source code
H	Include files
HEX	EEPROM/PROM code

The headers of modules and functions have been already described in section 4.2.
In both cases the Version.Revision number of the code shall be present in the header.
The configuration control will be performed according to what written in section 4.2.1.

The tools and the interfaces' simulators developed for the testing activities will not be put under configuration control.

Appendix A

Sample sheet for SPR

Software Problem Report		
Originator:	SPR Number:	Date:
Software Item Name:	Version:	Release number:
Priority (Major/Critical – Minor):		
Description of the Software Problem:		
Description of the environment:		
Suspected cause and recommended solution:		

SPR - Continuation sheet		
SRB decision – list of preventive/corrective actions		Verified:
SRB decision – list of preventive/corrective actions		Verified:
Attachments:		
SPR Close out date:	SRB approval - Name:	Signature:

Appendix B

Sample sheet for RID

Review Item Discrepancy		
Originator:	RID Number:	Date:
Document Title:	Document Ref:	Issue:
Description of the Discrepancy:		
Recommended solution:		
Supplier response:		
Attachments:		
RID Closeout date:	Closeout approval - Name:	Signature: