E U R O P E A N S O U T H E R N O B S E R V A T O R Y

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

LA SILLA OBSERVATORY

HARPS

Low-Level Maintenance Software

Design Description

Doc.No. 3M6-TRE-HAR-33110-0007

Issue 0.1

Date 17/05/01

Prepared S. Baeza 17/05/01 .....
Name Date Signature

Approved D.Megevand dd/mm/yy .....
Name Date Signature

Released dd/mm/yy .....
Name Date Signature

Change Record

Issue/Rev.	Date	Section/Page affected	Reason/Initiation/Document/Remarks
0.1	17/05/01	All	First preparation

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Purpose

This document contains the detailed design of Harps Low-level Maintenance Software and it is intended to provide all the necessary information about functional and non functional requirements presented in [7] and [9]. The requirements of the instrument hardware, the scientific goals and the scientific operation are described in [7]. User requirements and functional specifications of the rest of the Instrumentation Software modules (OSS, DRS and MS) are given in[9].

The intended audience for this document are the HARPS Project software managers, developers, instrument electronic designers, operational and scientific users of the instrument.

1.2 Scope

The detailed Design Description for OS/ICS is in [22].

VLT Software Specification for some tools, as part of the solution proposed are given in [5], [6] and [23].

1.3 Applicable Documents

The following documents, of the exact issue shown, form a part of this document to the extent specified herein. In the event of conflict between the documents referenced herein and the contents of this document, the contents of this document shall be considered as a superseding requirement.

[1] VLT-SPE-ESO-17212-0001, 2.0 12/04/1995 --- VLT Software - VLT INS Software Specification

[2] VLT-SPE-ESO-17240-0385, 2.2/prep2 05/05/1998 --- VLT Software - VLT INS Common Software Specification

[3] VLT-PRO-ESO-10000-0228, 1.0 10/03/1993 --- VLT Software Programming Standards

[4] 3M6-TRE-HAR-33110-0002, 0.2 22/02/2001 --- HARPS OS/ICS Software Users Requirements and Design Report

[5] VLT-MAN-ESO-17220-1332 1.5 06/03/2001 --- HOS/Broker for Observation Blocks User Manual.

[6] VLT-MAN-ESO-17240-2240 1.0 21/11/2000 --- INS Common Software for Templates User Manual.

1.4 Reference Documents

The following documents are referenced in this document.

[7] 3M6-PLA-HAR-33100-0005, 1.2 05/03/2001 --- HARPS Operation, Calibration and Maintenance Plan

[8] 3M6-TRE-HAR-33107-0001, 1.0 28/02/2001 --- HARPS Control Electronics Design Report

[9] 3M6-TRE-HAR-33110-0001, 1.3 02/03/2001 --- HARPS DFS Software User Requirements and Design Report

[10] GEN-SPE-ESO-00000-0266, 1.0 10/05/93 --- ESO Graphical User Interface Common conventions

[11] VLT-MAN-ESO-17240-0934, 2.3 01/03/2001 --- VLT INS Common Software - Base ICS User Manual

[12] VLT-MAN-ESO-17240-1973, 2.0 03/01/2001 --- VLT INS Software - Template Instrument User Manual

[13] VLT-MAN-ESO -17240-2265, 1.0 03/01/2001 --- VLT Software - BOSS User Manual

[14]VLT-MAN-ESO-17240-2153, 1.3 01/03/2001 --- VLT INS Common Software - Start-up Tool User Manual

[15]VLT-MAN-ESO-17240-2325, 1.0 10/10/2000 --- VLT INS Common Software - Configuration Tool User Manual

[16]VLT-MAN-ESO-17240-1913, 1.4 01/03/2001 --- VLT Software - Installation Tool For VLT SW Packages User and Maintenance Manual

[17]VLT-SPE-ESO-13730-1817, 1.1 08/09/2000 --- VLT Software - FLAMES/GIRAFFE ICS Design Description

[18]P.Ward, S.Mellor, Yourdon Press,1985 --- Struct. Development for Real-Time Systems

[19]J. Rumbaugh et. al., Prentice Hall,1991 --- Object-Oriented Modelling and Design

[20]VLT-SPE-ESO-10000-0011, 2.0 30/09/92 --- VLT Software Requirements Specification

[21]3M6-TRE-HAR-33110-0002, 1.2 22/02/01--- HARPS OS/ICS Software Users, Requirements and Design Report

[22]3M6-TRE-HAR-33110-0003, 0.2 09/05/01 --- HARPS OS/ICS Design Description.

[23]VLT-xxx-ESO-ppppp-nnnn, 1.0 26/02/99 --- AUTREP User's manual

[24]GEN-SPE-ESO-19400-794 1.1 25/11/97 --- Data Interface Control Document.

1.5 Abbrevations and Acronyms

The following abbreviations and acronyms are used in this document:

CASE	Computer Aided Software Engineering
CCS	Central Control Software
CMTL	Corrective Maintenance Task List
FDDI	Fiber Distributed Data Interface
DBMS	Database Management System
DCS	Detector Control System
FSDB	Function Status Database
FR	Functional Requirements
HOS	High Level Operating Software
HW	Hardware
ICS	Instrument Control Software
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IEDB	Instrument Environment Database
IMS	Instrument Maintenance Scheduler
IPDB	Instrument Performance Database
IPL	Instrument Parameter List
ISO	International Standardisation Organisation
LAN	Local Area Network
LCC	LCU Common Software
LCU	Local Control Unit

ME	Maintenance Engineer
MIDAS	Munich Data Analysis System
MLB	Maintenance Logbook
MUI	Maintenance user interface
N/A	Not Applicable
NFR	Non Functional Requirements
OA	Operation Astronomer
OB	Observation Block
OS	Observation System
OSI	Open Systems Interconnection
PMTL	Preventive Maintenance Task List
ROS	Remote Operating Software
SCCP	Software Configuration Control Plan
SRS	Software Requirements Specification
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined
TCS	Telescope Control Software
UI	User Interface
UIF	(Portable) User Interface (Toolkit)
VLT	Very Large Telescope
WAN	Wide Area Network
WS	Workstation

1.6 Glossary

word1
: definition of word1
.....

word2
: definition of word2
.....

.....
.....

1.7 Stylistic Conventions

The following styles are used:

bold
: in the text, for commands, filenames, pre/suffixes as they have to be typed.

italic
: in the text, for parts that have to be substituted with the real content before typing.

`teletype`
: for examples.

`<name>`
: in the examples, for parts that have to be substituted with the real content before typing.

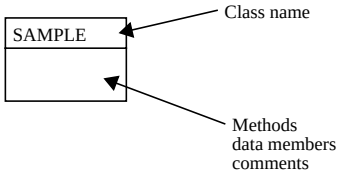
bold and *italic* are also used to highlight words.

1.7.1 Data Flow and Processor Model Diagrams

Data Flow and processor Model Diagrams are based on **De Marco/Yourdon** notation for real-time systems [18].

1.7.2 Class Diagrams

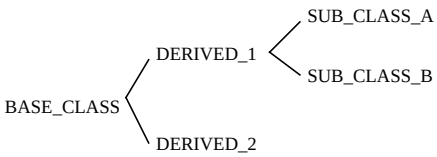
Class diagrams are based on OMT notation [19].
In order to simplify drawing of class symbols, a class is represented by a box, divided in two sections:



Lines connecting classes have to be interpreted as follows:

a ——— b generic association: a is associated to b
a ———● b multiple association: b (one or more) is used by a
a ———▷ b inheritance: a is a subclass of b
a ◇——● b multiple aggregation: b (one or more) is part of a

Class hierarchies are also represented as tree diagrams with the base class on the left and derived classes on the right, connected by lines:



1.8 Naming Conventions

This implementation follows the naming conventions as outlined in [3].

2 OVERVIEW

2.1 Introduction

The HARPS instrument is a High Accuracy Radial velocity for Planetary Search instrument dedicated to the search of extra-solar planets using precise radial velocity (RV) measurement. The instrument will be installed on the Cassegrain focus of the 3.60 m. telescope at La Silla Observatory.

HARPS OS is responsible for the coordination of all control activities related to a “single” exposure for its subsystems involved, namely HARPS instrument and CCD detector control system (FIERA).

The HARPS Instrument Control Software (ICS) is responsible for controlling and monitoring the various components of the HARPS instrument such as atmospheric dispersion corrector, carriage units, calibration unit, instrument environmental conditions, status and alarms.

The HARPS Low-level Maintenance Software will allow to manage low-level maintenance tasks. It also provides all information for everyday analysis for detecting problems and monitoring instrument status. Furthermore, it interacts with OS, ICS, OA and ME as shown in the following Figure 1.

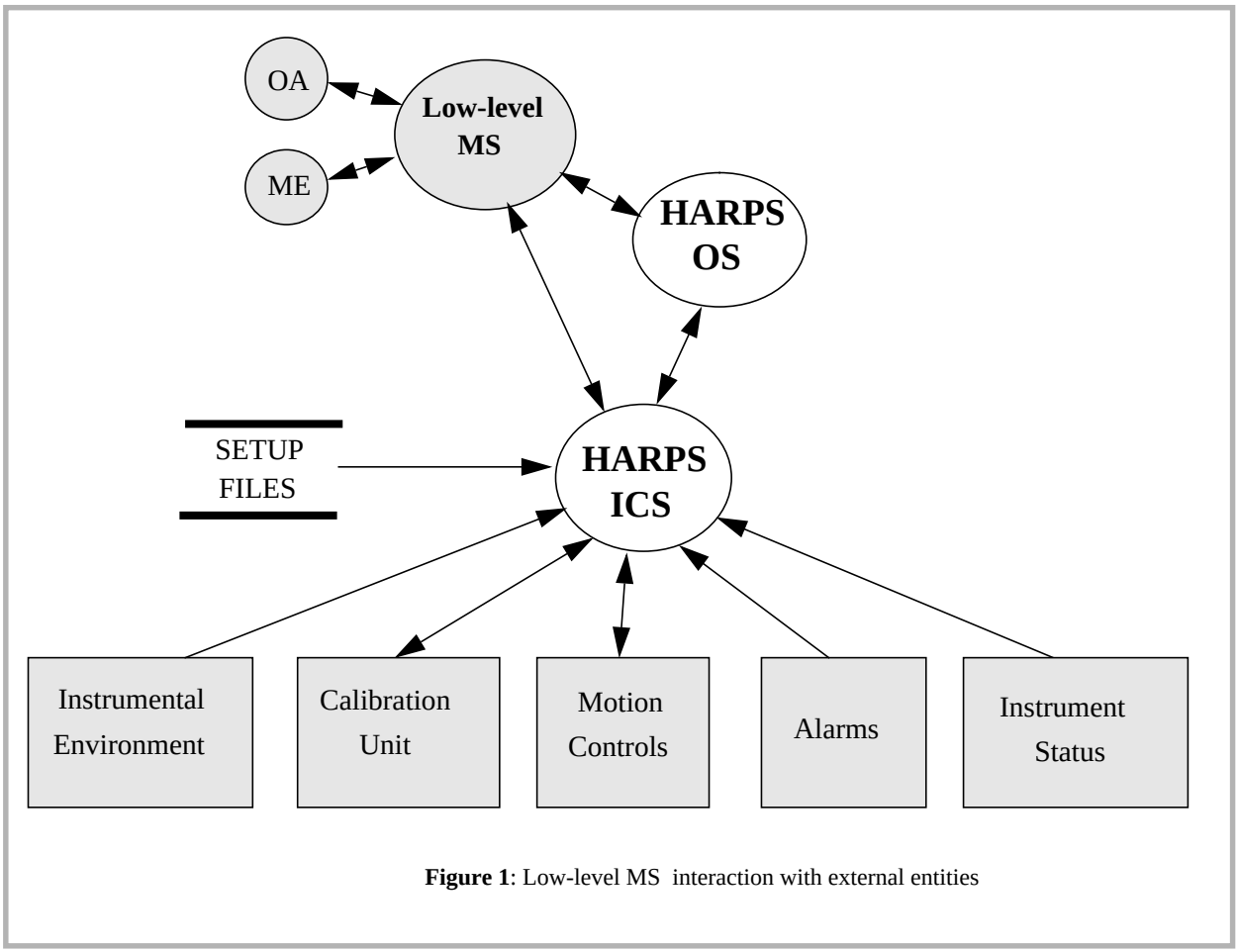


Figure 1: Low-level MS interaction with external entities

Low-level maintenance tasks are intended to be a preventive tasks, that avoid instrument failure, and corrective tasks, that initiate corrective maintenances.

High-level maintenance task are part of high-level maintenance, performed fully by dedicated maintenance templates which will be executed like calibration or observation templates. This kind of maintenance will not be part of low level maintenance software.

The rest of the manual is organized in the following chapters:

- Chapter 3 describes requirements for HARPS Low-level maintenance software.
- Chapter 4 describes Harps Maintenance Software Architecture
- Chapter 5 defines Data Description part of this low-level maintenance software.
- Chapter 6 describes Function Traceability Matrix

2.2 Hardware Environment

The WS platform consists of:

- HP Workstation, model B2000

The HARPS Cassegrain Fiber Adapter (HCFA) LCU platform consists of (see [8]):

- 1 CPU Motorola MVME2604-4331
- 1 Transition Module MVME712M
- 2 Motor controller MACCON MAC4-INC
- 2 Servo Amplifier ESO VME-4SA
- 1 Digital I/O board ACROMAG AVME-9481
- 1 Triple Voltage power supply KNIEL 358-006-02
- 2 Single Voltage power supply KNIEL 311-053-02
- 1 ISER - serial ports interface

The HARPS Spectrograph LCU platform consists of (see [8]):

- 1 CPU Motorola MVME2604-4331
- 1 Transition Module MVME712M
- 1 Motor controller MACCON MAC4-INC
- 1 Servo Amplifier ESO VME-4SA
- 1 Digital I/O board ACROMAG AVME-9481
- 1 Triple Voltage power supply KNIEL 358-006-02
- 2 Single Voltage power supply KNIEL 311-053-02
- 1 ISER - serial ports interface

2.3 Software Environment

he WS Software is developed on and for a Unix environment. HARPS OS will be based on BOSS (see [11]), which is a generic OS on its turn based on the CCS/evh toolkit. The GUIs will be built using the VLT Panel Editor.

The major pieces of software installed and running on the HARPS Instrument Workstation (IWS) are:

- UNIX operating system (HP-UX).
- VLT Common Software (VCS), including the Central Control Software (CCS).

- HARPS Instrument Control Software (ICS).
- HARPS Detector Control Software (DCS): FIERA.
- HARPS Observation Software (OS).
- VLT On-line Archive Client (volac, vcsolac).
- Real-Time Display (RTD).
- Broker for Observation Blocks (BOB).
- Optionally TCS in simulation mode.

CCS environments are described in [22].

2.4 Standards

The VLT standards are defined in [1], [2] and [3]. Templates development are described in [6] and [12].

3 HARPS MS SUMMARY

3.1 Purpose

This chapter covers a summary of software requirements specification for HARPS low level MS taken from [7] and [9]. These requirements have been divided into functional requirements and non-functional requirements

3.2 General Description

3.2.1 Product Perspective

In the VLT Instrumentation Software Specification (see[1]). There are six modules:

- Instrument Control Software (ICS)
- Detector Control Software (DCS)
- Observation Software (OS)
- Observer Support Software (OSS)
- Maintenance Software (MS)
- Data Reduction Software (DRS)

Maintenance software includes high-level and low- level maintenance software. MS will run on instrument workstation. In this chapter will be analyzed requirements for low level maintenance software. The other modules are analyzed in [4] and [22].

3.2.2 Product Functions

The MS is composed by a low level module related to hardware maintenance purpose, connected to ICS and DCS low level functions and self-test procedures, and therefore instrument dependent.

This software will include maintenance databases, templates and procedures to compare different instrument hardware behavior.

Low-level MS will provide all necessary information for future analysis. This information will be saved in maintenance databases or FITS log files.

3.2.3 User Characteristics

The different kind of users which will use these module are described in [21] and are listed here for completeness:

1. Operation Staff, that includes Operation Astronomers (OA)
   Operation Level: observing and maintenance level
   Privileges: access to direct commands and maintenance procedures
2. Software development and Maintenance Staff, that includes Maintenance Engineers (ME)

Operation Level: observing, maintenance and test level. Permission to incorporate other subsystems to be granted by procedures run by Operations Staff.

Privileges: access to command mode with highest privileges. Installation/test guidelines to be followed strictly to prevent interference with other users.

3.2.4 General Constraints

It is a general requirement for all ESO Instruments to use as much as possible the VLT Common Software.

3.3 Functional Requirements for Low-level MS Software

Functional requirements will define the most important characteristics of low-level MS. Without these requirements, it could be impossible to have a low-level MS.

1. (FR-MS1) Extract real environmental parameters and instrument functions and store them in maintenance Database (*MS-C1*)

    a. Real environmental parameters such temperature, pressure, liquid nitrogen level, etc. stored in Instrument Environment Database (IEDB).
    b. At start-up a self test of all Instrument functions will return status and values that will be stored in Functions Status Database (FSDB).

2. (FR-MS2) Compare real environmental parameters (key values) with reference values and value ranges (part of *MS-C3*)

    a. Value ranges, extracted from Instrument Parameter List (IPL) will be compare with values stored in IEDB and FSDB.
    b. Low-level MS will contain a function to compare key values and referenced values.
    c. Operation Astronomer and Maintenance Engineer will trigger defined tasks that belong to PMTL. These tasks can be triggered via MUI, and will compare values to discover possible errors.
    d. Comparing tasks will trigger automatic actions to solve possible problems or will send an email notification to the OA and ME.

3. (FR-MS3) Maintenance Scheduler to manage preventive maintenance tasks list (see [9],section 2.5.4).

    a. Check periodically in the PMTL whether a task is to be executed.
    b. Depending on the task, either send automatically the corresponding command to the low-level MS for execution, o inform OA.
    c. If PTML is updated and parameters tasks have changed, new tasks have been added or tasks have been deleted, the scheduler will execute this new version of PMTL.

4. (FR-MS4) Management of lists that contain preventive task (PMTL), corrective task (CMTL) and spare item (SIL).

a. it will be possible to add, delete, update a preventive task, corrective task and spare item in the corresponding list .
b. a template will be provided to add a new preventive task or corrective task in the corresponding list.

5. (FR-MS5) Every corrective task contained in CMTL has a procedure describing the actions to be performed.

a. detail explanation of procedures to follow when there is a problem listed in CMTL.
b. procedures will be written in a document.

3.4 Non-Functional Requirements

Next requirements are considered important, but they aren't a base for low-level maintenance software.

1. (NFR-MS1) Statistics analysis tool (see [9], section 2.5.3).
2. (NFR-MS2) Self-test at start-up of all instrument status (see [9], section 2.5.2).
3. (NFR-MS3) Editor via MUI to edit IPL, CMTL, PMTL, etc. (*MS-C4*)
4. (NFR-MS4) CMTL containing a link to a web page describing the actions to be performed (see [7])
5. (NFR-MS5) Maintenance Log Book.

3.4.1 Usability

This is related with necessary documentation to users to assure full understanding of low-level MS.

1. HARPS Low-level Maintenance Software Requirements and Design Report (this document) oriented to instrument software developers.
2. HARPS Low level MS Users Manual.
3. HARPS Low-level MS Acceptance Test Plan, oriented to instrument software developers.
4. HARPS Low-Level Maintenance Procedures for Preventive and Corrective Tasks. This document will include troubleshooting for corrective task.

3.5 Design Constraints

[MS-DC1]It is mandatory that the modules shall be written following VLT Programming Standards [3]

[MS-DC2]The ESO standard GUI Common Conventions will be followed for building user interfaces[10]

3.6 Hardware Limitations

See [21], section 2.3.4.2

3.6.1 User Interfaces

The most important characteristic of MUI are the following:

1. (UI-MS1) Operation and Maintenance staff will use MUI
2. (UI-MS2) The user will execute commands via MUI to know any instrument function status (MS-C2)
3. (UI-MS3) The user will execute commands via MUI to compare real parameters with reference values and range values (MS-C3)
4. (UI-MS4) Desirable to have a panel for scheduler.

3.6.2 Software Interface

The software interface for Low-level MS are OS, ICS and DCS. The following points show why is necessary these interfaces :

1. MS needs to read information about instrument status and environmental parameters periodically. This will be done through ICS software interface and DCS software interface. OS is the main interface.
2. Every preventive task could trigger procedures to check status, that will be executed by ICS. For example, check pressure of detector dewar, that triggers a command to ask to ICS for the actual status.
3. Some Preventive task could be executed through low-level maintenance templates via BOB (see [5]). This tool communicates with OS.

The interface with ICS is desirable because ICS is the standard software interface for every software system that needs to work with the instrument. If ICS is not considered as an interface, every command would be sent directly to LCU's, but it is not part of VLT standards.

There are some characteristics if the Low-level MS software considers these software interface:

- ICS should be alive if Low-level MS is working with HARPS instrument. The instrument maintenance will be done day and night.
- OS should be alive too.
- BOB will be an important tool for Low-level maintenance Software. This last point will be explained in detail in next sections.

4 HARPS MS ARCHITECTURE

This chapter describes HARPS Low level MS in terms of VLT standard solutions versus the solution recommended by users in [9] and [7].

4.1 Overall Software Configuration

Figure 3.1 of [9] shows the top level configuration of HARPS software. At this level the whole package is seen as only one module, grouping in it all the internal tasks. The external modules that interacts with HARPS are explained in [9].

The definition of the schematics elements are defined in [9] and listed here for completeness

- In the module diagrams, only the software parts relevant to the described module are shown, being other modules, units or data repository of any kind.
- Circles represent software functional parts internal to the described module.
- Rectangles represent external entities, being software, hardware or human.
- Data stores are represented by two strong parallel lines.
- The arrows describe interactions between different units.
    a. The main data flow is represented by strong arrows.
    b. Message between software functional parts appear as thin arrows.

4.2 Internal Software Configuration

The internal Software configuration in a deeper level is described in [9] in figure 3.2. Part of the figure 3.5 of [9], will be showed in the next figure:

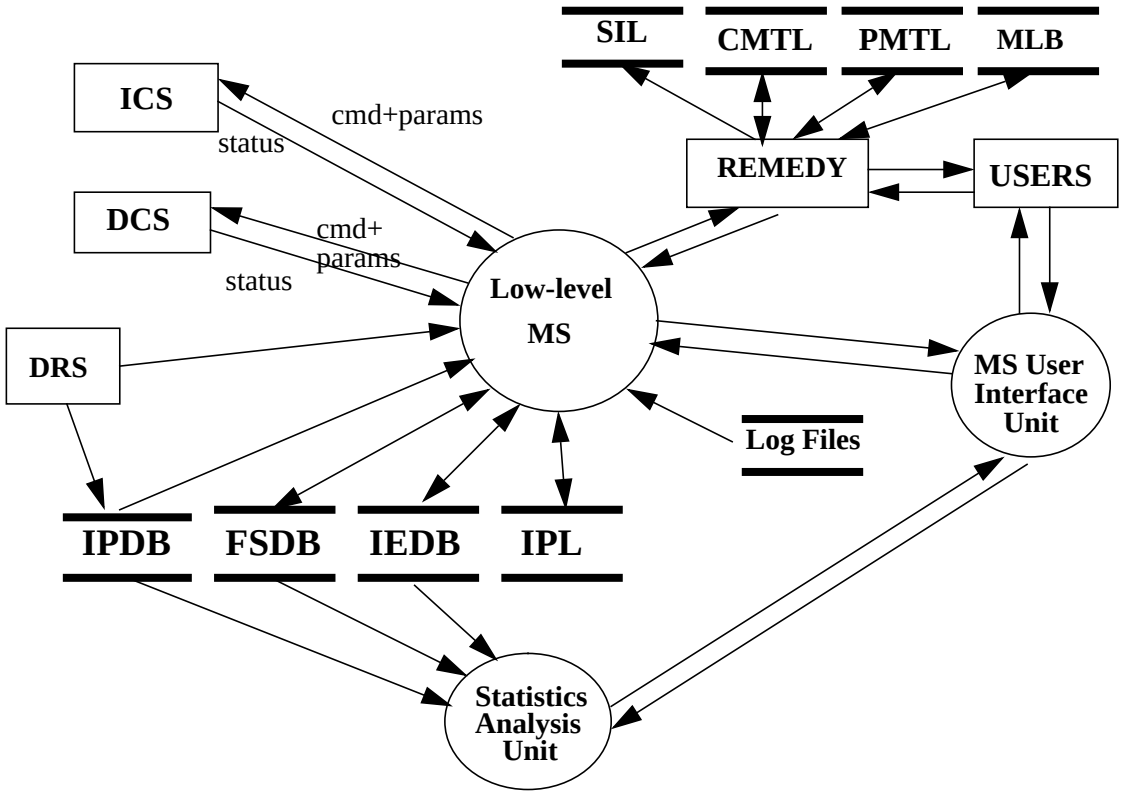


Figure 2: Low-level MS interaction with external entities and Databases

The model presented here will be adapted using VLT tools and standards. Next points will show these changes:

1. Remedy doesn't have direct interaction with Low-level MS. Remedy is in AYLLA server outside telescope network. If any telescope team needs to connect to this server, it will be through network connection outside of telescope. This tool will be considered to manage SIL. Another solution will be provided to manage CMTL, PMTL and MLB.

2. MS User Interface Unit will be BOB panel (see [5]).

3. For Statistics Analysis Unit will be considered AUTREP. The input of this tool is a log file with FITS keywords. With the information contained in the log file is possible to plot a graphic with the necessary FITS keyword.

4. For corrective task is possible to use templates and procedures for solving a problem. The procedures will be part of documents and internal web pages.

5. Preventive tasks will be executed via maintenance templates through BOB tool. Every task will have a special maintenance template that will be defined later in this document. Every preventive task presented in [7] in table 4.1, has different periodicity. Most of task presented in PMTL, will be executed with

periodicity greater than 1 day. This kind of task could be executed via templates as part of everyday maintenance procedures. OA and ME are the final responsible of the execution of templates.

There are preventive task that periodicity is equal to 1 hour (or periodicity less than 1 day). For this kind of task, the ICS will execute testing and checking of instrument parameters and this includes checking of parameters presented in PMTL. If there is a problem, the system will trigger an alarm (this could trigger a beeper call). It is possible to show a message on screen indicating the problem. An email could be sent too, but this is not an on-line solution and the idea is to solve the problem as soon as possible.

6. Every information considered in databases FSDB and IEDB will be stored in FITS LOGS files (see [5] and [6]). This kind of files are archived in Garching Databases. This is the VLT standard for all information that is associated to a FITS log file. A workstation process will update this information periodically and the information will be stored in a maintenance FITS log file.

7. Every template executed as part of maintenance procedures, will include sequences tasks to check Maintenance logbook. This checking will allow to know what kind of preventive/corrective tasks has been successfully executed.

8. Every command sent to DCS and ICS will be sent using maintenance templates. Actually all templates are executed via BOB. This tool is communicated with OS and every command to ICS and DCS is directed by OS.

9. All information contained in IPL, will be in ICS on-line database.

Next figure, will show the new configuration proposed using VLT tools and standards:

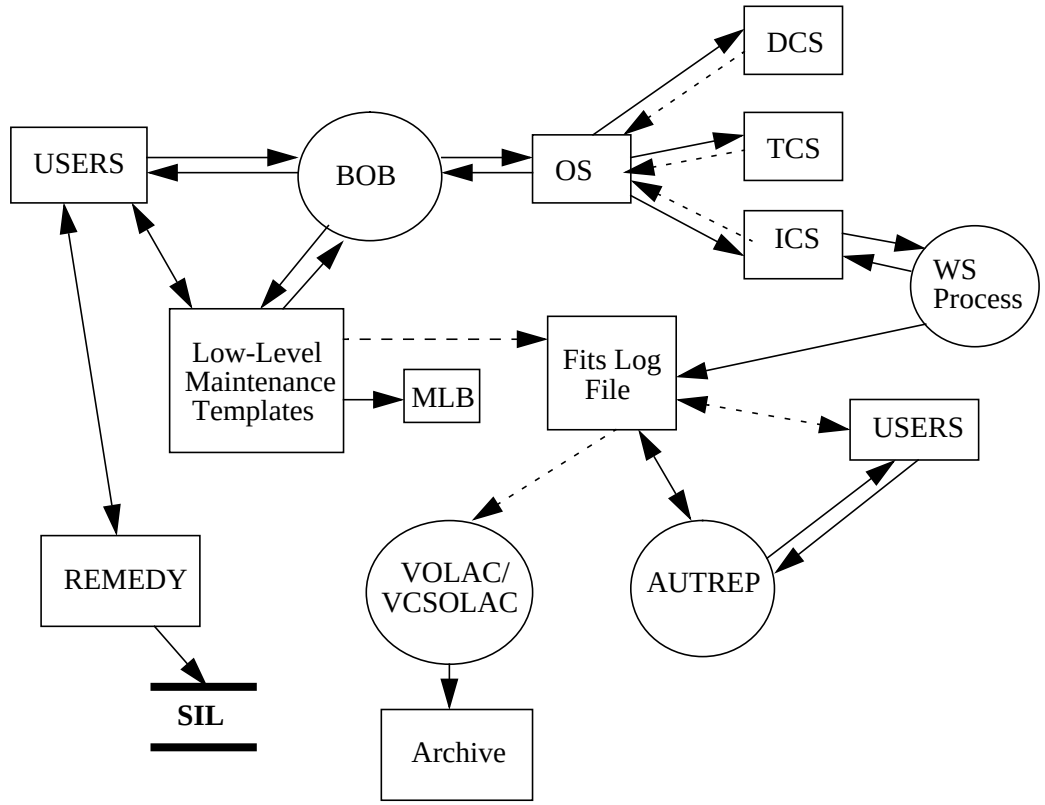


Figure 3: New configuration for Low-level MS

The solution proposed, has units and procedures that are part of VLT standards:

- BOB and the interaction with users and templates are used everyday in a observation night. Now, the functionality of these units and their interaction will be useful to execute low level preventive maintenance tasks using templates.

- Fits log files are part of output information obtained during observation night. These output are archived with the help of Volac and Vcsolac. Volac is an interface process between the Observation Software (OS) and the vcsolac process. Volac attaches to database events for new data and creates a symbolic link in vcsolac data directory when new data was communicated.

  Vcsolac is the final process that transfers fits log files to Archive. Is important to mention that this procedure is understood to store observation data with special fits log and image information. Clearly, the information obtained after an execution of a low level maintenance template isn't an image. Volac and vcsolac needs images in a fits log file, if there is not an image, the transferring process to Archive would fail.

Information archived is not available immediately. To solve this detail, it will need to have a directory structure and store files with the recent information.

Files and FITS log files will need to define a backup procedure (this include archiving).

- Remedy is a special tool to manage problem request, trips, night report, etc. This friendly tool is good to manage SIL, but is necessary to think possible changes in Remedy Database. (TBD)

- Autrep is an acronym of AUTomatic REPort (see [23]). It is possible to process log files and generate different kind of reports using the data stored in this kind of files. The idea is to process Fits log files using this application.

4.3 Advantages and Disadvantages

Automatic execution of Preventive Task is not part of this solution. This means that proposed Low-level maintenance software, will not have a Scheduler.

The exclusion of Scheduler will not affect the system, this is because the task will be executed with the necessary frequency to avoid problem in the instrument. Remember that preventive task has been created to minimize errors. This important affirmation is still valid for the proposed semi- automatic solution.

Is important to mention that, semi-automatic solution, implies a user dependent solution:

- With an automatic solution, the user could trust that system will never fail.
- Interaction of the user with the system could be lost with an automatic solution.
- A failure in the scheduler, could be not easy to detect it.

A user dependent solution is good, because:

- The user will interact with the system understanding that he/she is the main responsible of the preventive and corrective task execution.
- The user will have all necessary tools to create, modify and execute low-maintenance tasks. The absence of a scheduler could not avoid that user fulfills everyday work.

The following points show advantages of the proposed solution:

- Low-level MS software is developed with known VLT tools.
- The development time for the proposed solution is less than development of an automatic solution.
- The proposed solution is flexible. The execution of templates is done when the user need it.
- VLT software is known. This facilitates support every work for operation and software team.
- User interface for the solution proposed is known. This reduces training time.

For safety reason, no maintenance templates should run during normal night operation.

4.4 Monitoring Sensor values Workstation Process

There is an special task that should be considered; the IEDB and FSDB (remember that this is now, a FITS LOG file) should be updated periodically (periods less than 1 day). This means that this process or other should read the On-line ICS Database and update Instrument Environment and Function Status FITS LOG file.

This WS process is basically a continous loop, which reads the required values from OLDB and store it in maintenance FITS log file.

It will be a table stored in OLDB with the following information:

- Value or item that will be filled, for example, pressure of vacuum vessel
- fits log associated to item
- database point of On-line ICS Database that will be updated

OLDB will store update-period of execution of workstation.

The values selected by users will be updated periodically in the FITS LOG file. The pseudo-code for this task is the next:

```
while  update_period=actual_period
            read actual instrument environment parameters
            write actual values on FITS LOG file
end while
```

The preliminary commands are the following:

- START
  Indicating that the process begins execution.
- STOP
  Indicating that the process ends execution.

4.5 Low Level Maintenance Templates

Next list will show all low level task that will be executed using templates:

- check operation time of thorium lamp
- check operation time of tungsten lamp
- test and check instrument functions
- measure and check instrument performance

Most of task are part of low level preventive maintenance task list. The action are mostly checking tasks.The rest of preventive task will be part of procedures that operation team will execute with the required periodicity.

The templates initially defined are part of the next class diagram:

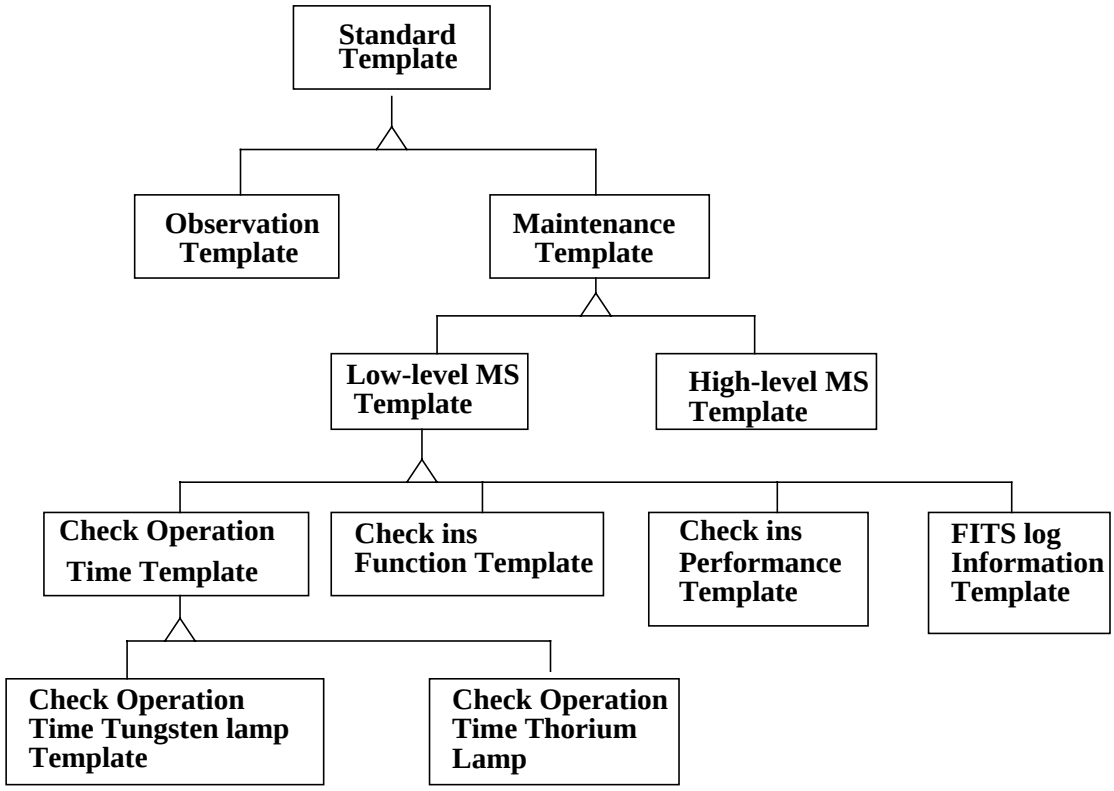


Figure 5: Class Diagram for templates

Standard Templates: This is an abstract class that has general definitions for every kind of templates.

Observation Templates: This class represents templates that are defined for everyday observation work.

Maintenance Templates: This is an abstract class that has general definitions for low level and high level maintenance templates.

High Level Templates: This maintenance template is explained in [7].

Low Level Templates: This abstract class represent low level maintenance templates.

Checking Operation Time Template: This class will be created to check operation time values. **Checking Instrument Function Template**: This templates class will check instrument functions

Checking Instrument Performance Template: This template class will execute different performance analysis in every instrument part.

FITS log information template: This template will allow to check information stored in FITS log files.

4.5.1 Maintenance LogBook

Maintenance logbook will be created to log information about execution of every preventive and corrective

tasks. This will allow to manage maintenance procedures, and to know a global status about task executed, number of tests executed, number of failures, responsible users that solve problems and historical information to control periodicity.

This logBook will be read at the beginning of every template execution. Initially, the idea is to check periodicity of execution of a task related to that template. If a task is done with periodicity less than reference period (formal period assigned to a task) it will asked to user if he/she wants to run the template (if the user decides "no", nothing dangerous would happen. In other case, the template will be executed without asking to the user. This is because periodicity is equal or greater than reference period, the execution of template could cause problems). The next pseudo-code will explain it better:

```
\\\\\\
\\ MLB testing
\\\\\\
if actual time < reference time then
   if user want to execute template then
       begin execution of template
   else
       exit \\ template not executed
   end if
end if
```

For example, if testing motor movements template is executed for a motor A, before execution, the maintenance logBook will be analyzed to see past execution for motor A testing.

4.5.2 Check Operation Time

This is an abstract class that has general definitions for templates class to check operation time of calibration lamps. If operation time is greater than allowed time stored on IPL, an email will be sent to responsible person.

FITS keywords associated to this class:

- INS.LAMPi (1-5) (complete information of calibration lamp to be tested)
- INS.LAMPi.ONTIME (operation time of calibration lamp)

4.5.3 Checking Instrument Functions

A self-test of all instrument functions will be done using a template script. The returned status and returned values after self-test will be analyzed. The next point will show characteristics of this method (that belongs to "Check instrument function template" class).

- the template script will obtain status and final values after start-up. Depending of global state of system (see [22], chapter 5) after start-up, it will be a predefined state (or value) for every instrument function (see also [22], table 5.1 with operational states).
- the template script will be executed after start-up
- the output will be all status and values returned after start-up

In order to check the validity of the test, it is possible to check allowed global state defined in IPL versus real global state obtained after start-up. The validation of values and status after start-up versus defined values in IPL will be an output too.

4.5.4 Check Instrument Performance

There are three basic task sequences:

1. Movement motors monitoring: This task will allow to check the correct movement of motor.
2. Shutter movement: This task will permit to check shutter functions.
3. Check Calibration Lamps: This task will test calibration lamps .
4. Check sensor: this task will check stability of instrument sensors.

Next figure will show class diagram for templates that check instrument performance:

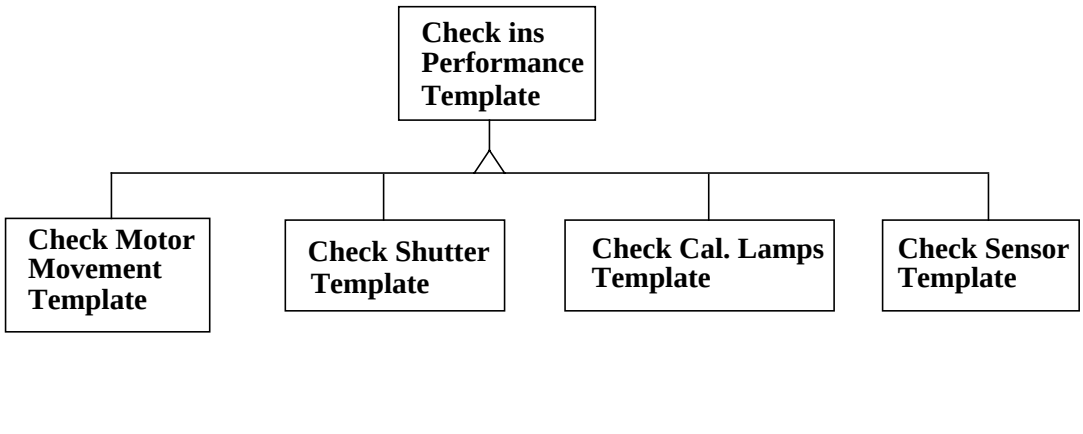


Figure 6: Class Diagram for Check Instrument Performance Template

The motors, shutters, calibration lamps and sensors to be tested are defined in [22].

4.5.4.1 Testing Motors Movement

The method of class “Check Motor Movement Template”, is a template script (see [5]). This template script will run motors to be tested in both directions.

Input

- motor to be tested
- current for servo amplifier
- initial position
- final position
- sample rate

Process

```
MLB testing
foreach motor
      setup initial position
      begin sampling plot
      moving motor until final position
      setup final position
      moving motor until initial position
      stop sampling plot
end foreach
```

Output

- plot file with position versus time
- plot with current of amplifier versus time
- file *.samp with all data obtained after testing
- log in MLB

Template Keyword	Keyword (ICS/OS)	Input/Output
TPL.<type>.<identification>	INS.<type>.<identification>	input/output
TPL.<type>.NO TPL.<type>.ENC TPL<type>.ENCREL TPL.<type>.LIMIT TPL.<type>.SPEED	INS.<type>.NO INS.<type>.ENC INS.<type>.ENCREL INS.<type>.LIMIT INS.<type>. SPEED	input/output
TPL.<type>.MIN	INS.<type>.MIN	output
TPL.<type>.MAX	INS.<type>.MAX	output
TPL.<type>.MEAN	INS.<type>.MEAN	output

Table 1: Keywords for Checking Motors Template

Values for <type> (see [22]):

- MIRRi
- ADCi
- OPTi
- FILTi

The <identification> is information to identify the motor, for example, NAME, TYPE, ID, etc. (see [22])

4.5.4.2 Checking Calibration Lamps

The method of class “Check Calibration Lamp Template”, is a template script (see [5]). This template script will detect faulty lamps and will examine temporal stability. There are two way of testing:

- For a select lamp, this test record traces of exposures meters counts for a predetermined period of time.
- Testing lamps with the information collected through CCD. It is necessary to do an exposure.

Input, process and output consider the second solution:

Input

- Lamp to be tested
- number of tests (number of exposures)
- time between exposures

Process

```
MLB testing
turn on  lamp
check returned signal
if lamp is not turned on then
     send  alarm indicating the problem
else
     while number of test executed < number of tests
          take exposure (1 seg)
          calculate intensity
          wait time between exposures
     end while
end if
```

Output

- min, max and average intensity
- file with all data obtained after test

Template Keyword	Keyword (ICS/OS)	Input/Output
TPL.LAMPi.<identification>	INS.LAMPi.<identification>	input/output
TPL.LAMPi.WAIT	INS.LAMPi.WAIT	input/output
TPL.LAMPi.TIME	INS.LAMPi.WAIT	input/output
TPL.LAMPi.INTENSITY		output
TPL.DET2.CTMIN	INS.DET2.CTMIN	output

Template Keyword	Keyword (ICS/OS)	Input/Output
TPL.DET2.CTMAX	INS.DET2.CTMAX	output
TPL.DET2.CTMEAN	INS.DET2.CTMEAN	output

Table 2: Keywords for Checking Lamps Template

4.5.4.3 Testing Shutters

The method of class “Check Shutters Template”, is a template script (see [5]). The characteristics are the next:

- check signal to test if shutter open/close when is necessary
- for a select shutter it will possible to execute open/close action without problems.

The output of the test:

- sampling time
- type of signal (open signal or close signal)
- results indicating if every test was successful or not

FITS keywords included in this test:

- INS.SHUTi.<identification>
- INS.SHUTi.TIME

4.5.4.4 Check Sensors

This template will be created to check sensors calibration. A preliminary test will be consider to change reference values, for example, change reference temperature. The test will verify if sensor can detect this change and return to initial reference value.

4.5.5 FITS log Information

The sequences tasks or method of the class “FITS log information template” will analyze all information contained in FITS log file.

Input

- FITS log file
- type of test (information of some keywords, behaviour of LN2 level, etc) (TBD).

Process

```
Stop WS Process
if user needs to backup-up FITS log file then
   Backup up actual fits log file
end if
if user needs to change periodidity of execution of WS process then
```

```
   change periodicity
end if
Begin fits log processes
```

Output

- TBD

The advantages of this template are the following:

- the template will allow to execute different analysis based on information contained in FITS log file.
- the execution of the test will be done when the user need it.
- it will be possible to execute manual backup when the user need it.

FITS keywords are not defined yet.

4.6 Proposed Solution for Preventive Maintenance Task

Preventive Task	Solution Proposed (templates or workstation or ICS/OS processes)
Check Level, LN2	ICS/OS processes, see [22]
Detector Dewar, check pressure	ICS/OS processes, see [22]
Vacuum vessel, check pressure	ICS/OS processes
Air conditioning	OA
Thorium lamp, check operation Time	Template
Tungsten lamp, check operation time	Template
fiber head	OA
Check and test Instrument functions	Template
Check and test Instrument Performance	Template
Optical fibers	OA
CCD detector	OA

Table 3: Solution proposed for Preventive Tasks

5 DATA DESCRIPTION

This chapter will describe the files, databases and directories hierarchy of the low-level maintenance software.

5.1 Low Level Maintenance Data File Types

The following file set is common to all tests:

- testX.samp

  temporary data file containing instrumental data produced during the test (e.g. sampled motor current, sampled exposure meters values, etc.). The results are marked with all information needed to trace the origin of the test (test name, date, time, instrument settings, comments, etc)
- MLB

  This log file will contain all information of logs of all preventive and corrective tasks executed.
- testX.fits

  FITS log file that will contain FITS keyword of tests and workstation processes output. The information of this file will be an input for AUTREP to plot and analyse selected FITS keywords.

5.1.1 Maintenance Log Book

The information stored in the MLB will be the next.

- test type
- instrument part tested (motor, sensor, calibration lamp, etc)
- responsible (who did the test?)
- execution time

5.1.2 FITS log file

The next concepts are related to this section:

- **FITS keyword**

  A word of maximum 8 alphanumeric characters used in FITS headers to encode parameter information related to the data formatted in the FITS file.
- **Log File**

  A computer readable file containing log records. Log file are written by handlers that receive log requests from distributed applications running in the on-line environment.

The log file will record a number of actions and parameters which are defined in the corresponding dictionaries (see [24)]. In addition, log files may temporarily include any number of parameter records to be used e.g. for troubleshooting purposes.

The log file format is designed to allow an accurate trace of VLT operations. Every log record is uniquely identified by the logging source and date/time stamp. This design allows to merge all log records in the log database independently of how many log files where created.
A log file consists of maximum 250 byte long records terminated with a newline character, however key-

words and values must be written within the first 72 character is due to the need to be able to include relevant log records in the FITS headers of observations.

Log records have the general format:

hh:mm:ss> *keyword / comments [<source mask>]*

or **hh:mm:ss>** */ comments [<source mask>]*

where:

- **hh:mm:ss>** is the time stamp
- *Keyword* is a hierarchical keyword which explains what happened (action keyword) or identifies the reported parameter (followed by = value)
- *comments* explain the keyword reported
- *[<source mask>]* is the event source identification mask

All log files will be stored and archived in the VLT Archive Facility. From there, they will be available for engineering monitoring and other needs. For more information see [24].

5.2 Setup Files

Every test requires the instrument and/or detector in a predefined configuration. A testX.ins and/or testX.det setup files are provided for each test.

5.3 Low Level data file Hierarchy

The following hierarchy will be implemented to maintain all the files needed by Low level maintenance to run tests.

- HARPS_MS=$INS_ROOT/SYSTEM/MISC/MS
  root directory of low level maintenance software. It will contain MLB, Fits log files, *.samp and historical information

Test templates will be in the next directory.

- $INS_ROOT/SYSTEM/COMMON/TEMPLATES

Inside this directory there are next subdirectories

- $INS_ROOT/SYSTEM/COMMON/TEMPLATES/OBD
  for observation block. For low level maintenance it will be an OB that will call all maintenance templates.
- $INS_ROOT/SYSTEM/COMMON/TEMPLATES/TSF
  for signature files with keywords definition as arguments of template scripts
- $INS_ROOT/SYSTEM/COMMON/TEMPLATES/SEQ
  for template script.

Section 5.4 will describe in more detail the structure of technical templates.

Under test directory the following structure is common to all test:

- $HARPS_MS/<DATE>/

  contains the historical information processed with date DATE. DAte is part of the name of the subdirectory.

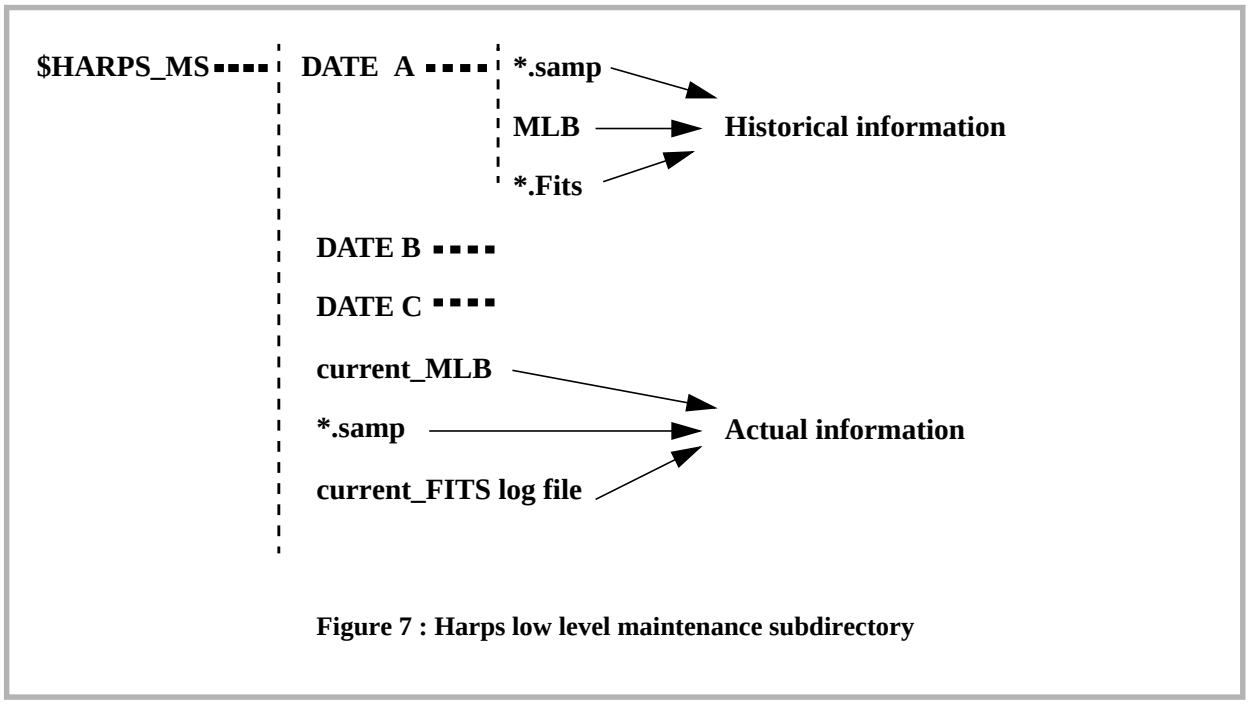


Figure 7 : Harps low level maintenance subdirectory

5.4 Archiving Results

- Templates.

  All low-level maintenance software templates will be stored in a VLT module. If module is installed, all templates will be installed in $INS_ROOT/SYSTEM/COMMON/TEMPLATES/.
- FITS log files.

  It will be stored in Archive
- Workstation special process.

  It will be stored in Archive as VLT module
- *.samp.

  It will stored in Archive.

5.5 On-line Databases

The IPL will be stored in ICS On-line database. The information to consider is the next (see [9]):

- name of the environment or function parameter
- nominal value
- allowed minimum value

- allowed maximum value
- normal status (for function parameter only)
- failure code

Remedy will manage SIL. The modifications to remedy database will consider to create attributes in one or more tables. The attributes are explained in [9] section 5.6.8.

Data description for ICS and OS are described in [22].

5.6 Technical Templates

There are some concepts related to this section:

- **Observation Block**

  The smallest schedulable observational unit for the ESO VLT. An observation block contains a sequence of high level operations, called *templates* that need to be performed sequentially and without interruption in order to ensure the scientific usefulness of an observation.
- **Template**

  High level VLT operation. Templates provide the means to group commonly used procedures in a well defined and standardized unit. Templates have input parameters described by template *signature*, and produce results that can serve as input to other templates

BoB will be used to execute Ob's. These OB's contain templates and of each templates contain instructions for the OS process of the instrument it is controlling. This communication is taken care of by BOB.

Next example, will show which is the differences and relation between *.obd, *.tsf and *.seq will send a signal SETVAL command to a process. The parameter of this command i supposedly a single integer, in the range 100-1000. This parameter will be passed to the template as an argument.. Next, is the template signature file.

```
PAF.HDR.START                  : #Marks start of header
...

PAF.HDR.END

#next set  of keywords  are mandatory

TPL.INSTRUM                      "TEST";
TPL.MODE                          "";
TPL.VERSION                      "0.1 ALPHA"
TPL.REFSUP                        "";
TPL.PRESEQ                        "waTemplate.seq";#sequencer script
TPL.GUI                           "";
TPL.TYPE                          "test";
TPL.EXECTIME                      "computed";
TPL.OVERHEAD                      "1.0";
TPL.RESOURCES                      "";

#here start the description of the various arguments of the template
#script. In this case we have only one parameter
```

```
TPL.PARAM                       “SEQ.VALUE”
SEQ.VALUE.TYPE                  “Integer”
SEQ.VALUE.RANGE                 “100..1000”
SEQ.VALUE.DEFAULT               “555”
```

The above file shows there is a unique argument for the script, and is defined with the name VALUE in the SEQ category . BOB will look for a corresponding template script called `waTemplate.seq` (TPL.PRE-SEQ). The name is the same for signature file, `wsTemplate.tsf`. The template `waTemplate.seq` look as follows:

```
#Example Technical Template script
proc waTemplate {array reserved}{
   foreach array $array {
      upvar $array $array
   }

   set TPL(REFSUP); set TPL(EXPNO) 1
   set timeout 10000

   #send command SETVAL
   tplLog “About to send SETVAL command...”
   set reply [sendCmd $timeout “SETVAL $SEQ(VALUE)”]

   return{}
}
```

It is possible to put this technical template into an OBD. In fact, the example below calls this template twice, with different arguments:

```
#Standard parameter file header

PAF.HDR.START
PAF.TYPE                    “OB description”;
PAF.ID                      “”;
PAF.NAME                    “”;
PAF.DESC                    “”;
PAF.CRTE.NAME               “BOB”;
PAF.CRTE.DAYTIM             “2000-04-20t11:34:54”;
PAF.LCHG.NAME               “”;
PAF.LCHG.DAYTIM             “2000-04-20t11:34:54”;
PAF.CHCK.NAME               “”;
PAF.CHCK.DAYTIM             “”;
PAF.CHCK.CHECKSUM           “”;
PAF.HDR.END                 ;

#Observation block description
```

```
OBS.ID                          “1”
OBS.PI-COI.NAME                 “Peralta”
OBS.GRP                         “”
OBS.NAME                        “workshop a”
OBS.PROG.ID                     “B”

#First TPL

TPL.ID                            “waTemplate”
TPL.NAME                          “demo template”
SEQ.VALUE                         “444”

#Second template

TPL.ID                            “waTemplate”
TPL.NAME                          “demo template”
SEQ.VALUE                         “111”
```

After the creation of these files, the user will execute next steps:

- load OBD into BOB
- execute the OB, sending proper SETVAL command to the process
- modify the value assigned to SEQ.VALUE

6 FUNCTION TRACEABILITY

Requirement References (section 3.3)	Solution Description	Section with Description of Solution
FR-MS1 Extract Real Env. Parameters and instruments functions	IEDB and FSDB (now, FITS log files) will be updated, through workstation process	section 4.2 number 6, section 4.5
FR-MS2 Compare real Env. parameters with reference value	A special template will analyze FITS log file (with real values) versus references values. This template will be executed when the user need it.	section 4.5.5
FR-MS3 Maintenance Scheduler	Preventive tasks will be executed via templates (task with periodicity equal o greater than 1 day). ICS/OS will execute testing of most of instruments sensors (preventive task with periodicity less than 1 day)	section 4.2 number 5,7 section 4.4, 4.6, 4.7
FR-MS4 Management of lists PMTL, CMTL and SIL	SIL will be managed via remedy PMTL is now a group of templates and WS process CMTL will have written procedures that can be updated.	section 4.2 number 1, section 4.5, 4.6 section 3.4.1
FR-MS5 Procedures for Corrective Tasks	CMTL procedures are described in Low-level MS manual	section 4.2 number 4
NFR-MS1 Statistics Analysis	Autrep	section 4.2 number 3
NFR-MS2 Self test at start-up	Check instrument functions template	section 4.5.3 section 4.5.5
NFR-MS3 Editor via MUI	TBD	
NFR-MS4 CMTL link to a web page	TBD	
NFR-MS5 Maintenance logBook	MLB will store all logs about low -level maintenance templates execution and workstation process	section 4.5.1 section 5.1.1