EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral
Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

INSTRUMENTATION DIVISION

New General detector Controller

Document title:	***NGC Control Software System - Optical Instruments - High-level Software Design Description***
Document number:	VLT-SPE-ESO-13660-3835
Issue No	2.0
Date	09.08.2006

Prepared by C. Cumani, A. Balestra .....

Approved by D. Baade .....

Released by A. Moorwood .....

CHANGE RECORD

Issue	Date	Section / Paragraph affected	Reason / Initiation / Remarks
1.0	2006-03-21	All	First version
2.0	2006-08-09	All	Processes labeled with _$CCDNAME environment variable are now labeled with _<camera> identifier
		All	All references to NGC requirements have been deleted and moved unto the NGC software requirements (if not already there). From now on, only references to NGC SW requirements are left.
		3.2.1	It is stated that the relevant information to startup NGCOSW - like the _<camera> identifier - can be set via environment variables or via startup parameters
		6.1	Error and logging interface added
		6.4	New paragraph for image file and rtd interface

TABLE OF CONTENTS

1 Introduction

1.1 Purpose

This document aims to present the design of the Next Generation detector Controller (NGC) Control Software for optical instruments (NGCOSW).

NGCOSW provides the services needed to operate the NGC installed either on ESO optical instruments or stand-alone in the laboratory.

1.2 Scope

This document describes NGCOSW architecture in charge of implementing NGC SW requirement specifications [AD7] specific to optical instruments.

The NGC base software, which provides the access to the NGC hardware and is used by both the optical and infrared instruments, is described in [AD9].

The NGC software specific to infrared instruments (NGCIRSW) is described in [RD11].

1.3 Applicable and reference documents

Applicable and reference documents are listed in the "NGC Project Documentation" document, VLT-LIS-ESO-13660-3906.

1.4 Acronyms

The acronyms used within the NGC project are listed in the "NGC Project Acronyms" document, VLT-LIS-13660-3908.

1.5 Glossary

The terms used within the NGC project are explained in the "NGC Project Glossary" document, VLT-LIS-13660-3907.

1.6 Overview

Architecture and detailed design of the NGC software for the optical instruments (NGCOSW) are presented using the Unified Modeling Language (UML) graphical notation [RD50] and [RD51].

UML diagrams in this document have been created using the Enterprise Architect tool.

DOORS is the tools used to manage requirements and design, and Word documents shall be generated from the DOORS version. As the document has been written directly in DOORS, it is advisable to examine it using DOORS itself. The printed (Word) version looses some of the dynamicity of the document.

This document is organized in the following chapters.

This first chapter defines the purpose and the scope of this document.

The Architecture chapter provides a short introduction to NGC (see [AD8] for a more detailed description) and an overview on the NGCOSW high level structure. NGCOSW processes, their relationship, and their physical location are described with a general

deployment diagram, while a package diagram shows the dependencies among NGCOSW packages. NGCOSW processes operational states and state-changes are illustrated through a state diagram.

The following two chapters describe the design of workstation and LCU modules respectively. Class diagrams show the classes of the system and their interrelationships together with the most important public methods. Moreover for each module the list of handled commands is illustrated.

The Overview chapter illustrates the component and deployment diagrams of the NGCOSW.

The public database structure of NGCOSW is listed in the Interfaces chapter.

The Processes Sequence Diagrams chapter presents an overview of the most common operational scenarios.

The Performance Analysis chapter provides a short description of the main performance bottlenecks in the system.

Finally the Traceability Matrix chapter traces NGCOSW requirements described in [AD7] to this design document's paragraphs.

2 Architecture

2.1 Overall description

NGC is the controller of the infrared and the optical scientific detectors for the ESO instruments.

While the low level software - operating system and drivers - can be common to both infrared and optical applications, as a consequence of the operational differences between infrared and optical instruments, at higher level the NGC software has been divided into the two different cases.

This document refers only to the NGC software specific to optical instruments (NGCOSW).

The NGC base software, which provides the access to the NGC hardware and is used by both the optical and infrared instruments, is described in [AD9].

The NGC software specific to infrared instruments (NGCIRSW) is described in [RD11].

NGCOSW is responsible for handling the optical detectors of the ESO instruments, interfacing to the ESO VLT environment, the NCG detector controller, different types of shutter controllers, temperature and pressure controllers.

NGCOSW incorporates all the functionalities of the FIERA controller software [RD15] and those of the IRACE controller software [RD16] which could be advantageous also for optical systems.

Modularity and proper design assure that the addition of any further functionality will not affect the existing ones.

The incompatibilities with the FIERA Controller software have been kept at the lowest reasonable level, with significant improvements in terms of code architecture and simplicity (maintainability) and system performances.

2.2 NGC hardware components

The NGC detector control system consists of one or more "embedded computers" or NGC Local Control Units (NGCLCUs) controlling one or more NGC Detector Front Ends (NGCDFEs, see [AD8]).

The NGCLCU has the following hardware components:

- One or more CPU(s)
- At least one back-end PCIbus board, responsible for the connection with the NGCDFE through fiber-optic link(s)

A NGCDFE - responsible for creating and receiving the detector signals - consist of:

- At least one front-end Basic Module
- an arbitrary number of front-end AQ Modules

All the boards contain a sequencer, which is responsible for issuing the voltages and clock signals to drive the detectors. A sequencer consists of a "clock pattern RAM" and a "program RAM". In the "clock pattern RAM" are stored the detector clock patterns, in the

"program RAM" are stored the sequences (or "sequencer programs") to be executed (LOOP and pattern EXEC tokens), i.e., the instructions about how to handle the patterns stored in the "clock pattern RAM". Voltage levels are set by the Clock- and DC-Voltage Driver CLDC (for more details, see [AD8] and [AD9]).

Figure 1 shows a typical hardware environment for NGC on an optical instrument.

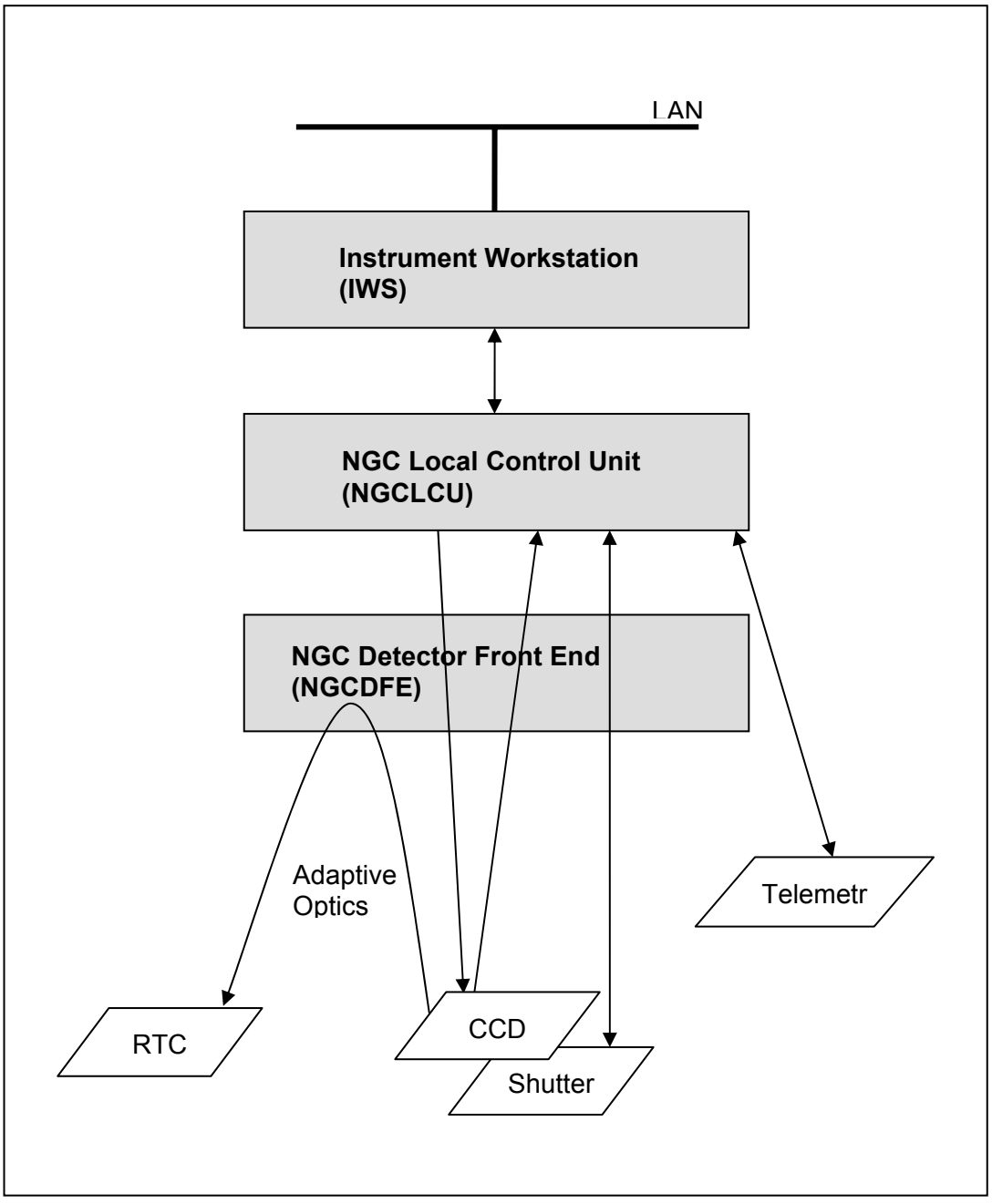

Figure 1 - NGC hardware environment

2.3 NGC software environment

NGCOSW is a software package developed to operate in the ESO VLT environment (see [AD26], [AD27] and [AD28]), running on platforms with Linux Operating System.

Figure 2 shows the interfaces of the NGCOSW with the ESO VLT environment.

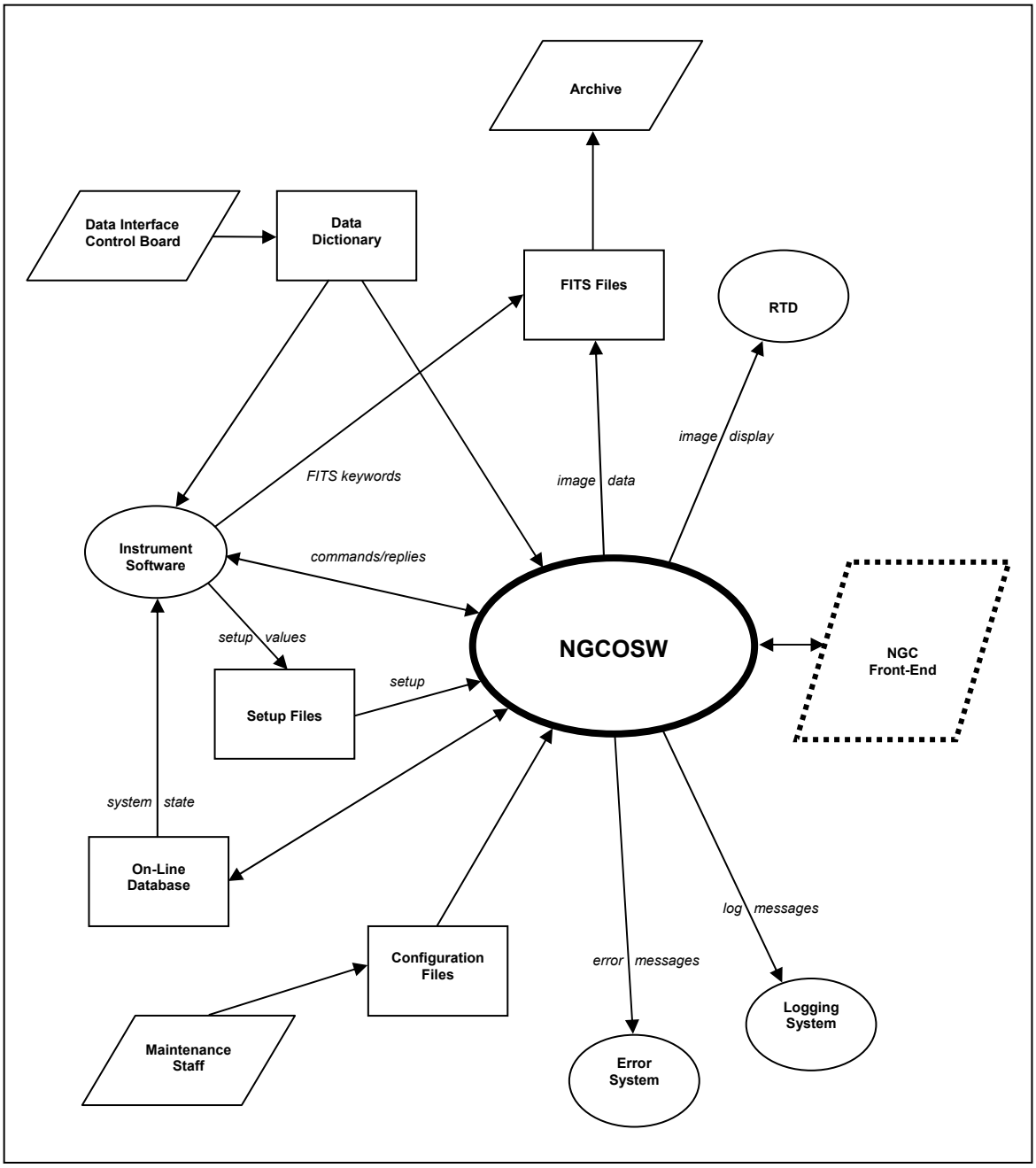

Figure 2 - NGCOSW in the ESO VLT environment

2.4 NGCOSW modules

NGCOSW is a distributed multi-process control software running on Instrument Workstations (IWSs) and NGCLCUs.

Figure 3 shows the modules used by NGC in the optical instrumentation. It represents the logical view of the system, i.e. the logical dependencies within the system. Modules common to optical and infrared instruments are in white (NGC Base Software [AD9]) and black boxes, modules belonging to NGCOSW are in the grey boxes.

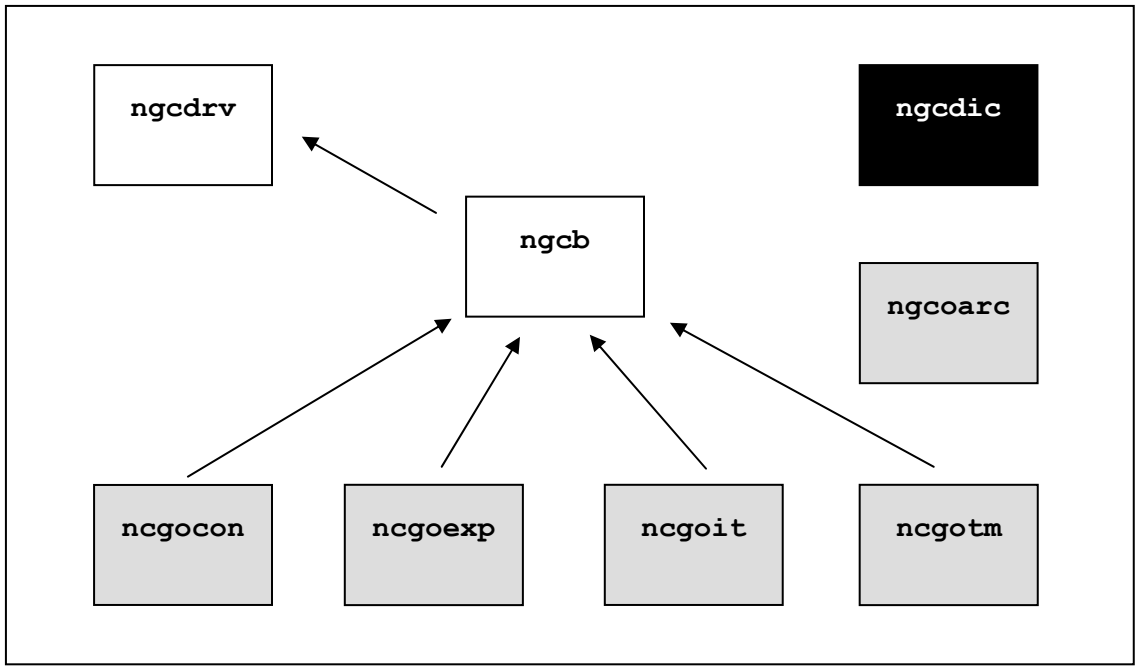

Figure 3 - System packages and their dependencies

The software modules which belong to the NGC Base Software [AD9]] and are common to both infrared and optical systems are:

- **ngcdrv**. NGC driver for the PCIbus back-end card
- **ngcb**. NGC base SW module

These software modules are described in [AD9]]. NGCOSW uses these modules to access the NGC detector electronics

A software module which is used by both infrared and optical systems is:

- **ngcdic**. NGC dictionary

It contains the dictionary common to both infrared and optical systems. Its description is outside the scope of this document.

The software modules which belong to the NGCOSW are:

- **ngcocon**. NGC system coordination module for optical instruments
- **ngcoexp**. NGC exposure handling module for optical instruments
- **ngcoit**. NGC image transfer module for optical instruments

- **ngcotm**. NGC telemetry module for optical instruments

These software packages - specific to optical systems - will be described in the following chapters of this document. Code and documentation design assures the maintainability of each module does not require the analysis of the other modules.

In addition to these modules, the NGCOSW contains also the **ngcoarc** module: it contains the include file for the NGCOSW version definition (for configuration control), the configuration file for pkginBuild usage [RD44] and all the scripts for the automatic VLTSW archive retrieval, generation and installation of the NGCOSW code.

As a support to software analysis/design and documentation, comments in the code of all NGCOSW modules are written in Doxygen format [RD53].

All NGCOSW modules are under CMM configuration control [RD42].

2.5 NGCOSW processes

Figures 4a, 4b and 4c illustrate some scenarios of the NGCOSW process deployment and interactions which could be reasonably used in the case of optical instruments.

In Figure 4a a system using one NGCLCU and more NGCDFEs is shown.

This figure puts in evidence the fact that a different online database environment (LCUENV) is used for each back-end board. In each LCUENV a pair of processes runs, to handle the exposure and the image transfer (see 2.5.2 and 4).

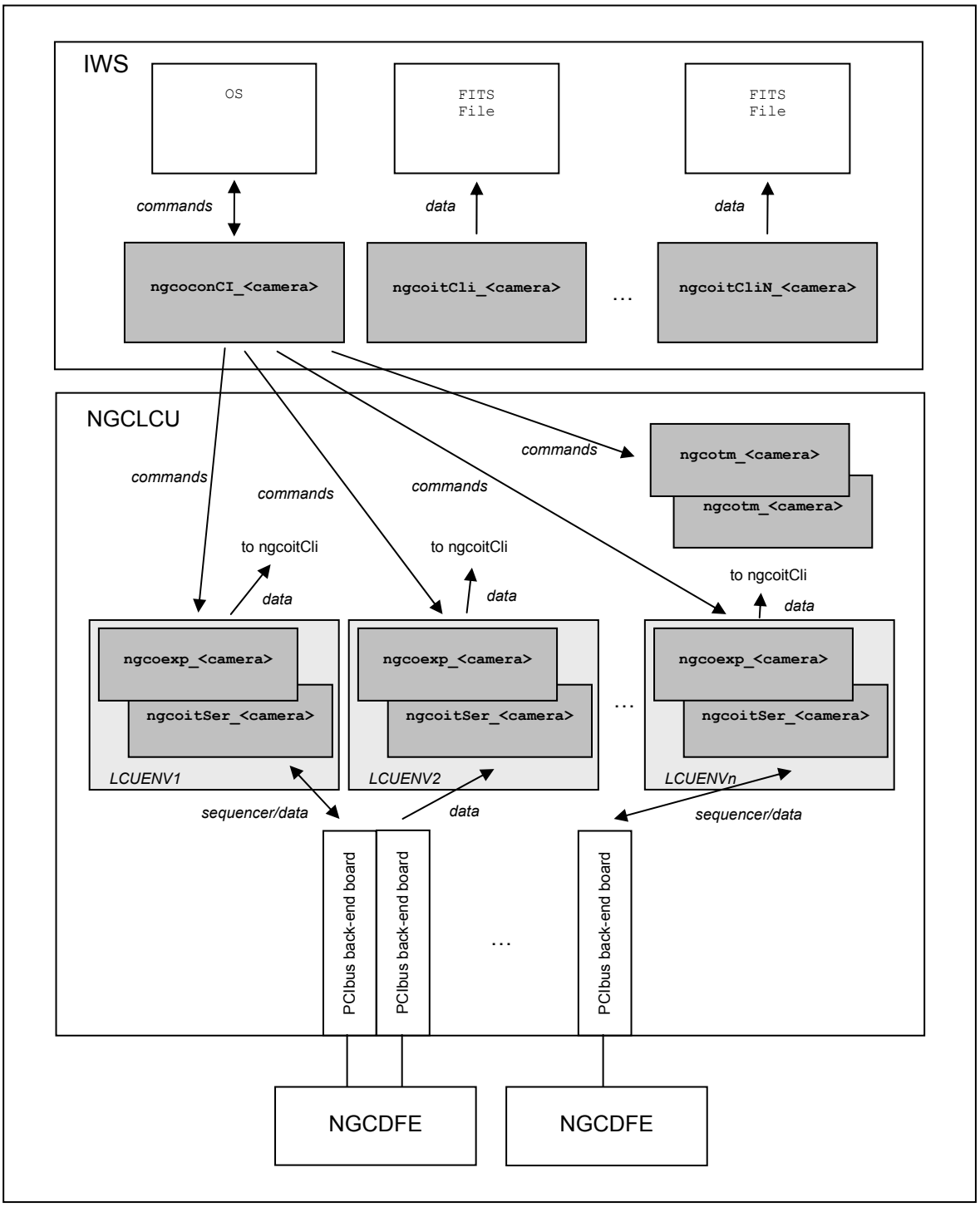

Figure 4a - Deployment and interactions diagram: one NGCLCU, more NGCDFEs

Figure 4b shows a system using more NGCLCUs and more NGCDFEs.

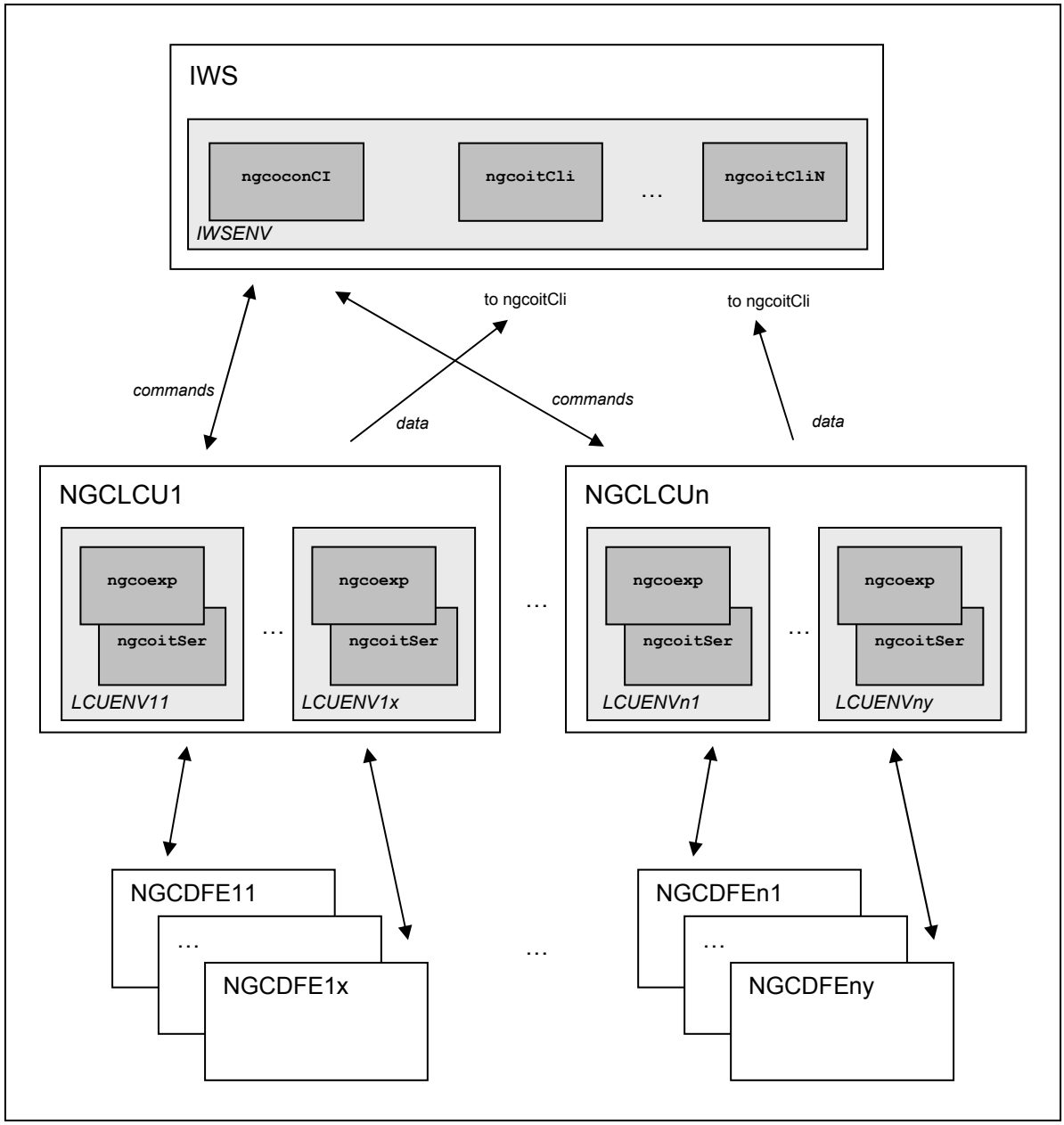

Figure 4b - Deployement and interactions diagram: more NGCLCUs, more NGCDFEs

Figure 4c shows a system using more NGCLCUs and NGCDFE. In this case, more front-end Basic Modules are used on the NGCDFE, and eoch NGCLCU is connected to a different one of them.

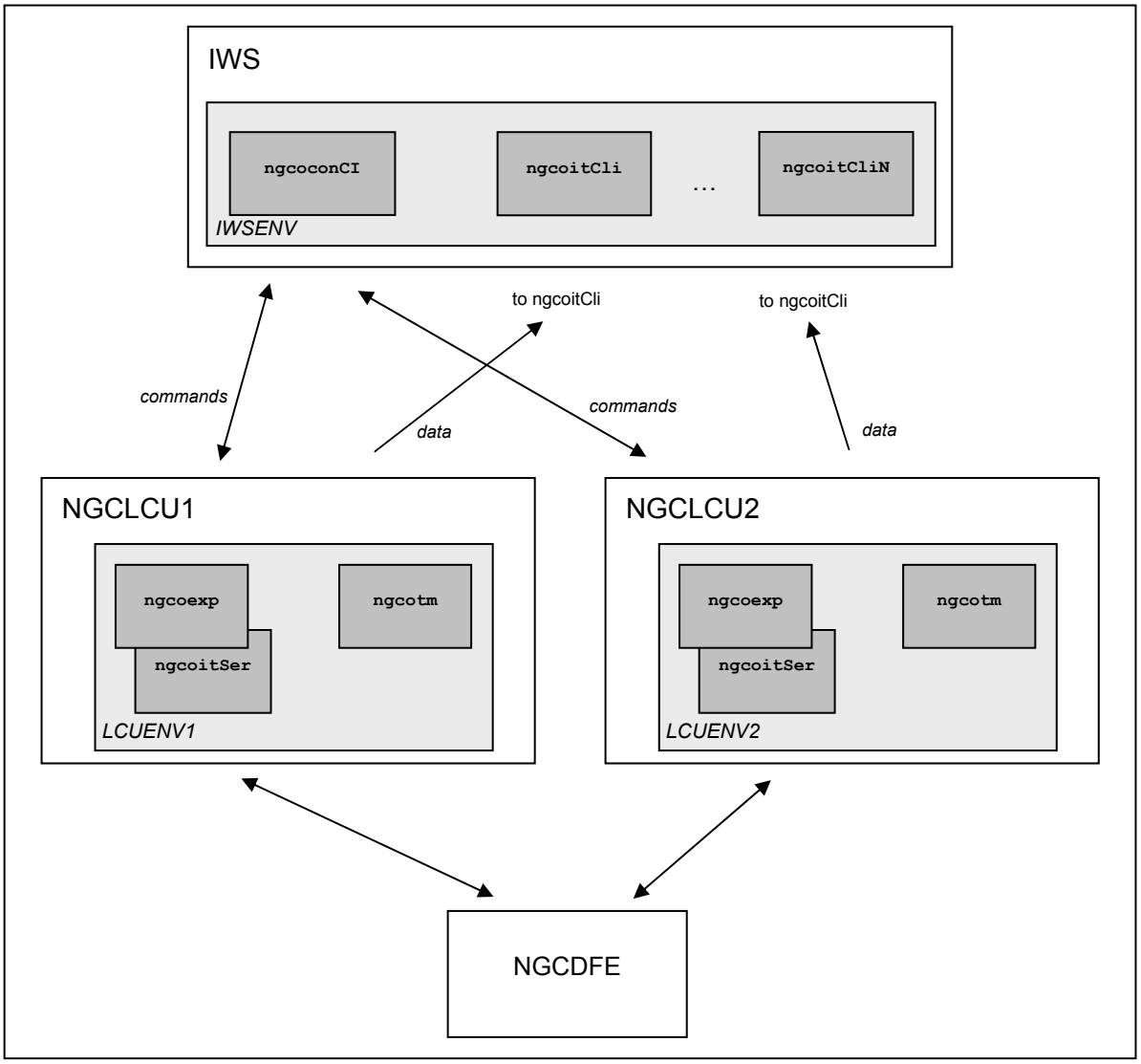

Figure 4c - Deployment and interactions diagram: more NGCLCUs, one NGCDFE

Communication between all NGCOSW processes and between the NGCOSW and the environment in which it operates are performed via the VLT CCS message system, logging system, error system, online database.

NGCOSW is command driven and is controlled by any "actor" (instrument software, operator, engineer, etc.) through the VLTSW message system (commands), error system, log system and the online database. In this way it is straightforward to directly interface to the NGCOSW from any process able to handle commands within the VLTSW environment, like BOB, DFS, general-purpose image processing programs. In particular, it is possible to emulate VLT-compatible instruments in the laboratory to the extent that VLT control software sequencer scripts can be executed.

NGCOSW can handle multiple independent detectors, by launching multiple instances, one for each detector. However, totally independent NGCDFEs can be controlled only if there is a dedicated back-end PCIbus board per NGCDFE.

Whenever superuser privelges are required, this is obtained without using explicit login, e.g., sticky bit, sudo operations, etc, so that the execution of the NGCOSW does not require any special user privilege to be granted by the operating system.

A more detailed overview of components and deployment will be given at the end of the description of all the modules (in Chapter 5).

2.5.1 Workstation processes

On the IWS - in the IWS online database environment IWSENV - the following processes run:

- ngcoconCI_<camera> is the Command Interface and handles the camera activities.
- one or more ngcoitCli<i>_<camera> (i=1,...,N), the Image Transfer Client(s), each handling the image transfer and storage at WS level. An Image Transfer Client receives image data from the Image Transfer Server(s) on the NGCLCU(s).

These processes are described in Chapter 3.

2.5.2 LCU processes

On the NGCLCU a different online database environment LCUENV runs for each back-end module which is installed. In each LCUENV the following processes run:

- ngcoexp_<camera>, the Exposure Control process, is responsible for the exposure control and the periodic wipe.
- one ore more ngcoitSer<i>_<camera> (i=1,...,N), the Image Transfer Server(s), each handling the image processing and transfer at NGCLCU level. On each NGCLCU there is one ngcoitSer<i>_<camera> process per back-end PCIbus Board. Each Image Transfer Client transmits the image data to one or more Image Transfer Clients on the IWS(s).

When needed, LCUENV hosts also the following process:

- ngcotm_<camera>, the Telemetry Control process, is responsible for the telemetry (setting and reading voltages and temperatures).

These processes are described in Chapter 4.

2.6 NGCOSW operational states

The NGCOSW can be in the following operational states (see [AD28]):

- **OFF**. The NGCOSW is OFF when it is not running. Consequently, the NGCOSW can never reply it is in the OFF state.
- **LOADED.** When the NGCOSW goes to LOADED state, the database is loaded and all processes are activated. Anyway the access to hardware is not allowed.
- **STANDBY.** The software and the hardware interfaces are initialized, all hardware

components are checked.

In detail all actions needed to bring the whole camera to STANDBY state are very dependent on the system hardware architecture and therefore cannot be defined in this document for all cameras. Typically the following actions are implemented:

a. Detector disconnected (voltages not applied).

b. Shutter control hardware is switched off, whenever the hardware architecture allows it.

c. Temperature control remains active

d. LAN connection active (command reception enabled)

- **ON-LINE**. This is the only state where the NGCOSW can perform exposures. All software and hardware is loaded, initialized and active. All voltages have been loaded. Telemetry has been acquired and checked. All the voltage switches are closed.

Figure 5 illustrates the NGCOSW operational states and the commands to switch between them (see [AD28]).

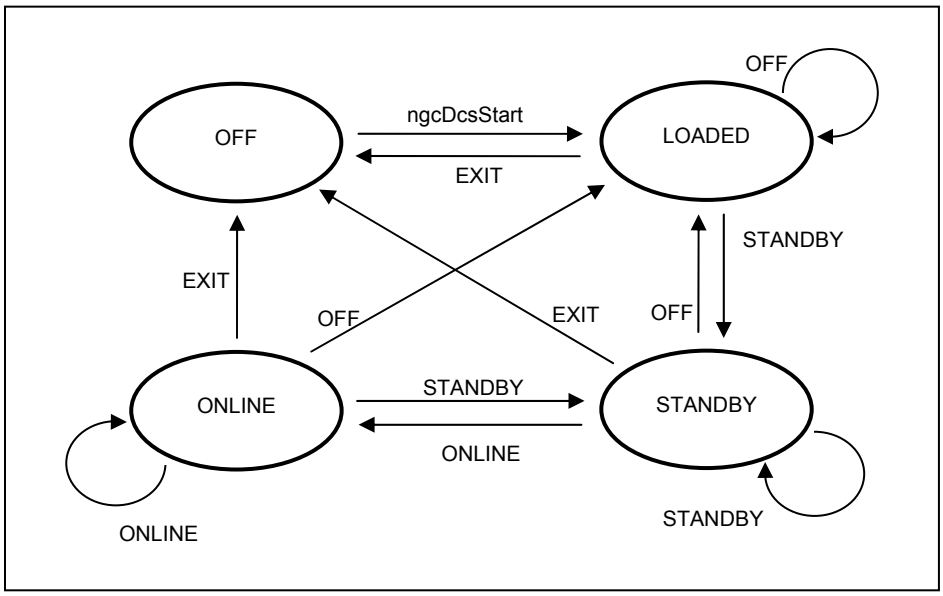

Figure 5 - Operational states and state transitions

2.7 Exposures

2.7.1 Exposure types

NGCOSW distinguishes among the different types of exposure defined in the Glossary [AD63].

2.7.2 Exposure status

The status of an exposure can be:

- NOT ACTIVE
- PENDING
- INTEGRATING
- PAUSED
- readout ACTIVE
- PROCESSING IMAGE DATA
- TRANSFERRING IMAGE DATA
- COMPLETED SUCCESSFULLY
- COMPLETED WITH ERROR
- ABORTED
- FINITE LOOP OF REPEATED EXPOSURES ACTIVE
- INFINITE LOOP OF REPEATED EXPOSURES ACTIVE
- WIPING

2.7.3 Parallelism

One of the main requirements for the NGCOSW is to optimize observation time, performing operations in parallel as much as possible.

For this reason, the portion of the image which has already been read out is transferred to the Workstation and saved in FITS file and/or displayed (depending from the setup), while the next image portion is being read out. This parallelism of readout and image transfer ensures that the time interval between the end of the readout and the completion of image display or FITS file on disk is minimized.

In case of highly demanding applications, implementations like multiple Image Transfer Clients on multile IWSs and/or multiple buffering will be taken into account.

2.7.4 Exposure Id

In order to be able to uniquely identify an exposure among those already finished and those running, an identification number (exposure Id) is associated to each exposure.

The exposure Id is passed to the NGCOSW as a parameter of the command START (as defined in [AD28]).

2.8 NGCOSW operational modes

NGCOSW operates in two different modes:

- **Instrument mode**

  This is the mode in which the NGCOSW operates when the NGC detector electronics are connected.

- **Simulation mode**

  In this mode the NGCOSW simulates the interactions with the NGC detector

electronics or other software processes (NGC image transfer, TCS, etc.)

This mode is used by the higher level OS software to test the interface with the NGCOSW during the development phase, when no NGC detector electronics or other software processes are available.

All the different NGCOSW processes can be independently set in instrument or simulation mode.

2.9 NGCOSW platform configuration

NGCOSW can operate in different scenarios:

- **Normal configuration**

  In normal operation, the NGCOSW is distributed on both the IWS and the NGCLCU, as shown in 2.5.

- **Hardware testing configuration**

  When operating in the laboratory with no IWS, all the NGCOSW runs on the NGCLCU, controlling the NGC detector electronics.

- **Software testing configuration**

  When a Instrument Team is developing the Instrument software and needs to test the interface with the NGCOSW with no NGCLCU available, all the NGCOSW runs on the IWS, simulating the interactions with the NGC detector electronics (see 2.8).

By using the ESO VLT message system, the system configuration (i.e., where the NGCOSW processes are running) is completely transparent to the actors (instrument software, operator, engineer, etc.), because the communications between the different processes are performed through the online database environments IWSENV and LCUENV, independently from the host where these are active.

In this way, always the same software is used in all the different scenarios, in order to guarantee system robustness, behaviour consistency, code development saving.

Summarising, these are possible operational scenarios:

- **Normal configuration, instrument mode**

  IWSENV runs on the IWS, LCUENV runs on the NGCLCU. NGC detector electronics are used.

- **Normal configuration, simulation mode**

  IWSENV runs on the IWS, LCUENV runs on the NGCLCU. NGC detector electronics are simulated.

- **Hardware testing configuration, instrument mode**

  IWSENV and LCUENV run on the NGCLCU. NGC detector electronics are used.

- **Hardware testing configuration, simulation mode**

  IWSENV and LCUENV run on the NGCLCU. NGC detector electronics are simulated.

- **Software testing configuration, simulation mode**

  IWSENV and LCUENV run on the IWS. NGC detector electronics are simulated.

2.10 Errors and logging

At all levels the NGCOSW uses the standard VLTSW error and logging systems to report important or critical information to the higher level control software ([AD32]).

Error severity levels are set in order to define the conditions when automatic recoveries from errors can be attempted or not.

Periodical (configurable) logging can be performed on request by processes monitoring the system (e.g., for telemetry).

2.11 Testing

For all the NGCOSW, automatic test scripts and programs are developed in parallel to the module code generation.

Regressive tat test scripts provide testing of all the commands defined in the module CDTs ([AD41]).

Regressive tat test scripts provide testing of all the usual possible scenarios which can appear during normal operations, including different configurations, hardware and software failures (NGC itself, network, lack of resources, access denial): TBD.

2.11.1 Data transmission testing

In order to test the image data path, NGC must be able to produce pre-defined data.

The data used to test the image data path can be generated at the level of the NGCDFE sequencers or inside the NGCLCU.

In the first case - useful for special applications like Adaptive Optics - data can be only simple patterns, due to the memory size of the NGCDFE sequencers. This case is fully transparent to the NGCOSW, because it corresponds to the normal operation of executing a readout sequence (which - in this case - does not read from the detector, but from the NGCDFE local memory, see 4.2.1).

In the second case, predefined data patterns or pre-stored image FITS images can be used, but the NGCOSW must be instructed on the data and readout simulation parameters (see 4.3.1).

2.12 Instrument specific modules

In addition to NGCOSW, all optical instruments are provided with an instrument module, ngc<instrument-name>, where all the specific configuration files, voltage tables and sequences (see 2.2) are stored.

At system installation, the configuration files, voltage tables and sequences are stored in the $INS_ROOT/$INS_USER/COMMON/CONFIGFILES directory.

Instrument modules are under CMM configuration control [RD42].

3 IWS Modules

3.1 Overview

The source code of NGC WS modules (with the exception of the GUI modules, system and test scripts) is written in C++ and it is documented by comments in Doxygen format.

GUI modules are developed using the Panel editor [RD39] and according to the ESO GUI conventions [AD38].

System scripts are written in VLTSW standard Unix shell programming language. Test scripts are written in TCL/TK.

NGC WS modules are:

- ngcocon module containing the Command Interface which handles all the messages (command/replies) to/from the camera
- ngcoit/ws submodule responsible for image transfer

3.2 Module ngcocon

3.2.1 Static structure and dynamic behavior

The ngcocon module contains:

- all the utilities used to interact with NGCOSW: the script used to startup the system and the Graphical User Interface (GUI) panels used by the engineers to configure and test all the functionalities and performances of a system, the utility to calculate off-line the basic offset values which optimize the bias equalization.
- the basic classes used to interface with the operating system.
- the Command Interface process

The ngcStartupCamera script is used to startup any camera for optical instruments: startup parameters as Camera Name, Online Database Environment, Operational Mode can be passed as script call parameters or environment variables (-inst or $CCDNAME, -env or $RTAPENV, -mode or $OPMODE).

The execution of the camera startup script shall not require more than 10 s for auto-recognition of the hardware and the ready-for-use initialization of hard- and software (it must still be clarified if some parallelism in the starting of all the NGCOSW processes is needed to reach this requirement, see Par. 8.1).

The GUI panels interact with the NGCOSW as any other "actor", i.e., through the VLTSW message system (commands) and the online database.

The GUI panels are used by the engineers to configure and test all the functionalities and performances of a system: to avoid misuse, their usage/operations could be password protected (TBD).

Their graphical aspect must be coordinated with the developers of the NGC software for the infrared detector ([RD11]), in order to avoid unnecessary differences.

The ngcoconOffset utility can be used off-line to calculate the basic offset values which

optimize the bias equalization: it performs a series of bias exposures with different offset values, and chooses the ones which best fit a required bias value, within a requested accuracy range.

All the operating system calls are encapsulated in the basic class (still TBD) used to interface with the operating system: in this way, future porting to different Unix flavors can be performed at reasonable cost.

The Command Interface process ngcoconCI_<camera> is responsible for handling the messages (command/replies) between the higher level software and the NGC processes. No direct command/reply exchange is foreseen between the higher level software and NGCOSW processes different from the Command Interface process.

The Command Interface process is based on the CCS Event Tool Kit EVH ([RD32] [RD33]). It is an instance of the CommandDispatcher class, interfacing with the other processes through the CommandInterface class and with the Online Database through the DatabaseInterface class.

Figure 6 shows the Class Diagram of the Command Interface.

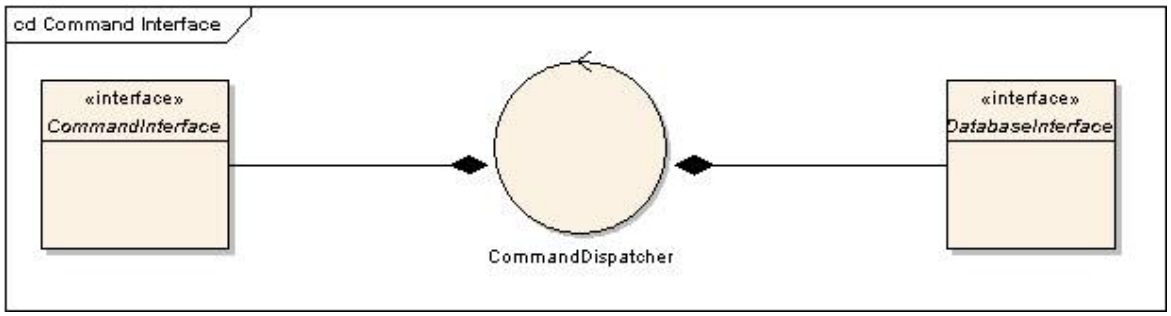

Figure 6 - Logical view of the Command Interface.

3.2.2 Command interface

Table 3 lists the commands accepted by ngcoconCI_<camera>.

These commands are listed in the ngcoconCI.cdt table, which represents the only Command Interface available to the higher level software.

Command	Parameters	Description
ABORT	expoId	Abort exposure with ID <expoId> (default last started exposure).
BREAK	-	Break execution of current command
CONT	at (format HH:MM:SS.TTT)	Continue a paused exposure at a given time (default "now").
CONFIG	-	Read again configuration of the system
DUMP	-	Dump the image in memory to disk
END	-	End the current exposure(s) and read out the data.
EXIT	-	Bring the system to operational state OFF and terminate it.
INIT	function	Initialize the functions contained in the list of arguments (default is "all").
KILL	-	Kill this process.
MSGDLOG	-	Disable autologging of messages sent or received

		by the application
MSGELOG	-	Enable autologging of messages sent or received by the application
OFF	-	Bring the system to operational state LOADED.
ONLINE	-	Bring the system to operational state ONLINE.
PAUSE	at (format HH:MM:SS.TTT)	Pause exposure at a given time (default "now").
PING	-	Verify whether this process is able to send or receive messages
SELFTST	function	Execute a self-test (sw and hw) of the specified function(s).
SETUP	expoId/file/function	Set-up for the exposure with ID <expoId> (default next exposure), taking it from file, ore defining it from the parameter function
SIM/SIMULAT	function	Put the processes contained in list of arguments into simulation mode.
STANDBY	-	Bring the system to operational state STANDBY.
START	at (format HH:MM:SS.TTT)	Start an exposure at a given time (default "now").
STARTLP	at (format HH:MM:SS.TTT)	Start an infinite loop of repeated exposures at a given time (default "now").
STARTTL	period/logperiod	Start monitoring of telemetry values.
STARTWP	periodic	Wipe chip(s) once or periodically.
STATUS	expoId/function	Get the status of the functions in the list of arguments
STOPLP	-	Stop an infinite loop of repeated exposures, at the end of the running exposure.
STOPSIM	function	Stop the simulation for the functions contained in the list of arguments.
STOPTL	-	Stop monitoring of telemetry values.
STOPWP	-	Stop a periodic wipe.
VERBOSE	on/off	Set verbose mode on/off.
VERSION	-	Return current version of the NGC sw.
WAIT	expoId/waitMode	Wait for exposure completion and return exposure status. Parameter expoId indicates the ID of exposure to wait for (see START). 0 means: last started exposure. Parameter waitMode can have values: "Single": wait until the current repetition is completed "Global": wait until all repetitions are completed. It makes sense only if the setup parameter DET.EXP.NREP is > 1.

Table 3 - Input commands for the Command Interface process

3.2.3 Testing and simulation support

TBD

3.3 Module ngcoit

3.3.1 Static structure and dynamic behavior

The ngcoit module contains the Image Transfer processes: the Image Transfer Server (see 4.3) and the Image Transfer Client (see below).

The Image Transfer Client process ngcoitCli<i>_<camera> receives the image data from one or more Image Transfer Server process(es) ngcoitSer_<camera> - running on the NGCLCU(s) - stores the data in memory (for rapid display by the RTD) and saves them into FITS files, together with all relevant system hardware and software configuration information and telemetry.

If requested, the Image Transfer Client process performs also some image processing, when this requires handling of several image files and cannot therefore be performed by the Image Transfer Server on the NGCLCU (e.g., averaging of frames with and without removal of outliers).

Image data transmission and display are performed in parallel.

If requested only one frame every n can be saved into a FITS file.

If requested data cubes containing n successive frames can be saved into a single FITS file.

The Image Transfer Client is responsible for handling and storing only the received data, without any further knowledge of the detector system.

In the usual configurations, only one Image Transfer Client process runs on an IWS, but - depending on the instrument topology - on the same IWS it is possible to run more Image Transfer Clients in parallel.

The Image Transfer Client process ngcoitCli<i>_<camera> is an instance of the ImageTransferClient class, interfacing with the Command Interface processes through the CommandInterface class and with the Online Database through the DatabaseInterface class.

The DxfInterface and Image classes are used to receive and handle the image data.

Figure 7 shows the Class Diagram of the Image Transfer Client.

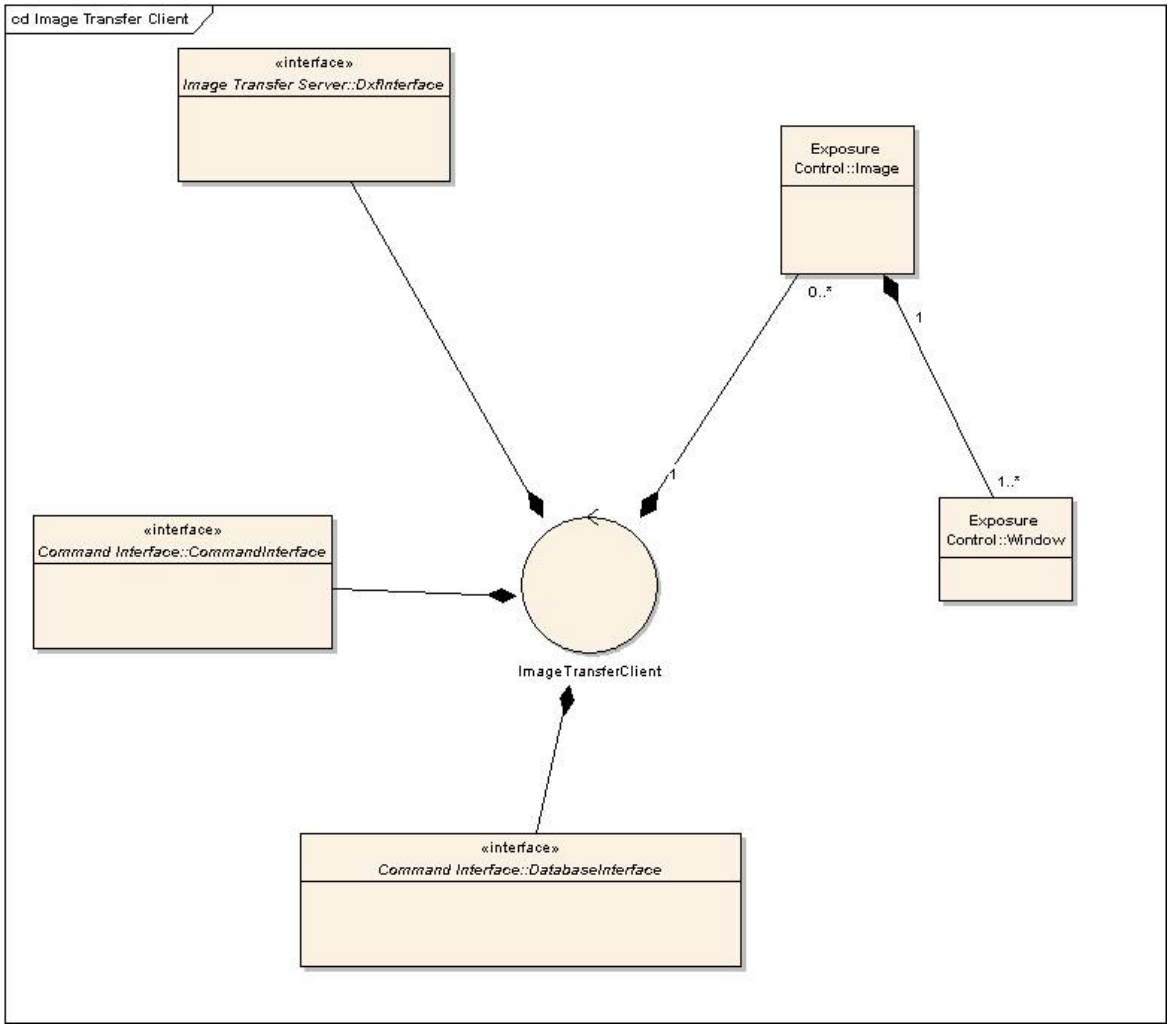

Figure 7 - Logical view of the Image Transfer Client.

3.3.2 Command Interface

Table 4 lists the commands accepted by ngcoitCli<i>_<camera>.

These commands are listed in the ngcoitCli.cdt table, which is not available to the higher level software.

Command	Parameters	Description
BREAK	-	Break execution of current command
EXIT	-	Bring this process to operational state OFF and terminate it.
INIT	-	Initialize this process.
KILL	-	Kill this process.
MSGDLOG	-	Disable autologging of messages sent or received by the application
MSGELOG	-	Enable autologging of messages sent or received by the application
OFF	-	Bring this process to operational state LOADED.
ONLINE	-	Bring this process to operational state ONLINE.

PING	-	Verify whether this process is able to send or receive messages
SELFTST	function	Execute a self-test (sw and hw) of the specified function(s).
SIM/SIMULAT	function	Put the processes contained in list of arguments into simulation mode.
STANDBY	-	Bring this process to operational state STANDBY.
STATUS	expoId/function	Get the status of the functions in the list of arguments
STOPSIM	function	Stop the simulation for the functions contained in the list of arguments.
VERBOSE	on/off	Set verbose mode on/off.
VERSION	-	Return current version of the process.

Table 4 - Input commands for the Image Transfer Client

3.3.3 Data interfaces

Image data transmission between NGCLCU and IWS is performed using DXF, as defined in [RD36].

Image data are delivered to the ESO RTD following [RD40].

Image data are stored as FITS files compliant with [AD37].

3.3.4 Testing and simulation support

TBD

4 NGCLCU Modules

4.1 Overview

The source code of NGC LCU modules is written in C++ and it is documented by comments in Doxygen format.

NGC LCU modules are:

- ngcoexp module responsible for exposure handling
- ngcoit/lcu submodule responsible for image transfer
- ngcotm module responsible for telemetry

4.2 Module ngcoexp

4.2.1 Static structure and dynamic behavior

The ngcoexp module contains the Exposure Control process.

The Exposure Control process ngcoexp_<camera> is responsible for the Exposure Control and the periodic wipe. This is performed by accessing the NGC detector electronics to run the sequences (see 2.2), the Exposure Control process being the only process accessing this lowest level of the NGC.

Reloading/upgrading any sequence can be performed without having to restart the whole software.

It must be noted that the Exposure Control process does not really know about the actions performed by the sequences that it has to run. This means that - for cases like testing the interface with Adaptive Optics systems - special readout sequences can be used to transmit predefined data patterns, previously stored inside the NGCDFE sequencer memory (see 2.11.1). Simulation parameters - like transmission speed - are passed via setup.

Before going ONLINE, the Exposure Control process is responsible for instructing the NGC hardware to execute a comprehensive self-test (also on external devices like shutter and telemetry controllers), reporting its status. Unique identification numbers read from NGC Back and Front End boards and - if possible - detectors allow automatic configuration checking against the instrument configuration description (provided by the Instrument Specific Module, see 2.12).

TBC: The Exposure Control process can interface to the Exposure Control process of other instances of the NGCOSW running on the same NGC in order to drive the execution of commands and sequences of multiple detectors controlled by a single NGC simultaneously (See [AD20], Req. AONGCREQ-005: is this what is requested, or does this requirement refer to mosaics or mosaic-like systems, i.e., detectors which are installed in different detector heads, but are always driven together, and therefore, from the software point of view, can be seen as a mosaic?).

The Exposure Control process checks the availability of critical resources before starting an exposure. It provides a generic interface to shutter controller devices (e.g., Pulpo [RD18], PULPO2 [RD19], LCU, etc.), including shutters with more moving parts and

bidirectional shutters.

Exposures can or cannot require shutter operation (e.g., normal or dark): as a consequence, the Exposure Control process defines the start of an exposure by the shutter motion (normal exposures) or by the end of the detector wiping (dark exposures).

The Exposure Control process is responsible also for driving an external light source, if required (like in the case, for instance, of HARPS).

The binning factors can be freely defined, with independent values in the vertical and horizontal dimensions.

There is no limit on the number of windows which can be read.

The Exposure Control process can handle finite or infinite loops of exposures or readout sequences, with user definable delays (in microseconds) between executions. The loops are terminated by a stop command, which is executed at the completion of the running cycle.

Window definitions (coordinates of the starting point and dimensions) can be dynamically modified during a loop of readouts.

If the time between end of detector readout and availability of the FITS file on disk becomes a significant overhead, the Exposure Control process can be instructed to start an exposure right after the end of the transmission of the image date of the previous exposure to the Image Transfer Client.

The default behavior is to wait till the end of the image saving on disk.

Sequences can be executed also during exposures, if required.

On-chip charge shifts by a user-definable amount (e.g., for through-focus sequences) is supported.

The Exposure Control process does not put any limit to the number of images produced by an exposure, in order to cover cases like, for instance, Adaptive Optics (when all image data are passed directly to the Real Time Computer) or drift scanning (where one exposure produces many FITS files).

The Exposure Control process is responsible for fulfilling some of the most time critical system requirements, like handling the starting of an exposure and the setting of all time stamps with an accuracy of - at least - 0.1s and handling time intervals (e.g., for shutter open and delay times) with an accuracy of - at least - 0.1% or - for intervals shorter than 10s - better than 0.01s (see Par. 8.2).

The Exposure Control process provides an interface to TIM modules, for the cases when its usage is required.

In order to support several techniques that require the usage of an external signal (for synchronisation with other NGCs - like in NGC clusters - or with mirror units - like in *chopping* - or with the telescope - like in *nod and shuffle* - or with the shutter - like in *high-time resolution imaging and spectroscopy* - see Glossary), the Exposure Control can instruct the NGCDFE sequencers to synchronise the execution of the sequences with an external trigger.

The Exposure Control process ngcoexp_<camera> is an instance of the ExposureController class, interfacing with the Command Interface processes through the CommandInterface class and with the Online Database through the DatabaseInterface

class.

The BaseSoftwareInterface class access the NGCDFE through the NGC Base Software (see [AD9]]), while the Shutter is controlled via the ShutterInterface class. In both classes the proper usage of semaphores protects from conflicts by multiple accesses to the external resources.

The OslxInterface class is used to handle the exposure Setup.

Information about the Camera system and the Exposure configuration is handled using the Detector class, which is composed by the CCD, Output and HardwareSetup classes.

The Image class is used to inform the Image Transfer processes about the image data which will be generated by the exposure.

Figure 8 shows the Class Diagram of the Exposure Control process.

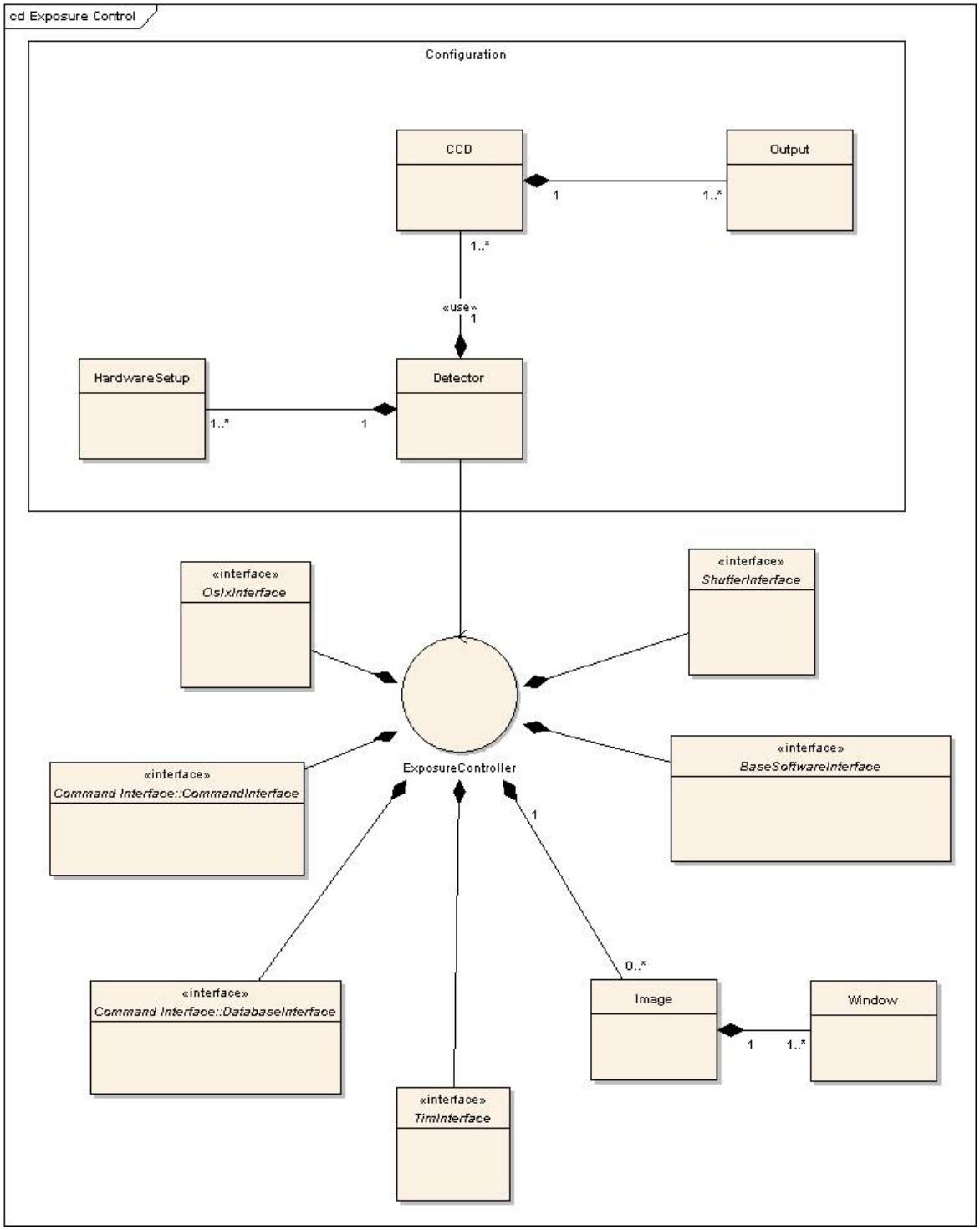

Figure 8 - Logical view of the Exposure Control.

4.2.2 Command interface

Table 5 lists the commands accepted by ngcoexp_<camera>.

These commands are listed in the ngcoexp.cdt table, which is not available to the higher

level software.

Command	Parameters	Description
ABORT	expoId	Abort exposure with ID <expoId> (default last started exposure).
BREAK	-	Break execution of current command
CONT	at (format HH:MM:SS.TTT)	Continue a paused exposure at a given time (default "now").
CONFIG	-	Read again configuration of the system
END	-	End the current exposure(s) and read out the data.
EXIT	-	Bring this process to operational state OFF and terminate it.
INIT	-	Initialize this process.
KILL	-	Kill this process.
MSGDLOG	-	Disable autologging of messages sent or received by the application
MSGELOG	-	Enable autologging of messages sent or received by the application
OFF	-	Bring this process to operational state LOADED.
ONLINE	-	Bring this process to operational state ONLINE.
PAUSE	at (format HH:MM:SS.TTT)	Pause exposure at a given time (default "now").
PING	-	Verify whether this process is able to send or receive messages
SELFTST	function/repeat	Execute a self-test (sw and hw) of the specified function(s).
SETUP	expoId/file/function	Set-up for the exposure with ID <expoId> (default next exposure), taking it from file, ore defining it from the parameter function
SIM/SIMULAT	function	Put the processes contained in list of arguments into simulation mode.
STANDBY	-	Bring this process to operational state STANDBY.
START	at (format HH:MM:SS.TTT)	Start an exposure at a given time (default "now").
STARTLP	repetition/at (format HH:MM:SS.TTT)	Start a loop of repeated exposures (repetition=0 means infinite) at a given time (default "now")
STARTWP	periodic	Wipe chip(s) once or periodically.
STATUS	expoId/function	Get the status of the functions in the list of arguments
STOPLP	-	Stop a loop of repeated exposures, at the end of the running exposure.
STOPSIM	function	Stop the simulation for the functions contained in the list of arguments.
STOPWP	-	Stop a periodic wipe.
VERBOSE	on/off	Set verbose mode on/off.
VERSION	-	Return current version of this process.
WAIT	expoId/waitMode	Wait for exposure completion and return exposure status. Parameter expoId indicates the ID of exposure to wait for (see START). 0 means: last started exposure. Parameter waitMode can have values:

		"Single": wait until the current repetition is completed "Global": wait until all repetitions are completed. It makes sense only if the setup parameter DET.EXP.NREP is > 1.

Table 5 - Input commands for the Exposure Control process

4.2.3 Testing and simulation support

TBD

4.3 Module ngcoit

4.3.1 Static structure and dynamic behavior

The ngcoit module contains the Image Transfer processes: the Image Transfer Server (see below) and the Image Transfer Client (see 3.3)

The Image Transfer Server process ngcoitSer_<camera> performs the image reordering (line by line), the offset correction (per readout channel) and - if requested - software-based windowing and/or the image processing (centroiding, max/min/fwhm), then transmits the image data to one or more Image Transfer Client process(es) ngcoitCli<i>_<camera>.

Centroiding algorithms from the VLTSW (technical CCDs) are used, since they have already shown to be able to reach the desired rates (see Par. 8.3).

The image data is kept in memory till the next readout has started, so that - in case of image data saving failures - the image data can be resent to the Image Transfer Client(s).

The parameters for readout simulation - when predefined data patterns or pre-stored FITS images are transmitted to the Image Transfer Client at a configurable transmission rate (see 2.11.1) - are passed to the Image Transfer Server via setup.

If data-transfer or processing rates exceed the capacity of one single computer, the Image Transfer Server is responsible for splitting the image transfer over several Image Transfer Clients running on different computing units.

If necessitated by high frame rates, only one frame every n can be passed to the Image Transfer Server for display and/or storage.

There is only one Image Transfer Server process per NGCLCU.

The Image Transfer Server process ngcoitSer_<camera> is an instance of the ImageTransferServer class, interfacing with the Command Interface processes through the CommandInterface class and with the Online Database through the DatabaseInterface class.

The Image class is used to handle the image data.

The DxfInterface class is used to transmit the image data, using the standard LVTSW package dfx, which has shown to cope with the required data transfer rates (see [RD22], [RD36] and Par. 8.4).

Figure 9 shows the Class Diagram of the Image Transfer Server.

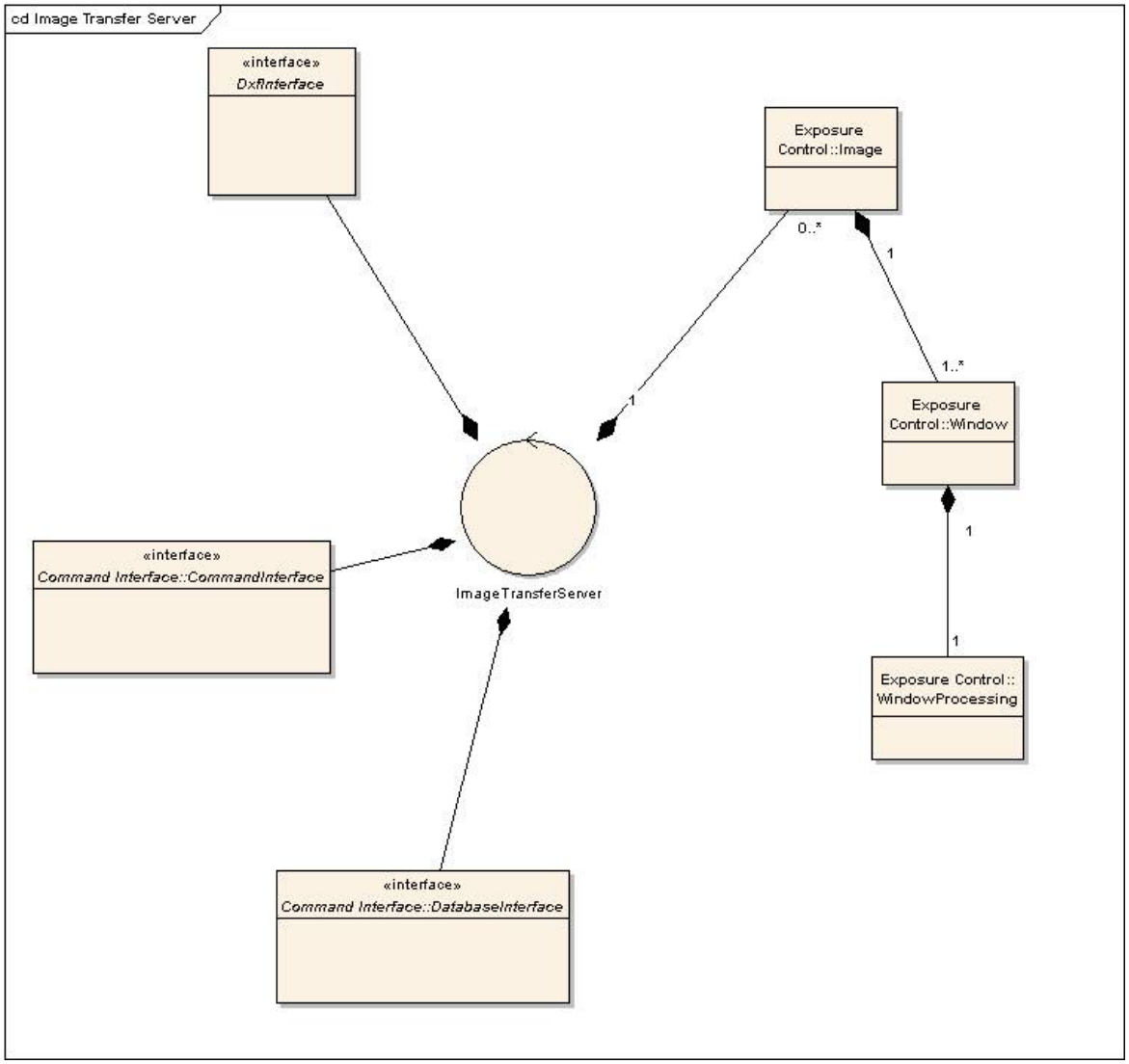

Figure 9 - Logical view of the Image Transfer Server.

4.3.2 Command interface

Table 6 lists the commands accepted by ngcoitSer_<camera>.

These commands are listed in the ngcoitSer.cdt table, which is not available to the higher level software.

Command	Parameters	Description
BREAK	-	Break execution of current command
DUMP	-	Dump the image in memory, retrasmitting it to the IWS
EXIT	-	Bring this process to operational state OFF and terminate it..
INIT	-	Initialize this process.
KILL	-	Kill this process.
MSGDLOG	-	Disable autologging of messages sent or received by the application

MSGELOG	-	Enable autologging of messages sent or received by the application
OFF	-	Bring this process to operational state LOADED.
ONLINE	-	Bring this process to operational state ONLINE.
PING	-	Verify whether this process is able to send or receive messages
SELFTST	function/repeat	Execute a self-test (sw and hw) of the specified function(s).
SIM/SIMULAT	function	Put the processes contained in list of arguments into simulation mode.
STANDBY	-	Bring this process to operational state STANDBY.
STATUS	expoId/function	Get the status of the functions in the list of arguments
STOPSIM	function	Stop the simulation for the functions contained in the list of arguments.
VERBOSE	on/off	Set verbose mode on/off.
VERSION	-	Return current version of this process.

Table 6 - Input commands for the Image Transfer Server

4.3.3 Data interfaces

Image processing (centroiding, etc) data are delivered to TCS via the Online Database, as defined in [RD21] (see 6.2).

Image data transmission between NGCLCU and IWS is performed using DXF, as defined in [RD22].

4.3.4 Testing and simulation support

TBD

4.4 Module ngcotm

4.4.1 Static structure and dynamic behavior

The ngcotm module contains the Telemetry Control process.

The Telemetry Control process ngcotm_<camera> is responsible for setting and reading an unlimited number of system telemetry values (e.g, cryostat pressures, system temperatures with the related alarms, and so on), by operating one or more pulpo[RD18], PULPO2 [RD19] or other similar devices.

Telemetry setting can be performed without having to restart the whole software and even during readout. Before telemetry setting, values are checked to guarantee the detector safety conditions.

The Telemetry Control process writes the telemetry values in the online database, to make them available to the Image Transfer Client process, which will store them in the header of the image data FITS files.

If requested, the Telemetry Control periodically sends the telemetry to the VLTSW logging system. The time interval can be configured by the user (default is TBD).

The Telemetry Control process ngcotm_<camera> is an instance of the

TelemetryController class, interfacing with the Command Interface processes through the CommandInterface class and with the Online Database through the DatabaseInterface class.

The temperature and pressure sensors and alarms are controlled via the SensorInterface class. The proper usage of semaphores protects from conflicts by multiple accesses to the external resources.

Figure 10 shows the Class Diagram of the Telemetry Control process.

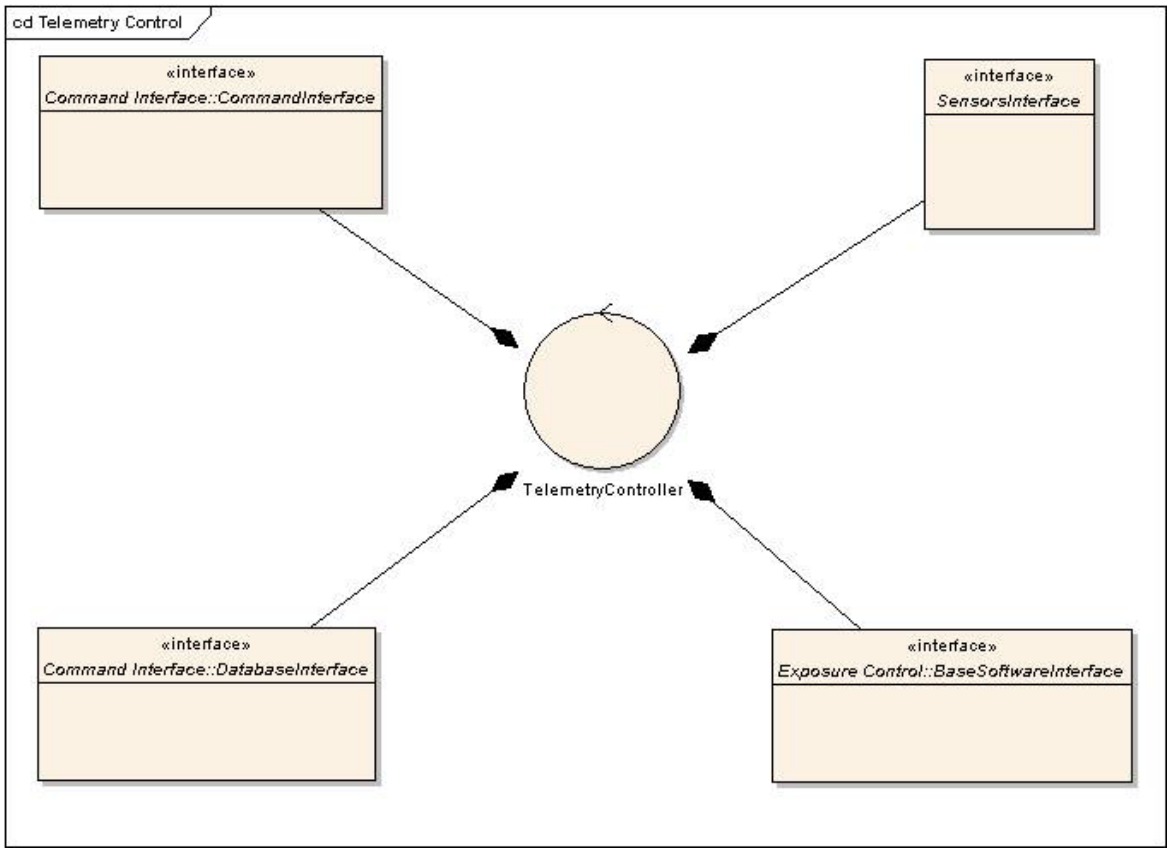

Figure 10 - Logical view of the Telemetry Control.

4.4.2 Command interface

Table 7 lists the commands accepted by ngcotm_<camera>.

These commands are listed in the ngcotm.cdt table, which is not available to the higher level software.

Command	Parameters	Description
BREAK	-	Break execution of current command
EXIT	-	Bring this process to operational state OFF and terminate it..
INIT	-	Initialize this process.
KILL	-	Kill this process.
MSGDLOG	-	Disable autologging of messages sent or received by the application
MSGELOG	-	Enable autologging of messages sent or received

		by the application
OFF	-	Bring this process to operational state LOADED.
ONLINE	-	Bring this process to operational state ONLINE.
PING	-	Verify whether this process is able to send or receive messages
SELFTST	function/repeat	Execute a self-test (sw and hw) of the specified function(s).
SIM/SIMULAT	function	Put the processes contained in list of arguments into simulation mode.
STANDBY	-	Bring this process to operational state STANDBY.
STARTTL	period/logperiod	Start monitoring of telemetry values.
STATUS	expoId/function	Get the status of the functions in the list of arguments
STOPSIM	function	Stop the simulation for the functions contained in the list of arguments.
STOPTL	-	Stop monitoring of telemetry values.
VERBOSE	on/off	Set verbose mode on/off.
VERSION	-	Return current version of this process.

Table 7 - Input commands for the Telemetry Control Process

4.4.3 Testing and simulation support

TBD

5 Overview of the NGCOSW

Figure 11 shows the component diagram of the NGCOSW, i.e., how the different processes are interconnected.

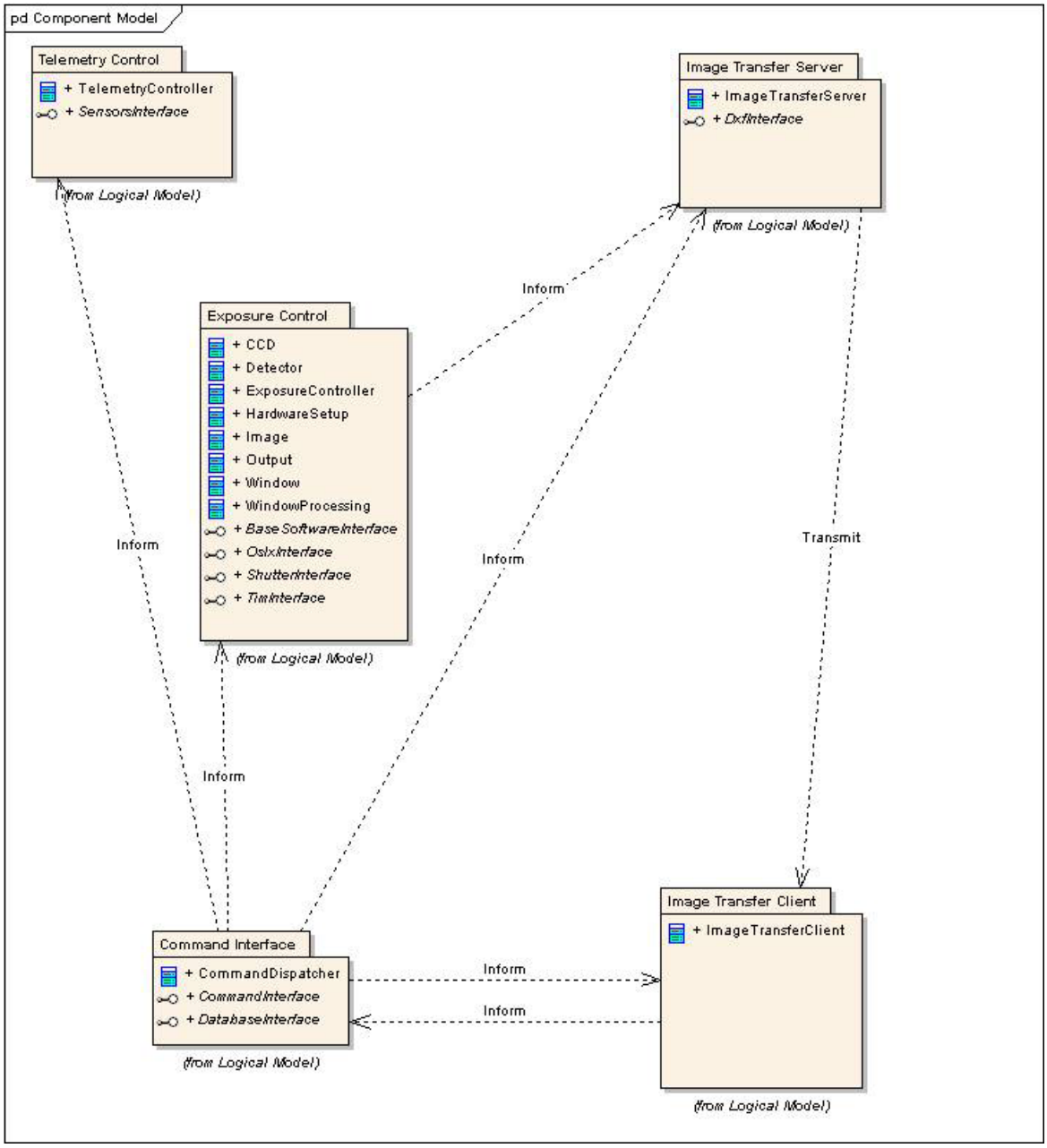

Figure 11 - Component diagram of the NGCOSW.

Figure 12 and Figure 13 show the detailed deployment diagram of the NGCOSW, i.e., where the different processes are located

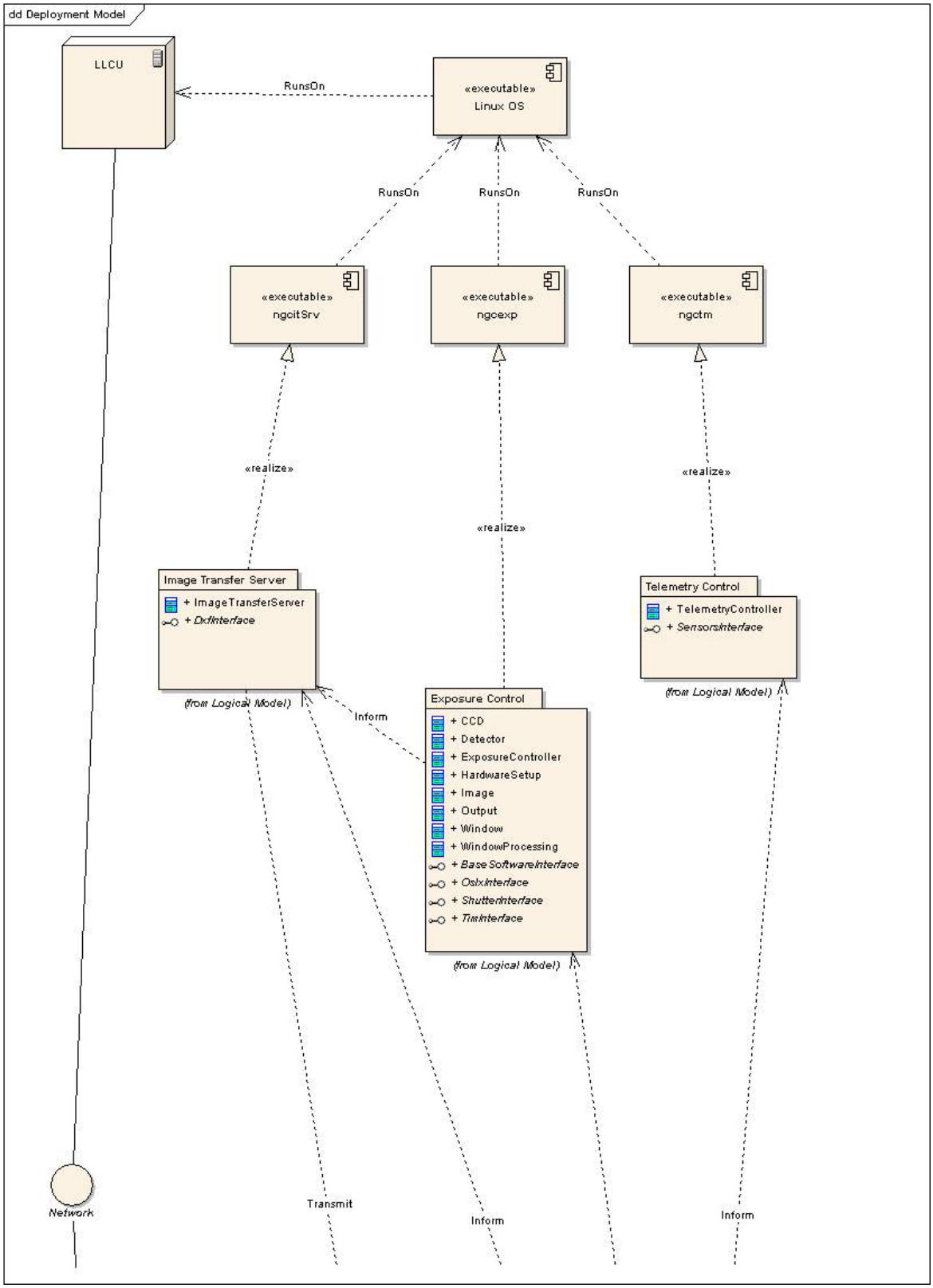

Figure 12 - Detailed deployment diagram of the NGCLCU part of the NGCOSW.

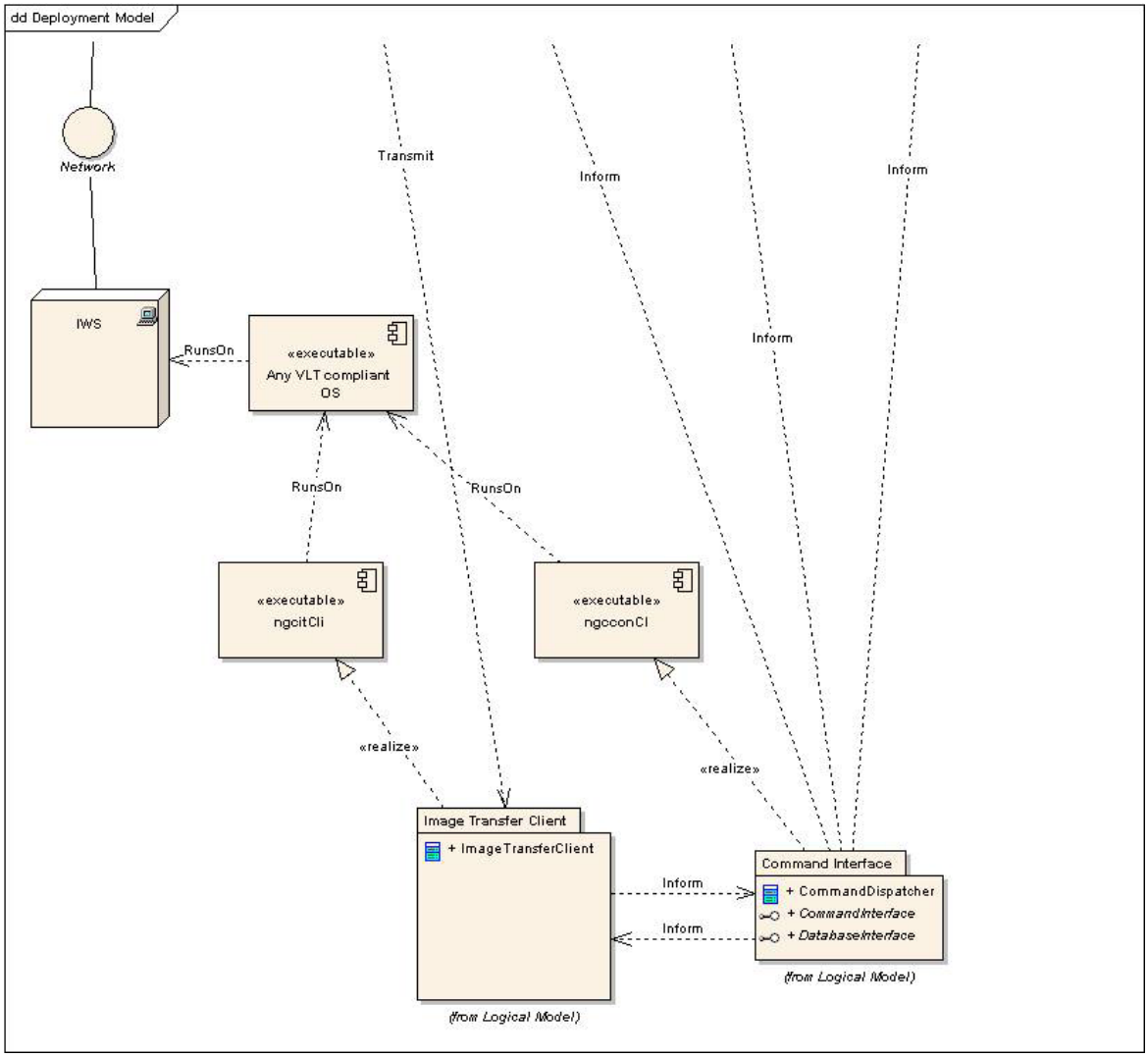

Figure 13 - Detailed deployment diagram of the Workstation part of the NGCOSW.

6 Interfaces

6.1 Interface between NGCOSW and the external environment

Any "actor" (instrument software, operator, engineer) interacts with NGC through the Command Interface process ngcoconCI_<camera>. All the commands accepted by the Command Interface process are listed in Table 3.

Errors are reported via the CCS error systems.

Diagnostic is performed via the CCS logging systems. The level of log details can be defined by setting different log levels.

The information about the state of NGC - hardware and software processes - is stored in the following online database attributes:

- **Read/write attributes**

  system.opMode (dbINT32) Camera operational mode

  system.imageDirectory (dbBYTES128) Directory where images are stored

  detector:chips:chip_<i>.crpix1 (dbINT32) Value for the CRPIX1 FITS keyword. If this attribute is set to ngcNO_DBCRPIX (99999), the value for CRPIX1 will be computed by NGC.

  detector:chips:chip_<i>.crpix2 (dbINT32) Value for the CRPIX2 FITS keyword. If this attribute is set to ngcNO_DBCRPIX (99999), the value for CRPIX2 will be computed by NGC.

- **Read only attributes**

  system.opState (dbINT32) System operational state.

  system.setupDirectory (dbBYTES128) Setup files directory.

  detector.description (dbBYTES32) Detector description.

  detector.installation (dbBYTES32) Detector installation date.

  detector:chips:chip_<i>.xLocation (dbINT32) Horizontal location of chip in mosaic (n=0,1,…).

  detector:chips:chip_<i>.yLocation (dbINT32) Vertical location of chip in mosaic (n=0,1,…).

  detector:chips:chip_<i>.xPixels (dbINT32) Number of columns.

  detector:chips:chip_<i>.yPixels (dbINT32) Number of rows.

  detector:chips:chip_<i> xPixelSize (dbINT32) Horizontal pixel size (microns).

  detector:chips:chip_<i>.yPixelSize (dbINT32) Vertical pixel size (microns).

  detector:chips:chip_<i>.numOutput (dbINT32) Number of outputs.

  detector:chips:chip_<i>:output_<j>.id (dbINT32) Output Id.

  detector:chips:chip_<i>:output_<j>.xLocation (dbINT32) Horizontal location of output in chip (n=0,1,…).

detector:chips:chip_<i>:output_<j>.yLocation (dbINT32) Vertical location of output in chip (n=0,1,…).

detector:chips:chip_<i>:output_<j>.prescan (dbINT32) Number of physical prescan pixels.

detector:chips:chip_<i>:output_<j>.leftToRight (dbLOGICAL) Horizontal readout direction, defining the direction of the charge motion towards the output.

detector:chips:chip_<i>:output_<j>.downToUp (dbLOGICAL) Vertical readout direction, defining the direction of the charge motion towards the output.

detector:shutter.description (dbBYTES32) Shutter description.

detector:shutter.available (dbLOGICAL) Shutter availability.

windows:window_<i>.expType (dbINT32). Exposure type.

windows:window_<i>.uit<j> (dbDOUBLE) Exposure time.

windows:window_<i>.xFirst (dbINT32) Horizontal coordinate of first pixel.

windows:window_<i>.yFirst (dbINT32) Vertical coordinate of first pixel.

windows:window_<i>.xDim (dbINT32) Horizontal dimension.

windows:window_<i>.yDim (dbINT32) Vertical dimension.

windows:window_<i>.xBinning (dbINT32) Horizontal binning factor for readout.

windows:window_<i>.yBinning (dbINT32) Vertical binning factor for readout.

windows:window_<i>.expRepeat (dbINT32) Number of repetitions.

windows:window_<i>.expModeIndex (dbINT32) Index of the mode chosen for exposure.

windows:window_<i>.expFileName (dbBYTES32) Name of FITS file where image data are written.

windows:window_<i>:exposure.expId (dbINT32) ID of exposure.

windows:window_<i>:exposure.expStatus (dbINT32) Status of exposure.

windows:window_<i>:exposure.shutStatus (dbINT32) Shutter status.

windows:window_<i>:exposure.timeRem (dbDOUBLE). Remaining time.

windows:window_<i>:exposure.readTime (dbDOUBLE). Readout time.

windows:window_<i>:exposure.transPercent (dbINT32) Percentage of image transferred to WS.

windows:window_<i>:exposure.transTime (dbDOUBLE). Time to transfer image to WS.

expModes.expMode_<i>.description (dbBYTES32) Description of mode.

expModes.expMode_<i>.wipeSeq (dbBYTES32) Sequence for wiping.

expModes.expMode_<i>.wipeVolt (dbBYTES32) Voltage table for wiping.

expModes.expMode_<i>.preIntSeq (dbBYTES32) Sequence for pre integration.

expModes.expMode_<i>.preIntVolt (dbBYTES32) Voltage table for pre integration.

expModes.expMode_<i>.intSeq (dbBYTES32) Sequence for integration.

expModes.expMode_<i>.intVolt (dbBYTES32) Voltage table for integration.

expModes.expMode_<i>.readSeq (dbBYTES32) Sequence for readout.

expModes.expMode_<i>.readVolt (dbBYTES32) Voltage table for readout.

expModes.expMode_<i>.outputs (Vector of dbINT32) List of the Ids of the outputs used for readout, in the appropriate order.

expModes.expMode_<i>.elAdu (Vector of dbDOUBLE) Conversion factor (Electrons per Adu) per output.

expModes.expMode_<i>.ron (Vector of dbDOUBLE) readout noise per output.

telemetry.enabled (dbLOGICAL) Telemetry enabled or not.

telemetry.opState (dbINT32) Current state of telemetry monitoring.

telemetry.current (vector of dbDOUBLE) Current telemetry values.

periodicWipe.enabled (dbLOGICAL) Periodic wipe enabled or not.

periodicWipe.opState (dbINT32) Current state of periodic wipe.

periodicWipe.period (dbINT32) Wipe period.

6.2 Interface between NGCOSW and TCS

Interactions with TCS are performed through the following online database attributes:

wcs.ra (dbDOUBLE) Centre right ascension in degrees for World Coordinates display

wcs.dec (dbDOUBLE) Centre declination in degrees for World Coordinates display

6.3 Image processing interface

Image processing results are stored in the following online database attributes:

windows:window_<i>:ip.min (dbINT32) min pixel value.

windows:window_<i>:ip.max (dbINT32) max pixel value.

windows:window_<i>:ip.rms (dbDOUBLE) Rms calculation.

windows:window_<i>:ip.xCen (dbDOUBLE) Error vector (x component).

windows:window_<i>:ip.yCen (dbDOUBLE) Error vector (y component).

windows:window_<i>:ip.valCen (dbDOUBLE) Centroid value.

windows:window_<i>:ip.xFwhm (dbDOUBLE) Full-width half maximum (x component).

windows:window_<i>:ip.yFwhm (dbDOUBLE) Full-width half maximum (y component).

windows:window_<i>:ip.NumPix (dbDOUBLE) Number of pixels above threshold level.

windows:window_<i>:ip.BackGnd (dbDOUBLE) Background value.

6.4 Image data

If data storage is enabled, image data are stored stored - as FITS files compliant with [AD37] - in the $INS_ROOT/$INS_USER/DETDATA directory.

For rapid display, image data are delivered to the ESO RTD following [RD40].

7 Process Sequence Diagrams

The following paragraphs illustrate NGCOSW most common scenarios.

The actors defined are:

- **Initiator**: can be the Instrument Software, an Operator or an Engineer
- **Operating System**: the operating system of the computer where the Command Interface runs
- **NGC Base Software**: the NGCSW layer responsible for interactions with the NGC detector electronics
- **Shutter**: any shutter controller device (pulpo [RD18], PULPO2 [RD19], LCU, etc.)
- **Telemetry Controller**: any device for telemetry control (pulpo [RD18], PULPO2 [RD19], etc.)

7.1 Startup

The NGCOSW startup procedure is controlled by the ngcDcsStart utility. When starting up the system, the following activities take place:

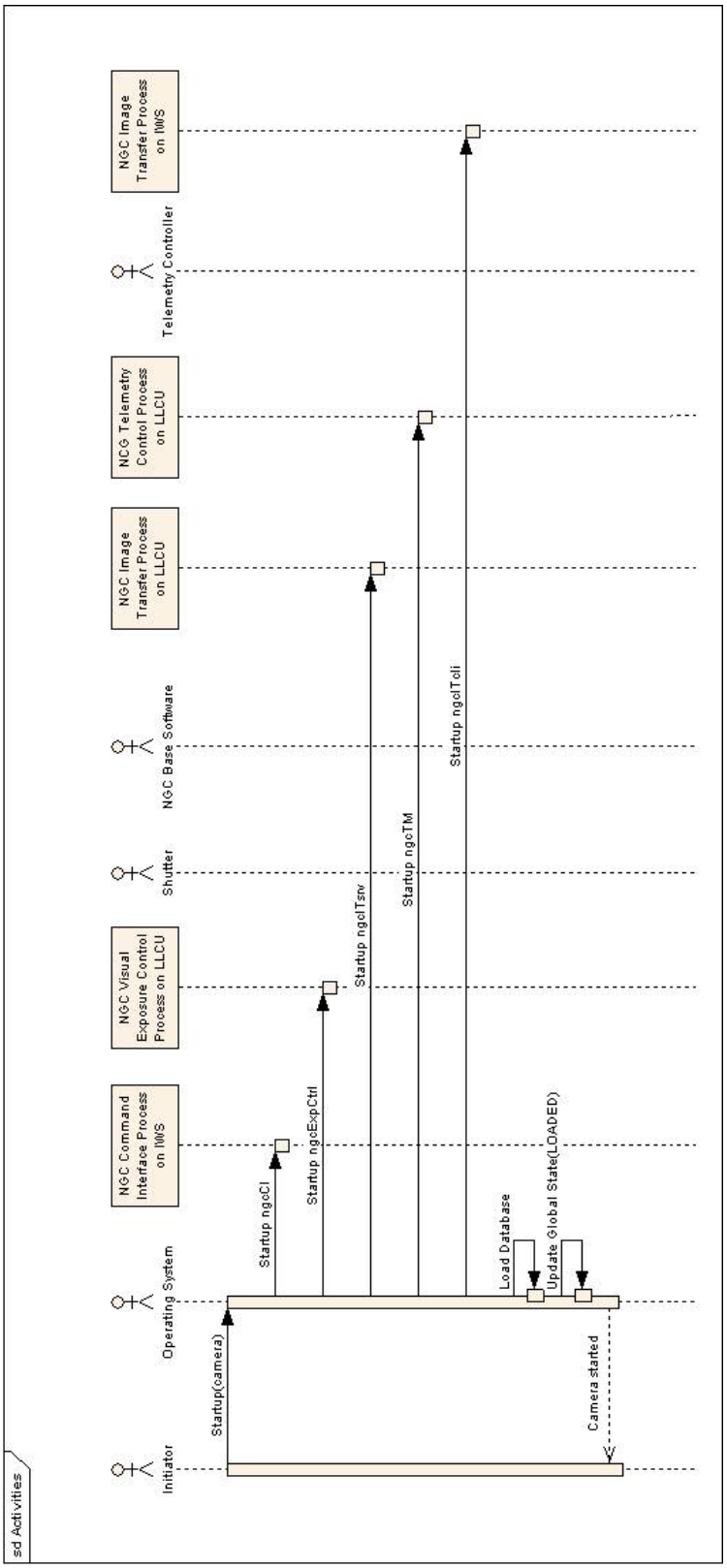

Figure 14 - Startup for NGC processes.

7.2 Standby, Online and Shutdown

The STANDBY command is used to bring the NGCOSW to the STANDBY state. When moving to STANDBY the following activities are performed:

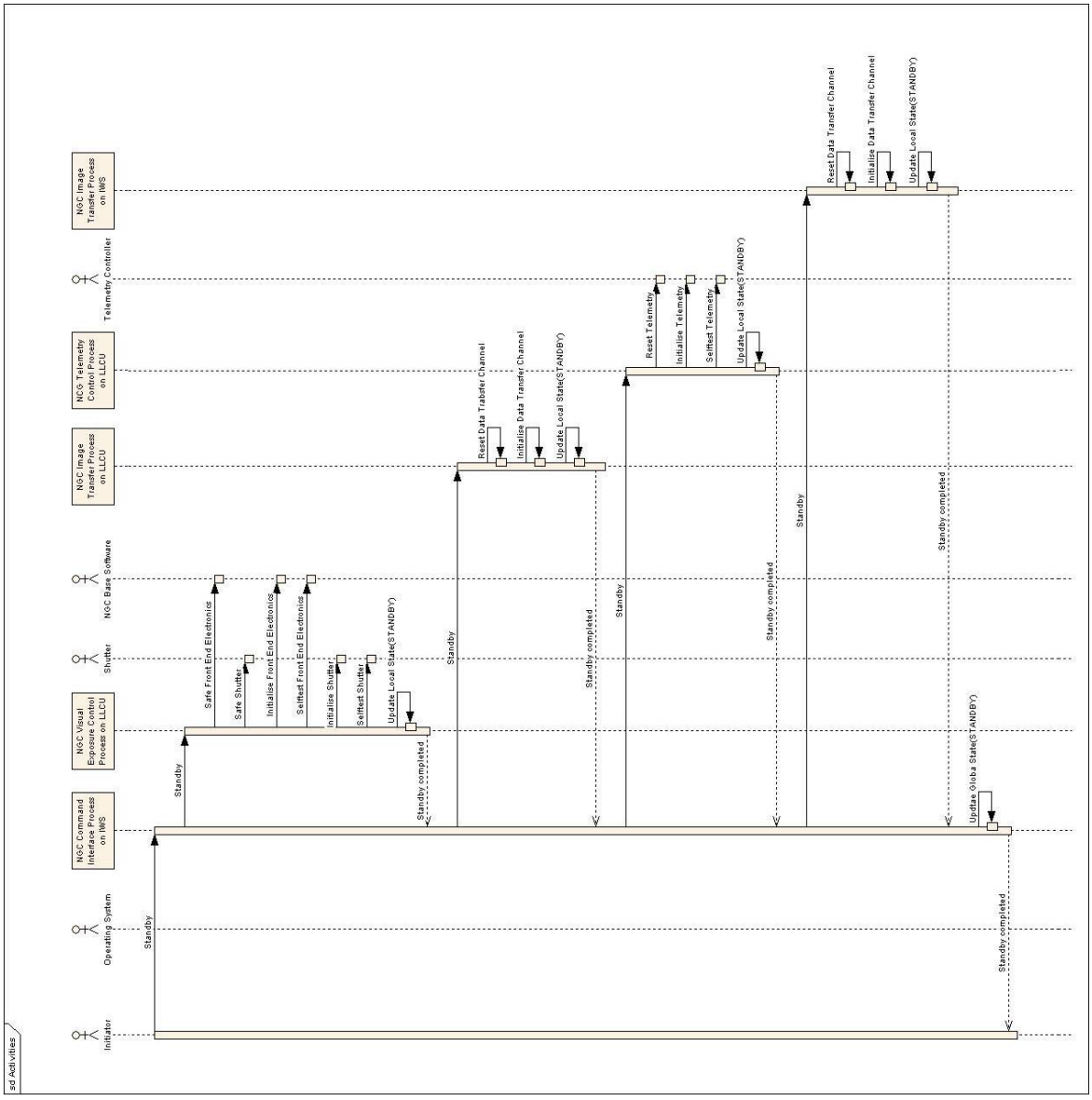

Figure 15 - Bring NGC processes to STANDBY.

The ONLINE command is used to move all the processes to the ONLINE state. When moving to ONLINE the following activities are performed:

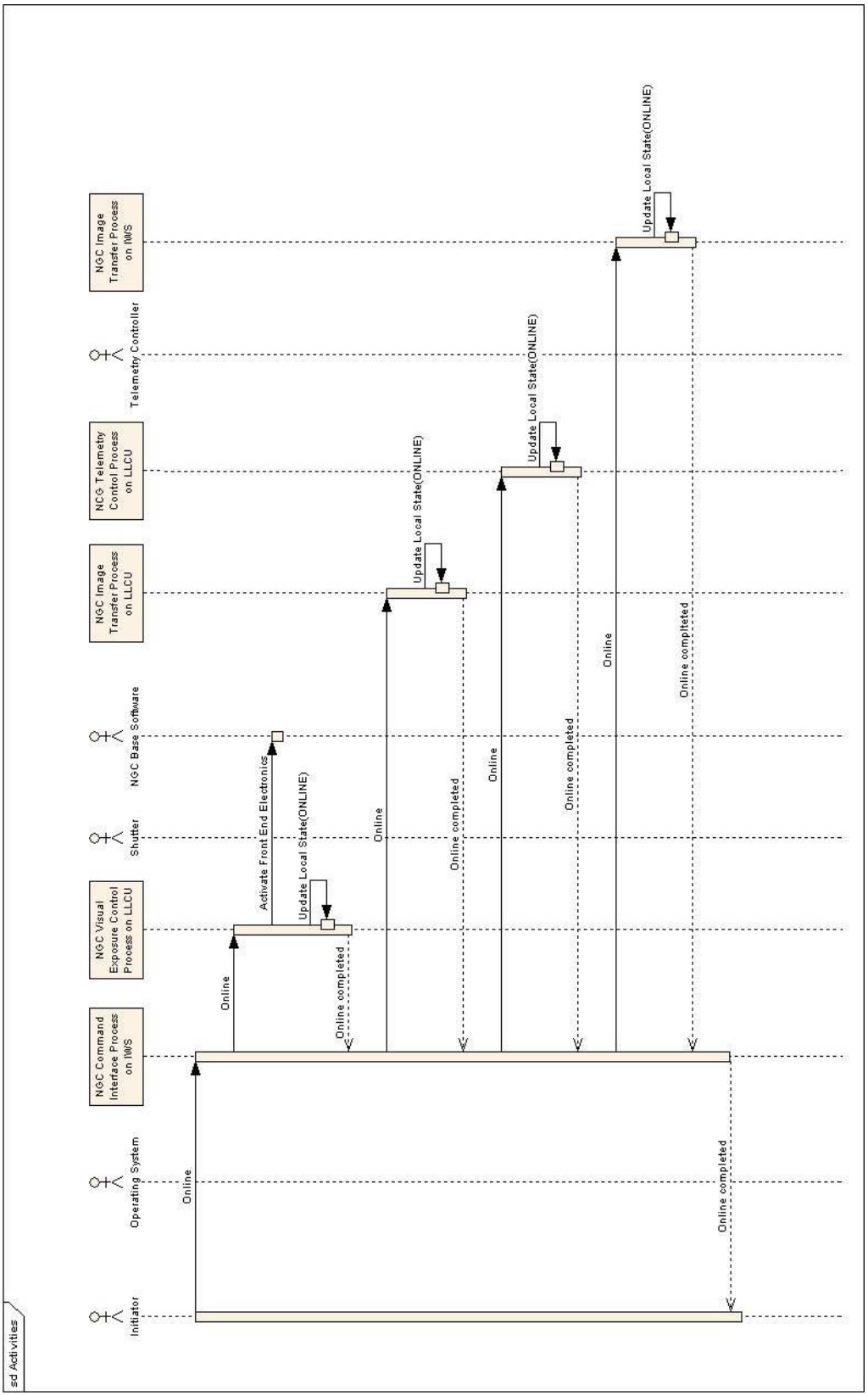

Figure 16 - Bring NGC processes to ONLINE.

The EXIT command is used to shut down NGCOSW and move all the processes to the OFF state. When moving to OFF the following activities are performed:

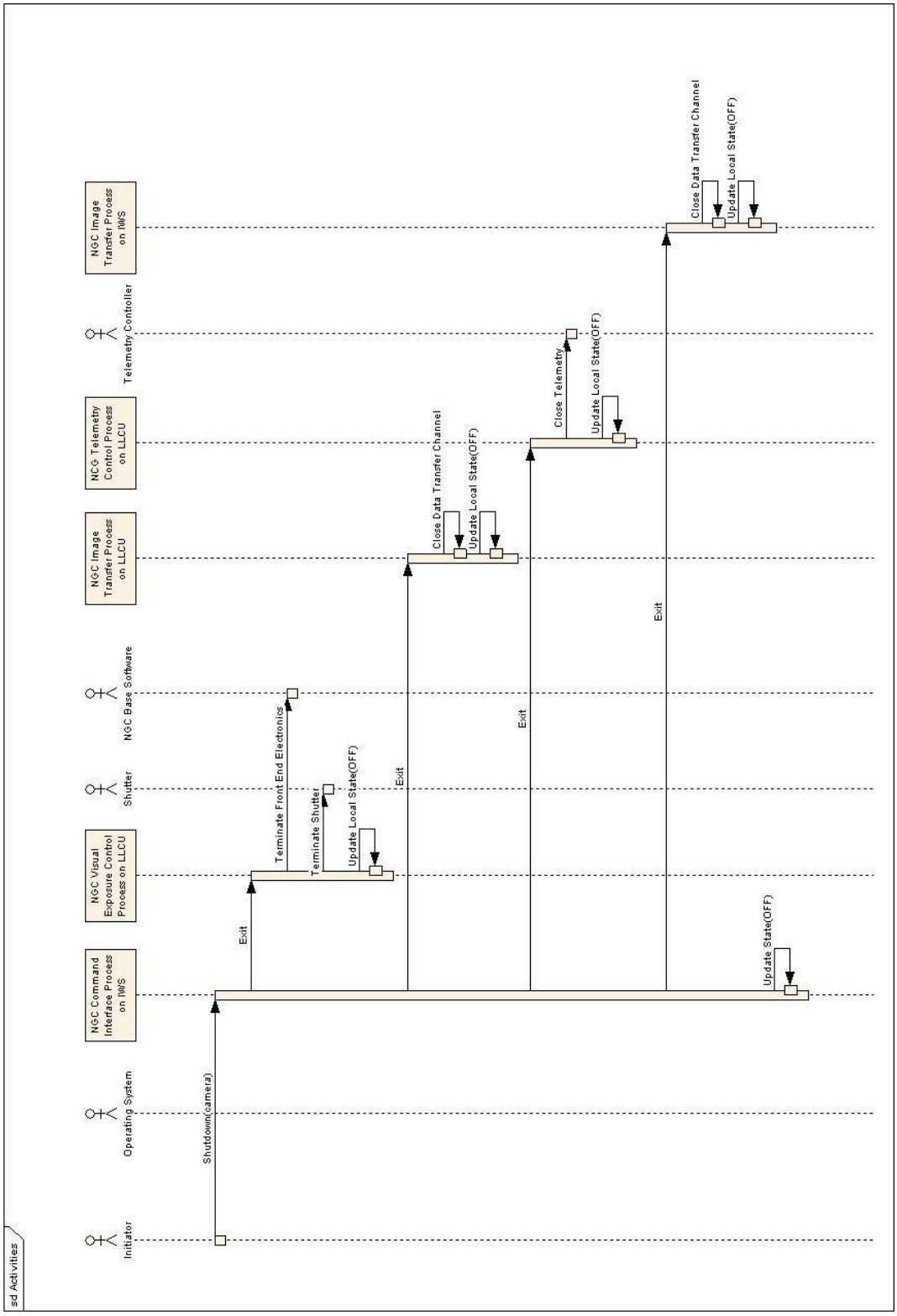

Figure 17 - Shutdown for NGC processes.

7.3 Exposure

During an exposure the following activities take place:

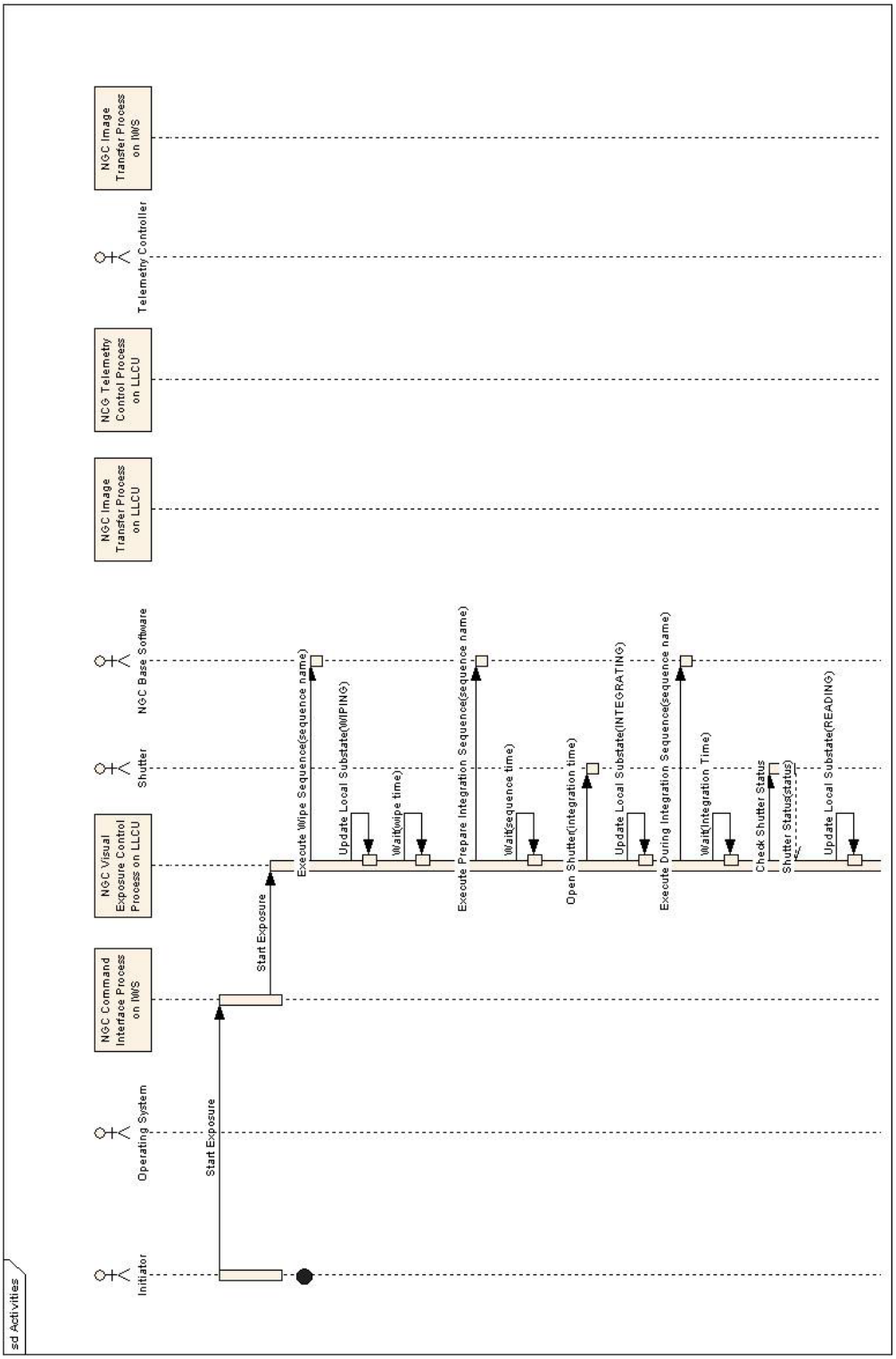

Figure 18 - Visual exposure with NGC (first part)

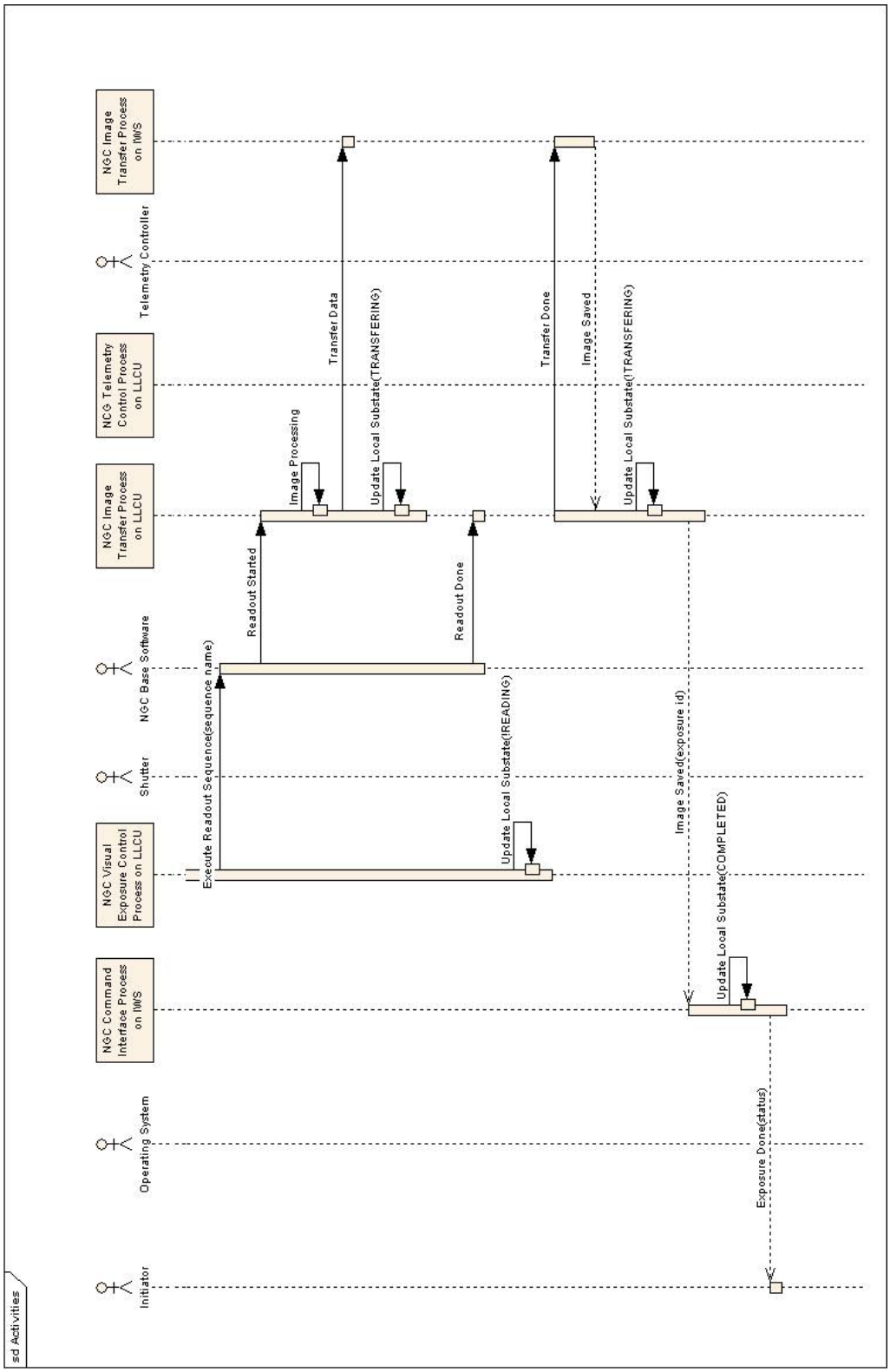

Figure 19 - Visual exposure with NGC (second part)

8 Performance Analysys

It is still TBD how to verify that the NGCOSW fulfills the quantitative performance requirements defined in [AD7].

9 Traceability Matrix

In the following paragraphs, all the requirements from [AD6], [AD7] and [AD20] which are relevant to NGCOSW are listed, together with the paragraph of this document where they are considered.

Requirements which should be handled at the level of a detailed design are marked as IDD (Item for Detailed Design).

9.1 NGC requirements

In this paragraph, all the requirements from [AD6] which are relevant to NGCOSW are listed. Since no numbering or labeling was defined in [AD6], requirement items have been labeled by incremental numbers.

9.1.1 Software requirements

Item	Requirement	Paragraph
	3.7 Software	/
1	Generally, software shall not limit the performance of the hardware.	2.4
2	It shall be command driven.	2.4 - 3.2.2 3.3.2 - 4.2.2 4.3.2 - 4.4.1
	3.7.1 High-level operating systems	
3	The high-level operating systems must be compliant with the VLT requirements. However, their number and diversity shall be kept to the minimum necessary for NGC.	2.3 - 2.4
4	A careful attempt shall be made to define an interface layer between the NGC control software proper and the operating system(s) and so to enable porting of all software above this layer at reasonable cost.	3.2.1
	3.7.2 Configuration control	/
5	At all times, all software and all parameter files shall be kept under configuration control.	2.4 - 3.1 - 4.1
6	For critical parameter files, an additional mechanism to ensure their integrity (e.g., check sums) should be considered.	TBD
	3.7.3 Programming	/
7	The usage of modern code-generating tools with a view towards testing, documenting, and debugging is encouraged. Their selection should be coordinated with the Technical Division. Island solutions should be avoided.	2.4 - 3.1 - 4.1

8	For each module, code and documentation shall be designed such that it can be maintained without analyzing other modules.	2.4
	3.7.4 Installation and start-up procedures	/
9	Fully automatic installation procedures and versatile configuration tools shall be provided.	2.4 (fcdarch) 3.2.1 (config. panels)
10	Execution of the standard control software in the telescope environment shall not require any special user privileges to be granted by the operating system.	2.4
11	The start-up script shall not require more than 10 s for auto-recognition of the hardware and the ready-for-use initialization of hard- and software.	3.2.1
	3.7.5 Resource checking	/
12	Software shall be able, prior to each exposure, to check the availability of all critical resources.	4.2.1
	3.7.6 Elementary functions	/
13	The set of elementary functions shall comprise those of IRACE and FIERA.	2.4
14	The addition of further functions shall be possible without affecting the others.	2.4
	3.7.7 Tests	/
15	Test software shall be developed in parallel to the control software itself.	3.2.1 - 3.3.3 4.2.3 - 4.3.4 4.4.3
16	The emulation of failures of other utilities (software, hardware, network, lack of resources, access denial) should be considered.	3.2.1 - 3.3.3 4.2.3 - 4.3.4 4.4.3
17	Standardized tests of the software corresponding to any supported hardware configuration shall be possible by merely selecting a single set of parameters.	3.2.1 - 3.3.3 4.2.3 - 4.3.4 4.4.3
18	A sequence of tests of several hardware configurations shall be possible without operator intervention.	3.2.1 - 3.3.3 4.2.3 - 4.3.4 4.4.3
19	Means should be considered to let NGC keep track of the frequency of usage of its key functionalities as a way to set usage-oriented test priorities.	TBD
	3.7.8 Times and timings	/
20	Without the VLT TIM, all absolute times shall be correct to within less than 0.1s.	4.2.1

21	Relative synchronizations and time intervals shall be accurate to better than 0.1% or, for intervals less than 10s, to better than 0.01s.	4.2.1
22	Stricter timing requirements shall be realized using TIM.	4.2.1
	3.7.9 Special modes	/
23	Support of the following techniques (in the order of decreasing priority) should be foreseen: o nod and shuffle o subpixel sampling and digital filtering so that during an exposure the built-up of the S/N can be followed by performing a regression analysis for each pixel o drift scanning o non-destructive readout o on-chip charge shifts by a user-definable amount (e.g., for through-focus sequences)	4.2.1
24	Device type-specific modes offered by state-of-the-art IR detectors shall be included.	Not related to NGCOSW
25	If centroiding functions need to be supported, this shall be possible at a frame rate of 1 Hz for data arrays of up to 256 x 256 pixels using a single Gaussian fit or similar. For much smaller data arrays, rates of up to 100 Hz should be possible.	4.3.1
	3.7.10 Consecutive exposures	/
26	If, e.g. due to on-line data processing, the time between end of detector readout and availability of the FITS file on disk becomes a significant overhead, it shall be possible to configure the software such that the next exposure begins right after the previous readout.	2.7.3 - 4.2.1
	3.7.11 Windowing and on-chip binning	/
27	Standard windowing and on-chip binning shall be provided	4.2.1
28	The number of windows should only be limited by the capabilities of the detectors.	4.2.1
	3.7.12 Pixel processor	/
29	A pixel processor shall be embedded in the system. Its interfaces to the remainder of the system shall be designed such that a replacement of the hardware plus operating system and/or of the processing software can be fully transparent to all other subsystems.	For NGCOSW, the (minimal) image processing is performed in the NGCLCU. If needed, a more powerful computer is used.

30	The following operations shall be supported from the beginning: ○ averaging of frames with and without removal of outliers (e.g., particle events) ○ bias subtraction ○ centroiding of point sources ○ TBC (If performance reasons so require, the implementation may be detector dependent.)	3.3.1 - 4.3.1
31	Close integration with NGC of a general-purpose image processing system featuring a user friendly scripting language could be considered.	2.4
32	More desirable is an interface to the ESO DFS and the inclusion of general-purpose algorithms and recipes in the DFS CPL for re-use by data reduction pipelines.	2.4
	3.7.13 ALMA control software	/
33	If this is in the general interest of ESO and supported by the Technology Division, elements of the ALMA control software may be used.	TBD
	3.7.14 Special utilities	/
34	For multi-port systems, bias equalization to within better than 1% shall be possible on demand but without any further operator supervision.	3.2.1 - 4.3.1

9.1.2 External interfaces

Item	Requirement	Paragraph
	3.8 External interfaces	/
35	Ideally, external interfaces (e.g., commands, databases) presently maintained by IRACE and FIERA would be supported by NGC with a minimum of changes so as to make the integration of NGC with the ESO operations scheme as seamless as possible. However, since in this regard the commonalities of FIERA and IRACE are very limited, this also limits backward compatibility. In no case shall NGC feature two different types of interfaces for the same purpose.	2.4 3.2.2 - 3.3.2 4.2.2 - 4.3.2 4.4.2
	3.8.1 Data format	/
36	The data format shall be compliant with the Data Interface Control Document	3.3.3
37	Comprehensive detector and electronics telemetry shall be included in the data headers.	4.4.1

38	From the FITS headers, it shall be possible to uniquely infer the complete set of hard- and software configuration and all parameter values.	3.3.1
	3.8.2 Output to real-time computers	/
39	A generalized, moderately configurable interface to real-time computers, e.g. for adaptive optics or fringe tracking applications, shall be defined (in cooperation with ESO software engineers working downstream from such an interface).	See 4.2.1, but this is more a HW requirement
40	It could be advantageous that (a possibly special incarnation of) the pixel processor can serve as the real-time computer (or vice versa).	HW requirement
41	Latency shall not exceed 100 µs.	HW requirement
	3.8.3 Real-time display	/
42	An interface to the RTD shall be provided.	3.3.1
43	For high frame rates, it shall be possible to request only every nth frame to be displayed.	4.3.1
44	Adaptive auto-selection shall be supported.	Part of RTD
	3.8.4 VLT telescope control system	/
45	It shall be possible to synchronize detector operations with the following functions: o Telescope nodding o M2 chopping o Non-sidereal tracking	4.2.1
	3.8.5 VLT time distribution system	/
46	The possibility of an interface to the VLT Time Interface Module shall be foreseen.	See Req. 22

9.1.3 Ancillary utilities

The development of the utilities does not directly involve the development of the NGCOSW.

9.1.4 Control of auxiliary functions

Item	Requirement	Paragraph
	3.10 Control of auxiliary functions	/
47	The scope of NGC shall be restricted to the control of detectors. However, the control of auxiliary functions and devices shall be integrated into the overall concept. A study shall be made whether, and for what reasons, some derivative of the present PULPO II device, a set of commercially available controllers, or some new customized development is most promising. One unit shall support o 32 temperature sensors o 8 temperature control loops o 8 shutters; it shall be possible to use all 8 either simultaneously or in two fully independent groups o 8 generic functions (e.g., control of an LED) that can be synchronized with the shutter operation	4.4.1
48	An NGC-controlled detector system shall be able to employ at least 4 of such units.	4.4.1
49	A generalized interface to such peripheral control functions, especially of shutters and for temperature stabilization, shall be considered.	4.2.1 - 4.4.1
50	Shutters with more than one moving part shall be covered. For bidirectional shutters, the direction of motion shall be controllable. The case that the start of an exposure is not defined by the shutter motion but by the end of the wiping of the detector shall be included.	4.2.1
51	Development of a generalized controller of standard ESO cooling systems, incl. compressor- or pulse tube-based systems, should be considered. Full support of the safety devices supplied with these cooling systems is vital. For alarm conditions with potentially serious consequences, an interface to the Paranal CAS is required.	4.2.1
52	Logging of all functions with a resolution of 5 minutes shall be possible. Where practical, the VLT logging utilities shall be used. Any stand-alone log files shall have a capacity of at least 10 days and be cyclically overwritten.	2.10 - 4.4.1

9.1.5 Diagnostic tools

Item	Requirement	Paragraph
	3.11 Diagnostic tools	/
	3.11.1 Hardware self-test	/
53	The hardware shall be able to execute a comprehensive self-test. It shall be possible to start it by pressing a physical button as well as by software. Due consideration shall be given to the protection of the detectors. The execution shall not exceed 5 minutes. An automatic log shall be produced.	HW requirement
54	In order to save space on the electronics boards, it is acceptable to let remote software (e.g., on the xLCU) execute these tests with hardware only reporting its status. This software shall be developed in parallel to the one of the hardware.	4.2.1
	3.11.2 Read-back of parameter values	/
55	It shall be possible to read back the actual values of all parameters set by software.	HW requirement
	3.11.3 Automatic identification of hardware components	/
56	All LRUs (line Replaceable Unit) shall have a unique identification that is readable by software.	HW requirement
57	An extension also to detectors shall be considered.	HW requirement
58	Software shall be able to use this information for auto-configuration.	4.2.1
	3.11.4 Error handling	/
59	Meaningful error messages and log files are essential; they shall enable software staff not familiar with the software or its scope to identify and fix minor problems. Different severity levels shall be distinguished. The status and options for the next actions shall be clear at all times.	2.10
60	It shall be possible to set the severity level up to which automatic recoveries from errors shall be attempted.	2.10
61	After a failed data saving, the OS shall have the possibility to recover the last frame.	3.3.1
62	After an interruption in the power supply, software should be able to automatically restore the status at the beginning of the last successful exposure.	Not possible

	3.12 Support of engineering work	/
	3.12.1 Engineering mode	/
63	There shall be a password-protectable engineering mode. It may contain extra modules and options while otherwise may be omitted for reasons of convenience. However, modules used for normal operations shall be identical.	3.2.1
64	This mode shall offer access to all essential elementary detector control functions and allow hardware engineers rapid proto-typing of experimental software.	3.2.1
	3.12.2 Change of parameters	/
65	A change of software-configurable parameters shall not require a re-start of the system and, where possible, be supported also during readout. This would also benefit multi-mode instruments where, e.g., imaging and spectroscopy require different parameter sets for optimal performance. Switching between modes shall not lead to any hysteresis.	4.2.1 - 4.4.1
66	A mechanism shall be implemented to reduce the risk of out-of-range parameter values being set accidentally that could damage the connected detector(s). One possibility might be to let the controller hardware request a unique electronic ID (such as the serial number) from the detector	4.2.1 - 4.4.1
67	An interface to BOB shall be provided that permits parameter values to be set from dedicated observation blocks / observing templates. To take advantage of this, laboratory setups would need to be able to emulate VLT-compatible instruments to the extent that VLT control software Sequencer scripts can be executed.	2.4
	3.12.3 Detector library	/
68	A repository with parameter files for specific detector types and their baseline operating modes shall be offered. For engineering purposes, easy copying and editing of such files shall be supported. To the extent possible, different installations of comparable detector systems shall share these data.	2.11
	3.12.4 Disabling of components	/
69	It shall be possible to declare LRUs and channels defunct. In response to this, the software should be able to automatically adapt itself to the remaining hardware configuration.	4.2.1

	3.12.5 Special modes	/
70	The following shall be foreseen: ○ pocket pumping ○ convenient connection of monitoring equipment such as oscilloscopes, multimeters, and logic analysers ○ determination of PTF of DC-coupled IR and CMOS devices by a capacitive comparison technique	For the first two items, these are just performed by special sequences. The third item is a requirement on hardware
	3.12.6 Programming interface	/
71	Thought shall be given to the provision of an efficient programmer's interface, ideally with a standard scripting language such as Tcl/Tk, that permits engineers rapid proto-typing of detector control and data processing software.	Req. for the NGC Base Software [AD9]
	3.12.7 Test facility	/
72	A cost-effective test facility for all types of LRUs shall be supplied. It may either be integrated into the controller or stand-alone.	Outside the scope of this document, since related not to NGCOSW, but to sw which uses NGCOSW
73	Its software shall use the one of the NGC only.	same as above
74	An expandable collection of standard test functions shall be considered.	same as above
75	Where applicable, test results should also be offered in graphical form with an option for hardcopies.	same as above
	3.12.8 Simulation modes	/
76	Standard VLT simulation modes shall be supported. Simulation should be one of the standard modes of NGC rather than an add-on.	2.7 - 2.9
77	To the greatest possible extent, simulation of key elements shall be supported in both soft- and hardware.	2.7 - 2.9
78	Hardware simulators to generate programmable test pixel patterns and video waveforms shall be devised.	HW requirement
79	Software simulators shall be hierarchically structured and permit the simulation of data streams with real numbers and realistic data rates so that relative timings, etc. can be tested.	2.7 - 2.9

9.2 NGCOSW requirements

In this paragraph, all the requirements from [AD7] which are relevant to NGCOSW are listed. Since no numbering or labeling was defined in [AD7], requirement items have been labeled by incremental numbers.

9.2.1 Common functional requirements

From [AD7], par 3.1.1:

Item	Requirement	Paragraph
80	NGCSW shall handle at least TBD clocks, TBD biases, TBD preamps and TBD video channels.	TBD
81	Pixels read out from detector(s) shall be reordered before transferring to the IWS. Delivered images will not need further reordering.	4.3.1
82	NGCSW will implement, as a minimum, the commands already used by FIERA and IRACE and described in their CDTs with an interface which will allow backward compatibility.	3.2.2 - 3.3.2 4.2.2 - 4.3.2 4.4.2
83	The ONLINE status requires that all voltages are loaded and switches closed as well as telemetry is acquired and checked.	2.6
84	NGCSW will handle multiple independent detectors.	2.4
85	It shall be possible to read number of windows limited only by detector properties.	See Req. 28
86	Windows shall be read either in hardware through sequences or in software. The latter case implies that a full frame is read out and then a window of data is computed in memory.	4.3.1
87	Binning shall be implemented to an arbitrary value and with independent values in x and y.	4.2.1
88	Telemetry shall be available at all times, with the possibility to have a separate period for logging on the VLT logMonitor.	4.4.1
89	NGCSW shall transfer all computed results to display (RTD) and/or to FITS-file: **DIT** (result frame after one integration) **INT** (average of NDIT integrations) **STDEV** (standard deviation of NDIT integrations) **HCYCLE** (intermediate results after half chopping period	IR only
90	Display visualization is done in parallel to all other data transfers (i.e. one may look at the DIT frame while storing the INT frame to disk and while the next data is already being processed).	3.3.1

91	If processing- or data-transfer-bandwidth exceeds the capacity of one single computer, the task is split up to N computing units.	4.3.1

9.2.2 Visual specific functional requirements

From [AD7], par 3.1.2:

Item	Requirement	Paragraph
92	NGCSW will handle exposures with and without shutter operation (e.g. Normal or Dark) and will also implement the possibility of driving an external light source (LED exposures on HARPS).	4.2.1
93	It shall be possible to execute sequences during the integration.	4.2.1
94	Periodic wiping will be possible.	4.2.1

9.2.3 Infrared specific functional requirements

Requirements in [AD7], par 3.1.3 are not applicable to the visual case.

9.2.4 User interface requirements

From [AD7], par 3.2.1:

Item	Requirement	Paragraph
95	NGCSW user interfaces will be developed following rules described in [AD38] and [RD39].	3.2.1
96	The user interface for telescope operations will merge functionalities of current FIERA and IRACE user interfaces.	2.5 (Dictionary) 3.2.2 - 3.3.2 4.2.2 - 4.3.2 4.4.2 (CDTs)
97	There will be an engineering user interface which will allow access to all functionalities of NGC. This user interface will be meant for HW engineers use only.	3.2.1

9.2.5 Hardware interface requirements

From [AD7], par 3.2.2:

Item	Requirement	Paragraph
99	Interface between AO/RTC and NGCSW is defined in [AD20].	See 9.3
99	Interface between NGCSW and VLTI applications are defined in [AD?].	TBD

9.2.6 "Sequencer programming" - software interface requirements

These requirements, in [AD7], par 3.2.3, are related to the NGC Base Software [AD9]

9.2.7 "RTD" - software interface requirements

From [AD7], par 3.2.3:

Item	Requirement	Paragraph
100	Interface to standard RTD shall be provided	See Req. 42
101	The Interface and the library to be used are defined in [RD40]	3.3.3

9.2.8 "Pixel processor" - software interface requirements

These requirements in [AD7], par 3.2.3 are not applicable to the visual case.

9.2.9 "TCS" - software interface requirements

From [AD7], par 3.2.3:

Item	Requirement	Paragraph
102	NGCSW shall deliver centroiding data to TCS. The interface, i.e. the parameters needed by TCS, are described in [RD43]	4.3.3
103	If needed (i.e. infrared exposure) flat-field frame and bad pixels mask shall be downloaded to NGCSW which will then distribute them to the relevant subsystem.	in the optical case, this is part of the image processing performed by the Imagre Transfer Server, see 4.3.1

9.2.10 Communication interface requirements

From [AD7], par 3.2.4:

Item	Requirement	Paragraph
104	Communication between internal subsystems of NGCSW shall be implemented using VLTSW standard messaging tools as well as CCS database.	2.5.3
105	Whenever not otherwise specified, the same standards will be used for all other communication interfaces.	2.5.3
106	Data transfer to the IWS will use standard VLT DXF software, unless NGCSW runs on the same IWS.	3.3.3 - 4.3.3

9.2.11 "Timing" performance requirements

From [AD7], par 3.3.1:

Item	Requirement	Paragraph
107	General timing requirements on SW are given in [AD6].	See Req. 20 - 22
108	Synchronization requirements are given in [AD6].	See Req. 21

9.2.12 "Data transfer" performance requirements

From [AD7], par 3.3.2:

Item	Requirement	Paragraph
109	Data transfer rate to the IWS shall be limited only by the readout speed. An overhead of max. 5 seconds will be considered acceptable.	4.3.1 - 8.1

9.2.13 "Post processing" performance requirements

From [AD7], par 3.3.3:

Item	Requirement	Paragraph
110	Centroiding performance requirements are specified in [AD6].	See Req. 25

9.2.14 "Standard compliance" design constraints

From [AD7], par 3.4.1:

Item	Requirement	Paragraph
111	ESO software standards.	2.3 - 2.4

9.3 AO requirements

In this paragraph, all the requirements from [AD20] which are relevant to NGCOSW are listed. The item labeling used in [AD20] has been kept.

9.3.1 Functional requirements

From [AD20], par 4.1:

Item	Requirement	Paragraph
AONGCREQ-004	Number of detectors controlled from single NGC: 1-4	2.3 - 2.4
AONGCREQ-005	It should be possible to control multiple detectors with a single NGC	2.4 - 4.2.1
AONGCREQ-006	The detectors controlled by a single NGC all be read with the same read-mode and frame rate. Operations such as start and stop, should operate on all detectors simultaneously	4.2.1
AONGCREQ-007	It should be possible to synchronize the start of a read sequence between NGCs	Implicitely in AONGCREQ-014
AONGCREQ-008	the synchronization should allow the start of the readout in separate NGCs to be started within the frame jitter time	Limited in AONGCREQ-014
AONGCREQ-009	Ability to visualise 1 of N frames asynchronously on instrument WS using standard RTD, the value of N will be chosen to allow visualisation on the WS of 1-4Hz, the maximum frame rate required to the WS will be 50Hz	See Req. 43
AONGCREQ-010	Ability store 1 of N frames asynchronously in FITS format on instrument WS, the value of N will be defined to allow a maximum frame rate of 50Hz	3.3.1

AONGCREQ-011	Ability to store a series of N guaranteed consecutive frames, where N should be sufficient to store a minimum of 2 seconds of data with a goal of 10s.	3.3.1 is this related to data cubes?
AONGCREQ-012	At least one ROI (windowing/regions of interest) per detector should be supported, as a goal multiple ROIs.	See Req. 28
AONGCREQ-013	It should be possible to update the ROI definition (at a minimum start coordinates) dynamically when a loop of readouts is in operation	4.2.1
AONGCREQ-014	Ability to synchronize the start of a readout sequence with an external trigger to within 30 micro seconds	4.2.1
AONGCREQ-015	1x1 to 16x16 defined in steps of 1 pixel binning is the same for each window within a given detector	See Req. 87
AONGCREQ-024	It should be possible to command the NGC to execute a defined read sequence for the defined mosaic	4.2.1
AONGCREQ-025	N times	4.2.1
AONGCREQ-026	with a user definable (in microseconds) delay (from 0 to TBD) between executions	4.2.1
AONGCREQ-027	where N is a value between 1 and Inf	4.2.1
AONGCREQ-028	A stop command for a given mosaic should stop the current loop of readout sequences at the completion of the next cycle	4.2.1
AONGCREQ-028 (same number as above!)	It should be possible to instruct the NGC to pass a 15bit data ramp over the real time data link in a standard frame including both start and end of frame words at frame rates up to the maximum supported	2.11.1 - 4.2.1
AONGCREQ-029	As a goal it should be possible to load simulated images into the NGC memory and have them ‘played back’ over the RT data link as though coming from a normal readout sequence	Not possible, see 2.11.1 - 4.2.1
AONGCREQ-030	the replay speed should be a user parameter up to the maximum possible frame rate	4.2.1

9.3.2 Interface requirements

From [AD20], par 4.2.3:

AONGCREQ-038	Define one (or more) ROIs for a specified detector with start pixel and window dimensions	Same as AONGCREQ-012
AONGCREQ-039	Define readout mode for detector mosaic	Same as AONGCREQ-024
AONGCREQ-040	Enable/Disable storage of detector frames	4.2.1
AONGCREQ-041	Define sub-sampling of readout to be passed to instrument workstation for visualisation or storage	4.3.1
AONGCREQ-042	Start readout of mosaic of detectors for N cycles, 1<=N<=Inf	Same as AONGCREQ-024 AONGCREQ-025 AONGCREQ-026 AONGCREQ-027
AONGCREQ-043	with optional synchronization with external synch signal	Same as AONGCREQ-014
AONGCREQ-044	Stop readout of a mosaic of detectors	Same as AONGCREQ-028
AONGCREQ-045	Acquire and store a contiguous set of N frames from a ‘group’ of detectors	Same as AONGCREQ-011 ?
AONGCREQ-046	Upload a set of detector frames to be replayed in simulation mode	Same as AONGCREQ-028 and AONGCREQ-029
AONGCREQ-047	Replay previously uploaded simulated data frames at defined loop frequency	Same as AONGCREQ-030

__ oOo__