Array Control & Data Acquisition Technical Design Report

This Version:				
Ver.	Created	Comment	Distribution	Corresponding...
1.5	2015-05-04	CDR release version. Minor revision of design and prototyping section, revision of plans and costs section, revision of maintenance section.	Open	Editor: See appendix F Checker: ____________ Approver: ____________

Keywords:

Technical Design Report; TDR; Array Control; Data Acquisition; ACTL

Version History:				
Ver.	Date	Comment	Distribution	Corresponding...
1.0	2014-04-15	Initial version based on pTDR	ACTL	Editor: Matthias Fuessling
1.1	2014-05-09	Split of document into separate sections	ACTL	Editor: Matthias Fuessling
1.2	2014-10-31	First released version	Open	Editor: See appendix F Approver: Peter Wegner
1.3	2015-01-05	Various updates of individual sections, update of costs	Open	Editor: See appendix F Approver: Peter Wegner
1.4	2015-03-30	Revision of individual sections, based on corrections by CTA PM	Open	Editor: See appendix F Approver: Peter Wegner

Table of Contents

1 Summary and Introduction

1.1 Overall Concept

ACTL (Array Control and Data Acquisition) provides the hardware and software that is necessary to (i) monitor and control all telescopes and auxiliary devices in the CTA arrays, to (ii) schedule and perform observations and calibration runs, and to (iii) time-stamp, read-out, filter and store data. CTA operations will be carried out from on-site data centres (OSDC) at the CTA sites. Each OSDC comprises computing hardware for execution of ACTL software and for mass storage, and is connected to the local CTA array (for readout and control) and the outside world (for data export and control).

The ACTL software system depends on external inputs (e.g. for the long-term scheduling), accepts incoming alerts (e.g. directly from other observatories or in connection with ToOs), and manages the execution (but not the implementation) of a real time analysis (RTA, level-A analysis) software.

The concept for the ACTL hardware emphasises the usage of standard, off-the-shelf networking and computing elements and tries to minimize the amount of hardware (e.g. electronics cards, PCBs) that must be specifically developed for CTA. The concept for the ACTL software accounts for the need to develop, maintain, and operate a more complex and more stable (when compared to existing IACT arrays) software system with limited manpower at moderate cost. It prescribes the use of software frameworks, the application of widely accepted standards, tools and protocols, and follows basically an open-source approach.

1.2 Summary of Design

Besides the control of the numerous devices associated with telescopes and the common calibration facilities (cf. Table 1.1), the triggering, time-stamping and storing of the shower data is the most important task of the ACTL system. The overall data stream that the ACTL system has to cope with (cf. Sec. 2.2.1 and 2.2.2) is dominated by the telescope cameras whose local camera triggers select events and generate data rates that are assumed to vary between 10 MB/s and about 5 GB/s. The requirement of stereoscopy (i.e. ≥ 2 telescopes have triggered simultaneously) will result in array trigger rates of O(30 KHz) and a data rate of O(3 GB/s). These data must be cached in an on-site data repository for about two months to allow for (i) offline-like data suppression and event selection and for (ii) export of data to off-site data centres.

The ACTL hardware will comprise (see Fig. 1.1):

1. A single-mode fibre ethernet wide area network (several 10 Gbit/s, cf. Table 3.2 on page 43) connecting the telescopes with the OSDC and facilitating data transfer, array-level triggering, and clock-distribution. In addition, a local WLAN will be made available close to each telescope to facilitate access with laptops and tablets for debugging and commissioning.
2. Computers (so called camera servers) located in the OSDC and assigned to one telescope to receive the data after a camera trigger. The camera servers are provided by the telescope projects and buffer the Cherenkov data while the array trigger (see below) makes its decision.
3. A central computing cluster (also located in the OSDC) for execution of ACTL software, event building and filtering, and operation of the data repository. Estimates (cf. Table 1.2) for number of

computing cores and the capacity of the data repository are about 1550 (870) and 3 PB (1.5 PB) for CTA-South (CTA-North).

4. Hardware (few computers) for a software array trigger (SWAT), also located in the OSDC. The SWAT inspects the telescope event time-stamps and selects stereoscopic events.
5. A WhiteRabbit (WR) network connecting a central GPS clock with each telescope and the SWAT. The WhiteRabbit system provides time-stamps with sub-nanosecond precision that are used to associate data and telescope events.
6. WR interface cards for time-stamping and array-level triggering, respectively, that are deployed close to any hardware using their services, in particular the Cherenkov cameras.

The design intentionally does not prescribe hardware standards for transfer of data and control (such as CAN or VME bus). Dedicated protocols and Ethernet are used for the transfer of the bulk data (the Cherenkov camera data); for all other applications the usage of one particular software standard (OPC UA, see below) is enforced to ensure connectivity with the many different hardware devices (cf. Fig. 1.2, left).

The high-level ACTL software is developed on top of ACS (ALMA Common Software, see [1]), a software framework for the implementation of distributed data acquisition and control systems. ACS has been developed by ESO and has been successfully applied in projects of similar scale (e.g. ALMA [2]). ACS distributions (provided by ALMA computing) are executed on a variant of the Linux operating system (Scientific Linux) which will therefore be used on all full-scale ACTL computers. ACS is based on a container-component model and supports the programming languages C^{++}, Java and Python (in particular for scripting). The high-level ACTL software will be executed predominantly on the central computer cluster and will access most hardware devices via OPC UA (see [3]) using the mechanism detailed in [4]. OPC UA is an industry standard, so OPC UA servers can either be provided by device vendors or be developed (using two suggested software development kits [5]) by the device teams. In most cases, the functionality of the OPC UA servers (variables, methods etc.) define the interface between ACTL and a hardware device (e.g. a weather station or CCD camera) that must be controlled and read-out.

The ACTL software system will comprise the following major parts (see Fig. 1.2, right):

1. A scheduler for CTA (SCTA) optimizing the array usage for observations at any point in time given a list of selected targets, their CTA-assigned scientific priority, the available telescopes, and external conditions (e.g. weather).
2. A central control system (about O(1000) distributed processes), implementing the execution of observations using the scheduler under local (human operators with GUIs) or remote control.
3. An acquisition system for the telescope Cherenkov data, implementing the further filtering of data and its storage in the on-site data repository.
4. A monitoring, configuration and slow-control system, commanding each hardware device and examining and recording (in databases) its state and configuration regardless of whether observations are currently ongoing.

In general, the ACTL software system will be used by a trained operator crew to perform observations and calibration and maintenance procedures. The system will support the authentication of users and the authorization of access to the telescopes. The safety of the operator crew (e.g. in the event of rapid unexpected telescope movements in response to a GRB alarm) will be ensured by a system of hardware and software interlocks. Capabilities to allow for remote control and monitoring of the system (e.g. from a remote control room) will be pursued as a goal and considered in the design. Any remote operation will be accompanied by a well-defined set of rules and safety procedures (cf. Sec. 2.3.1).

Camera	Telescope	Array
Camera Lid	Drive Systems	Lidar(s)
Camera Readout	Safety and Monitoring System	Ceilometer(s)
High Voltage System	Camera Lock/Shelters	Weather Station(s)
	CCD camera(s)	Optical Telescope(s)
	Environment Monitoring Devices (Radiometer, ..)	Anemometer(s)
	Calibration Devices (LED pulsers, Lasers, ..)	Rain Sensors
	Active Mirror Control	UVScope(s)

Table 1.1 – Summary of (external to ACTL and not exclusive list of) hardware subsystems that need to be controlled and read out by ACTL, respectively. Subsystems associated with a Cherenkov camera, telescope and a CTA array are shown from left to right.

	CTA-South	CTA-North
disk space	3 PB	1.5 PB
computing cores	1550	870

Table 1.2 – Summary of the ACTL installations in the on-site data centres for CTA-South and CTA-North. The disk space is the total disk space for archives and data bases. The computing cores comprise all machines in the OSDC that do not serve the disk space but cover ACTL tasks (data acquisition, triggering, monitoring), IT service tasks and data analysis tasks (Level A/B analysis).

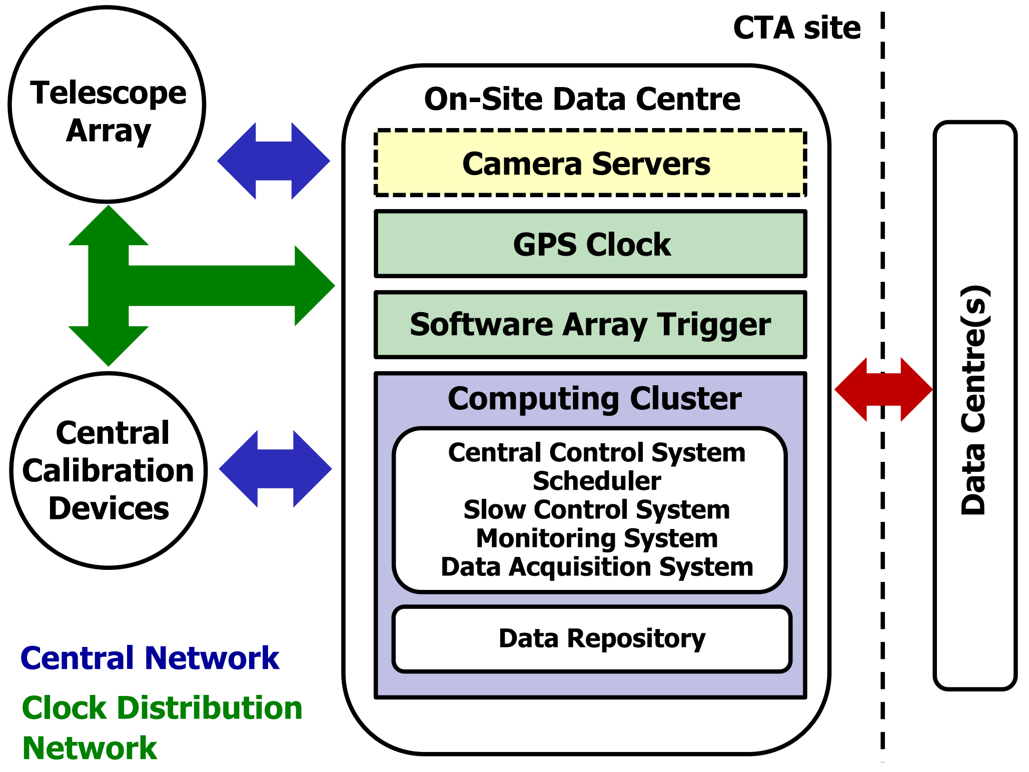


Figure 1.1 – Overview of the ACTL system on a CTA site. The camera servers (dashed box) are not provided by the ACTL WP but by the camera projects. See text for explanations.

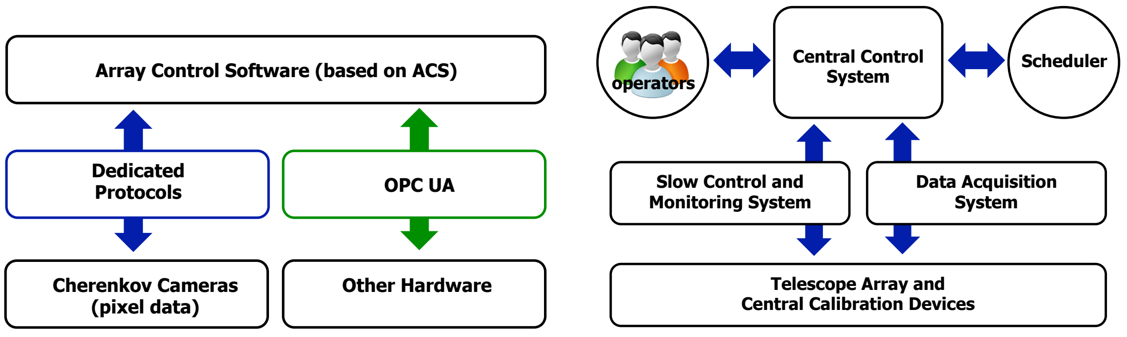

Array Control Software (based on ACS)
Dedicated Protocols
OPC UA
Cherenkov Cameras (pixel data)
Other Hardware
operators
Central Control System
Scheduler
Slow Control and Monitoring System
Data Acquisition System
Telescope Array and Central Calibration Devices

Figure 1.2 – *Left:* Schematic of the hardware access in the ACTL system. The ACS-based data acquisition and central array control software will communicate with most hardware devices via OPC UA (right branch). Due to the high bandwidth requirements, the readout of the camera data will employ dedicated protocols (left branch). *Right:* Overview of the basic building blocks of the data acquisition and central array control software.

1.3 Summary of Plans

In the pre-construction phase, software prototypes have been developed, e.g. for the data acquisition and array trigger systems as well as for the control and monitoring framework of devices in the context of prototype telescopes. Architectural design work was started with use-case discussions, but the development of a full architecture for the software system is one of the most important current tasks. The software architecture is the key ingredient for a timely implementation of the ACTL software and is given priority. Recently, the ACTL work-package has set up a test computing cluster (5 data base servers, 10 compute nodes, network) at DESY. This computing cluster will serve as the main platform for integration and verification of ACTL software products. It will host systems for building and automatically testing ACTL software and is also open to the OPC UA developers in the telescope and device teams. The development of OPC UA servers in Java and C^{++} will be supported with examples and advice on a best-effort basis (cf. [4]). Another important step is the release of virtual machines (VMs). These virtual machines will come with a *Scientific Linux* system with pre-installed ACTL software and tools (e.g. databases) and a development environment for ACS and OPC UA. The VMs can be used by the device teams for software development and for testing the interoperability of the newly developed software with ACTL software.

In the construction phase, ACTL will make software available to aid in the testing of single assemblies (e.g. a Cherenkov camera) before shipment to a CTA site. Most of the ACTL products (instrument slow control software, timing system) and their integration with products of other work-packages (telescopes, Cherenkov cameras) will be fully tested and validated at the prototype telescopes sites or in the laboratories, before actual deployment of the telescopes and ACTL systems at the respective CTA site. As soon as the basic infrastructure for the first telescopes has been built on a CTA site, ACTL will provide an initial on-site computing cluster. A first version of the ACTL software will be available for the commissioning of telescopes. The commissioning of telescopes and auxiliary devices (e.g. devices for atmosphere monitoring) will be supported with manpower and tooling (e.g. generic OPC UA clients). It is expected that the commissioning will be finished in 2020 and that the ACTL software and hardware will be handed over to the observatory for routine operation and maintenance.

The total costs of the ACTL project comprise investment money (for computing and networking hardware) and manpower (for software development and system commissioning). The manpower costs are the dominating factor. The total estimated investment for the southern installation is 3.5M € and includes the computing and storage cluster (20%), the network infrastructure (70%), and infrastructure costs (racks, UPS; 10%). The largest piece (the network infrastructure) is dominated by the cabling of the entire array (22% of the total cost) and the central switches in the OSDC (24%) while the money spent on WhiteRabbit network equipment is small (5%). The total estimated investment cost for the northern and southern CTA site is about 5M € when assuming that the costs for CTA-North are 50% of the costs for CTA-South. The manpower costs will be higher than the investment costs. The currently estimated manpower cost is about 12.9M €.

1.4 Summary of Organization

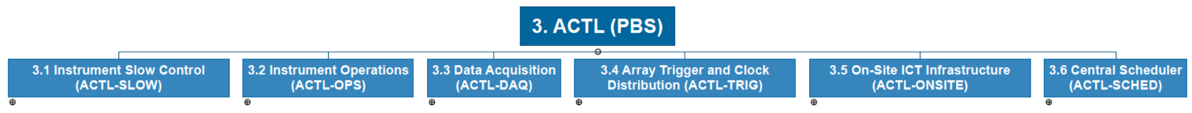

3. ACTL (PBS)
3.1 Instrument Slow Control (ACTL-SLOW)
3.2 Instrument Operations (ACTL-OPS)
3.3 Data Acquisition (ACTL-DAQ)
3.4 Array Trigger and Clock Distribution (ACTL-TRIG)
3.5 On-Site ICT Infrastructure (ACTL-ONSITE)
3.6 Central Scheduler (ACTL-SCHED)

Figure 1.3 – Overview of ACTL Product Breakdown Structure together with the identification number of the corresponding products.

The ACTL work-package is led by a work-package leader (Peter Wegner (DESY)) and deputy (Gino Tosti (University of Perugia)), a Project Manager (Igor Oya (DESY)), and a Systems Engineer (Matthias Füßling (DESY)). Currently, contributions to the ACTL WP are made by groups from 10 countries (list of institutes not yet complete):

Croatia	Rudjer Boskovic Institute, Zagreb [RBI]
France	LAPP, Annecy [LAPP]
	APC, Paris [APC]
Germany	DESY, Zeuthen [DESY]
	Humboldt-University, Berlin [HUB]
India	TATA Institute of Fundamental Research, Mumbai [TATA]
Italy	INAF [INAF]
	University of Perugia [UP]
Netherlands	Anton Pannekoek Institute for Astronomy, Amsterdam [AMST]
Poland	CAMK, Torun [CAMK]
Spain	IEEC-CSIC, Barcelona [BAR]
Switzerland	ISDC, Geneva Observatory, University of Geneva [UNIGE-ISDC]
USA	Iowa State University [IOWA]

The work is organized in six sub-work packages that correspond to the top-level items in the ACTL product breakdown structure (PBS, Fig. 1.3). The six sub-WPs (acronyms: OPS, SCHED, ONSITE, SLOW, DAQ, TRIG) are led by one leader each and have the following scope of work:

Instrument Slow Control Software (acronym: **SLOW**, led by Thierry Le Flour (LAPP))

- slow-control for telescopes, camera subsystems, and auxiliary devices
- GUIs for engineering purposes

Instrument Operation Software (acronym: **OPS**), led by Igor Oya (DESY))

- physics use-cases
- central array control and operation
- monitoring, logging
- error detection and recovery

Data Acquisition Software (acronym: **DAQ**, led by Etienne Lyard (UNIGE-ISDC))

- data readout and bulk data transfer
- online data format
- integration of RTA

Array Trigger and Timing System, (acronym: **TRIG**, led by Amanda Weinstein (IOWA))

- inter-telescope and array-level triggering
- clock distribution and time-stamping

On-site IT Infrastructure (acronym: **ONSITE**, led by Rico Lindemann (DESY))

- on-site network and computing farm
- interface to external network
- remote array-control centre

Central Scheduler (acronym: **SCHED**), led by Josep Colome (BAR)))

- long-term and short-term scheduling
- scheduling of observations and calibration/maintenance procedures

With the beginning of the pre-production and production phase, the work is organized more and more around the development of an overall software architecture, the implementation and test of subsystems, and system integration and verification. Table 1.3 presents the tentative contributions of the groups to the PBS items and central services (such as project management, systems engineering and software architecture). It is noted that the distribution of work is still under discussion and that contributions will depend on the success of funding applications.

	SLOW	OPS	DAQ	TRIG	ONSITE	SCHED	Central Services
INAF	C	C			c		C
Perugia (UP)	C	C					
DESY	c	R		C	R		R
Berlin (HUB)	c	C			c		
Zagreb (RBI)		c					
Annecy (LAPP)	R						
Paris (APC)				c			
Mumbai (TATA)	c						
Amsterdam (AMST)				C			
Torun (CAMK)				C			
Barcelona (BAR)		c				R	
Geneva (UNIGE-ISDC)			R				
Iowa (IOWA)				R			

Table 1.3 – Responsibility matrix for the ACTL PBS (subject to updates). The sequence of the institutes reflects the ACTL manpower expected from their country of origin. An "R" denotes responsibility, a "C" ("c") a major (minor) contribution.

1.5 Work Remaining and Major Risks

In the next months, the ACTL team will work to finalize the hardware and software interfaces of ACTL with the telescope, camera, COM and DATA teams. This convergence will be achieved via a set of workshops with the involved parties, ending up in the final agreement on the interface control documents (ICDs).

Another major work remaining consists in establishing the dynamics for the software creation within the ACTL team. The first step in this direction, to be performed in the next couple of months, is to formalize the software architecture for the whole ACTL system in such a way that the architecture describes all the contained building blocks and internal interfaces well. In parallel, an optimized tooling for software creation, sharing, testing and deployment will be installed and offered to the team. With this framework and taking the existing prototypes and design concepts into consideration, the creation of the ACTL software products will follow. The implementation work is to be performed with an agile approach and will be supported by the tooling system which contains an extensive automatic test suite. Early versions of the products will be offered to the prospective users via dedicated workshops. (For example, early versions of the operator GUI will be offered for evaluation to professional operators.)

A full performance, reliability and failure mode analysis of all ACTL products, including hardware and software, is yet to be done. The analysis of the ACTL software will be performed incrementally on the test bed cluster. This involves the validation of the proposed solutions for the repositories, and the decision on the technology (plain files, databases) for the Cherenkov data repository. This analysis may show that changes in the software architecture or in some components are needed and thus some re-design, implementation and test cycles will follow. Additionally, the integration of those software components to be incorporated into the ACTL system –like low-level telescope control software or the real-time analysis– will be verified with analogous procedures, using simulated versions of the hardware responses when applicable. A plan for the automatic deployment of verified software in the on-site data centres will be devised in the following months. For the case of major software updates, on-site commissioning will be performed by ACTL experts. Minor updates will be performed remotely with the help of local staff.

Once the CTA sites have been decided, specific plans for the deployment of the on-site computing infrastructure, including the computer cluster, the network and other on-site ACTL hardware will be prepared. The associated costs will also be revised.

When in the final stage the responsibility of the ACTL system maintenance is handed over to the observatory staff, the ACTL team will take care of training of the personnel via workshops. Documentation and supporting manuals will also be made available.

The biggest risk for a timely delivery of all the ACTL products is a lack of manpower. A reduced manpower scenario will result in a downgrading of the functionality of the array control software, thus limiting the capabilities or performance of the CTA observatories. To mitigate the risk, a core group of programmers is being assembled and organized via face-to-face and remote workshops, in order to prioritize the main activities and to avoid duplication of work. The risk is further mitigated by optimizing the available manpower using a tooling setup described earlier.

A further risk is associated with the increased complexity of an ACTL system that has to cope with many different interfaces instead of unified interfaces (e.g. to the telescope drive systems). This risk would lead to a system which is difficult to maintain and potentially unstable. This risk is being mitigated by revising and optimising the software design of the ACTL system with the help of professional software architects, by introducing standard software interfaces via the OPC UA protocol, and by aiming for uniform interfaces with instruments via the validation of the ICDs.

2 Design Assumptions

CTA as the next generation ground-based very-high-energy gamma-ray observatory is creating new demands on the ACTL system. Currently considered array designs deploy about O(100) (O(20)) telescopes on an area of roughly 10 km^2 (1 km^2) on a southern (northern) site. The CTA performance requirements and the increased complexity in operation, control and monitoring of a large distributed multi-telescope array lead to new challenges in designing and developing the CTA control software system. In the following, an overview of the assumptions taken into account in order to provide the preliminary design of the ACTL system (see Sec. 3.2) is given.

2.1 Main Drivers for the ACTL design

The CTA Observatory will be composed of Cherenkov telescopes of three different sizes (with typical reflector diameters of 23 m, 12 m, and 4 m, respectively) whose cameras will comprise 1,000–10,000 pixels. One of the main tasks of the ACTL system for CTA is to transport, process, and store these data. Along the way, inter-telescope and array-level trigger schemes and possibly an on-line filtering of events will be used to further suppress the background while selecting electromagnetic showers with high efficiency. For triggering and later association the ACTL system needs to provide a mechanism to timestamp data from the various telescopes with nanosecond precision in an array whose extent implies signal round-trip times exceeding 10 μs. Besides the camera, telescopes comprise further systems (drive systems, CCD cameras, sensors, LED flashers, LASERs, mirror control units etc.) that are needed for the operation and calibration of each Cherenkov telescope. The configuration, control, readout, and monitoring of these devices is another important ACTL task. In addition there are devices at the array level (trigger, LIDARs, weather stations, optical telescopes etc.) that must be operated in connection with the Cherenkov telescopes.

In contrast to current-generation experiments CTA will be an open astronomical observatory. A significant fraction of the total available dark time (about 1300 h per year) will be filled with proposal-driven observations and all observations should be performed in a largely automatic fashion under the control of a very few professional operators. The ACTL systems must support the optimal use of the observation time, diverse operation modes (surveys, pointed observations, multi-wavelength campaigns), the generation of outgoing alerts (by means of a level-A, pseudo real-time analysis of CTA data), the prompt and proper reaction to alerts arriving from other observatories or to targets of opportunity, and the parallel operation of sub-arrays, groups of CTA telescopes operating independently of each other. Since the CTA will go through a telescope deployment phase of several years, the system design will need to provide for an adequate infrastructure that can be incrementally built for such activity. A moderate CTA installation will already surpass the sensitivity of the current experiments and the ACTL system must therefore be very flexible and highly configurable to operate such partial installations and later the complete array with the same success.

The external interfaces to ACTL to all telescope work packages xST, DATA, OBS, INFRA and COM, summarized in Sec. 3.3.2, are also main drivers for the ACTL design.

Site	Trigger Rate for Hadrons (Hz)	LST/event	MST/event	SST/event	SCT/event
CTA-South	39941	0.5421*	1.8260	0.6149	0.6523
CTA-North	17017	1.1895*	2.2048	-	-

Table 2.1 – Monte Carlo results that are input parameters for the data rate calculation. The telescope multiplicities (columns 3–6) refer to stereoscopic events where at least two telescopes triggered. The LST values (marked with a $*$) are not used in the rate calculation as discussed in the text (cf. Section 2.2.2).

2.2 Rates and Data Volumes

2.2.1 Event Rates

The event rates that the ACTL system has to cope with are dominated by

1. the acquisition of images selected by the local camera triggers during observations,
2. the selection of stereoscopic events with the help of the SWAT, and
3. the filtering, compression, processing and storage of array-level events.

All other event types (e.g. camera calibration events or data from atmospheric monitoring devices) occur at a much lower rate.

The signal roundtrip time over a distance of 1 km is about 10 μs at fibre speed, i.e. much longer than the typical buffer depth (few microseconds) in a Cherenkov camera. Camera readout will therefore occur after (i) a local camera trigger or after (ii) a coincidence with neighbouring telescopes (e.g. LSTs or neighbouring MSTs). The single-telescope trigger rates induced by hadronic showers (protons and heavier nuclei) are a few kHz for MSTs and LSTs; it is assumed that telescopes must be read out at rates of about 0.5–10 kHz and that stereoscopy will be enforced while the data are buffered outside the cameras.

Monte Carlo simulations of CTA candidate arrays (e.g. the 2KC array [6]) predict that hadrons fulfilling the stereoscopic requirements (≥ 2 telescopes) will result in array trigger rates of 39 kHz (17 kHz) for CTA-South (CTA-North). Typical telescope multiplicities for such stereoscopic events are about 3.6 (3.4). Table 2.1 summarises the relevant simulation results for a typical southern and northern installation. The data in the table have been used to jointly estimate data rates and data volumes for the DATA and ACTL work-packages.

2.2.2 Data Volumes

The different camera projects aim for readout of camera-triggered events at goal rates in the range $0.6 - 15.0$ kHz. The required rates are in general a factor of 2 lower. The translation of readout rates into data rates requires assumptions about the amount of data that needs to be read out per camera pixel. The results of a joint survey of the DATA and ACTL work-packages among the camera projects are shown in Table 2.2. The table lists the length (in time) of the readout window, the number of samples per unit time and number of bytes used per sample. It is the base for the translation of event rates into data rates.

In the current base line design for CTA, the addition of medium-sized Schwarzschild-Couder Telescopes (SCTs) are treated as an extension, so they are not included in the array designs used here. In the following, the impact of SCTs is discussed separately below.

Transfer between camera and camera server: The estimation of the data rates that need to be transferred between an individual camera and its camera server is based on the following assumptions:

Telescope type	Pixels per camera	Window (ns)	Samples (per ns)	Bytes/ sample	Bytes for header	Bytes/ pixel
LST	1855	30	1	4	5	125
MST-NectarCAM	1855	60	1	4	5	245
MST-FlashCAM	1764	60	0.25	2	5	35
SST-2M-G	2048	100	1	2	5	205
SST-2M-AS	2048	N/A	N/A	N/A	N/A	7
SST-1M-DG	1300	100	0.25	2	5	55
SCT	11328	60	1	2	5	125

Table 2.2 – Readout characteristics of the different camera projects. Readout will occur for a time window between 30 and 100 ns (column 3). During readout, the signals will be sampled at a given frequency (column 4), and each sample will be decoded in a few bytes (column 5). The total number of bytes per pixel (rigthtmost column) is the product of the columns 3–5 plus 5 bytes of header information (column 6). The SST-2M-AS cameras will only provide integrated signals, so the columns 3–6 do not apply in this case.

camera type	goal rate	data rate
LST	15.0 kHz	27.8 Gb/s
MST-NectarCAM	9.0 kHz	32.7 Gb/s
MST-FlashCAM	9.0 kHz	4.4 Gb/s
SST-2M-G	0.6 kHz	2.0 Gb/s
SST-2M-AS	0.6 kHz	0.07 Gb/s
SST-1M-DG	0.6 kHz	0.34 Gb/s
SCT	2.5 kHz	28.3 Gb/s

Table 2.3 – Readout rates of the different camera types. Readout of all pixels (column 2 of Table 2.2) of a single camera at the goal rate (column 2) results in the data rates shown in the rightmost column. The data must be transferred at this rate from the camera to the camera server. The SCT cameras can read out at 10 kHz, but will be regulated by a hardware array trigger to 2.5 kHz. See text for more explanations.

Telescope type	Qty for CTA-South	Nb Tel. /Trig.	Bytes /pixel	raw rate (GB/s)	red. rate (GB/s)
LST	4	1.00*	125	9.3	1.34
MST-NectarCAM	12	0.91	245	16.5	1.48
MST-FlashCAM	12	0.91	35	2.2	1.00
SST-2M-G	24	0.20	205	3.4	0.34
SST-2M-AS	24	0.20	7	0.11	0.11
SST-1M-DG	24	0.20	55	0.57	0.17
SCT	26	0.65	125	36.8	5.38
Total without SCTs				32	4.5
Total with SCTs				69	9.8

Table 2.4 – Data rates expected after the SWAT for CTA-South. The data rates in the two rightmost columns refer to the sum of all telescopes of a given type which contribute to an array trigger. The raw rate (column 5) results when the full waveform is kept for each pixel; the reduced rate (rightmost column) was calculated under the assumption that the full pixel information (number of bytes per pixel in column 4) is kept for 3% of the (image) pixels while 15 bytes/pixel (7 bytes per pixel for SST-2M-AS) are stored for the remaining 97% of the pixels. Please refer to the text for a discussion of the LST multiplicities (marked by a *).

Telescope type	Qty for CTA-North	Nb Tel. /Trig.	Bytes /pixel	raw rate (GB/s)	red. rate (GB/s)
LST	4	2.35*	125	9.3	1.36
MST-NectarCAM	8	1.18	245	9.1	0.82
MST-FlashCAM	7	1.03	35	1.1	0.48
Total				19.5	2.7

Table 2.5 – Data rates expected after the SWAT for CTA-North. See the caption of Table 2.4 for a description of the columns.

1. The transfer must be possible at the goal readout rates.
2. The entire camera must be read out and the data sent without any compression.
3. A header information of 5 bytes length is added to the data of each pixel detailed in Table 2.2.

The multiplicative combination of the number of bytes per pixel from Table 2.2 with the goal rates gives then the data rates (in Gb/s) listed in the rightmost column of Table 2.3. Data must be transferred between a single camera and its camera server at this rate. The estimated rates do not include protocol overhead, so they should be increased by at least a 25 % margin. It is evident that the needs of all camera projects can be served with 4 bundled 10 Gb/s-lines.

In the base line scenario, the total data rate moved between all cameras and their servers amounts to about 80 GB/s for CTA-South. This data rate is similar to the data rate read out of the ATLAS detector after a level-1 trigger.

Acquisition of stereoscopic events: A suppression of the event rates arises from the selection of stereoscopic events. Table 2.4 (Table 2.5) provides an overview of the expected data rates that will occur after the SWAT for CTA-South (CTA-North). Column 6 gives the data rate for transfer of pixel data from all camera servers to a central place when the full waveform is kept for all pixels in those cameras which were involved in the array-level trigger. The total rate is 32 GB/s (19.5 GB/s) for CTA-South (CTA-North), i.e. the SWAT is reducing the rate only by a factor of about 2. It is noted that the estimates presented in Table 2.4 and Table 2.5 does not use the LST multiplicities from Table 2.1 but are based on the assumption that the LSTs will be connected by an inter-telescope trigger that stabilizes the trigger rate of the LST subarray at 10 KHz. The equivalent telescope multiplicities (marked by a * in Tables 2.4 and 2.5) have been set to $10\,\text{kHz} \cdot 4/R$ where 4 is the number of LSTs and R the array trigger rate from Table 2.1.

Processing and filtering of stereoscopic events: A further processing of the selected events in the ACTL system is needed to decrease the event rates and the involved data volumes. The data volume is assumed to decrease due to the fact that the full waveform is kept only for selected (important) pixels while for the other pixels only basic information (e.g. charge integral, time of maximum) is stored. The following assumptions were made (based on [7]):

1. The full waveform is stored only for 3 % of all camera pixels.
2. For the other 97 % of the pixels, 15 bytes (7 bytes for SST-2M-AS) are used to characterise the basic charge and signal shape.

This kind of data reduction results in the data rates shown in column 6 of the tables 2.4 and 2.5. The rate is reduced to 4.5 GB/s (2.7 GB/s) for CTA-South (CTA-North). It is noted that this rate is more than 10 times larger than the rate at which the ATLAS experiment stores data permanently. A factor of 10 suppression is planned before data can be exported off-site.

Extension of CTA-South with medium-sized SCTs: The inclusion of SCTs in CTA-South would increase the data rates that the ACTL system has to cope with. The estimates presented here show an increase of the reduced rate from 4.5 GB/s to 9.8 GB/s. It is envisaged to connect neighbouring SCTs with a hardware trigger (DIAT) that works transparently for the SWAT and reduces and stabilizes the rates of SCTs. In the estimates presented here, a conservative reduction by a factor of 4 (i.e. from 10.0 kHz to 2.5 kHz) due to the DIAT has been applied; ongoing Monte Carlo studies suggest that higher reduction factors (based either on timing coincidence or identification of events as originating from cosmic rays) may be possible. In addition, the time delay before the deployment of the SCT extension will allow the investigation of other data reduction methods and to take advantage of the upgrades/replacements of the initial storage servers with more performant hardware. It is also possible that the SCT data will be intrinsically sparser than those of the DC-MSTs due to the lower rate of photons (Cherenkov and NSB) per pixel, and more amenable to compression or aggressive reduction.

Storage of CTA data in the data repository: The volume of observational data has been estimated under the assumption that the typical observation time is 1314 h per year (15 % duty cycle). A data rate of 1 GB/s during observations results then in a data volume of 4.7 PB per year, so 4.5 GB/s imply a total volume of 21 PB. Storage of data for 2 months requires 3.5 PB. The inclusion of SCTs would roughly double all these estimates unless the SCT data are stronger compressed than those of other cameras as discussed earlier.

Data Export: With one 1 Gb-line and assuming that about 75 % of the theoretical line capacity can be used for data export, about 0.12 GB/s = 3 PB/year can be exported. Clearly, more event filtering and compression during data-acquisition or a temporary on-line data repository would be needed. It is expected that the algorithms for event filtering and data compression will improve during the operation of CTA, so processing on-site is an option.

2.3 Operational Concept

The overall operational concept for CTA defines how the observatory will ensure that good-quality data products are provided to users in response to a scientific observation proposal. In general, the operational concept addresses (i) the submission, evaluation and selection of proposals, (ii) the acquisition of observational and calibration data, and (iii) the processing and release of data products. In the context of this document, step (ii) is the most relevant one and a responsibility of ACTL. Since the overall operational concept is still under development the basic assumptions about step (ii) are briefly discussed in the following.

CTA will be a more complex observatory and will have more users than any other existing installation of ground-based Cherenkov telescopes. The operational concept for ACTL is therefore based on the following assumptions:

1. CTA operations will be carried out from an on-site data centre that is located within or at the rim of the telescope installation. The data centre will host the operators and the technical infrastructure (computers, switches) needed for ACTL.
2. As a rule, the acquisition of observational and calibration data with the CTA arrays will be carried out by a trained crew of professional operators.
3. Remote (i.e. via network access) operation of the CTA arrays will be possible after a series of safety checks and once the local operators have explicitly handed over control. See Sect. 2.3.1 for details.
4. The maintenance of ACTL hardware and software (software upgrades etc.) will be done by additional local personnel who are supported by external experts.
5. The external experts will have a remote control room (i.e. a replica of the on-site control room far away from the CTA sites) at their disposal. The remote control room will be suitably equipped (screens, network) to support remote control and debugging.
6. As a rule, all regular telescope operations (observations, calibration procedures, scheduled maintenance) will be performed under control of a scheduling software (the scheduler, see Sect. 3.2.6). Local scheduling by the operators is possible but not the default operation mode.
7. The scheduler will base its planning on a database with accepted observation proposals, information about the amount and quality of recorded data, the state of the atmosphere, presence of triggers (see below) and further configuration data (array configuration, calibration and maintenance schedules).
8. Coordinated observations using Southern and Northern CTA observatories can be considered for scheduling purposes.
9. Scheduling Blocks (SBs) are the basic and atomic tasks considered when scheduling scientific proposals and calibrations. The minimum duration of a SB and the coordinated execution of different SBs from the same proposal are still to be defined.

10. The Scheduler assumes all constraints defined by the user have to be fulfilled. Responsibility to obtain the expected scientific results based on the predefined acquisition constraints belongs to the user submitting a proposal.
11. External triggers (ToOs, gamma-ray burst alerts etc.) will be filtered, accepted, and immediately included in the scheduling.
12. Recorded data will be subjected to a preliminary analysis aiming to identify signatures (such as "flares") of VHE gamma-rays (real-time analysis, responsability of the DATA WP). The presence of such signals will be communicated to other observatories and to the scheduler (as an internal trigger).

2.3.1 Remote Control

The CTA arrays will be operated by local crew on site. However, the ACTL team plans for an architecture that allows remote control to be technically possible. This remote control would help to provide expert support to the local crew. At least, remote access to the system by ACTL experts would be needed for obtaining monitoring, debugging and logging information, and in addition to permit the emulation of faulty situations. This access could become specially important during the construction phase, when potential system instabilities and errors are to be expected. Remote control would allow that experts provide a better support to the operators, in particular when help is needed on short notice.

The guiding principles and assumptions for such a remote access are presented in the Appendix D.

3 Design and Prototyping

3.1 System Overview

The “Array Control” product is a first-level item of the CTA PBS corresponding to the “Array Control and Data Acquisition” (ACTL) work-package. The ACTL product is further developed in a dedicated ACTL Product Breakdown Structure (PBS, see Fig. 1.3 and Sec. 5.2.1 and A) according to a hierarchical definition of corresponding sub-products. The ACTL organizational structure is grouped as “sub-work packages” around these main products. The ACTL main activities are defined in a full Work Breakdown Structure (see also Sec. 5.2.3 and B), developed to deliver the PBS items.

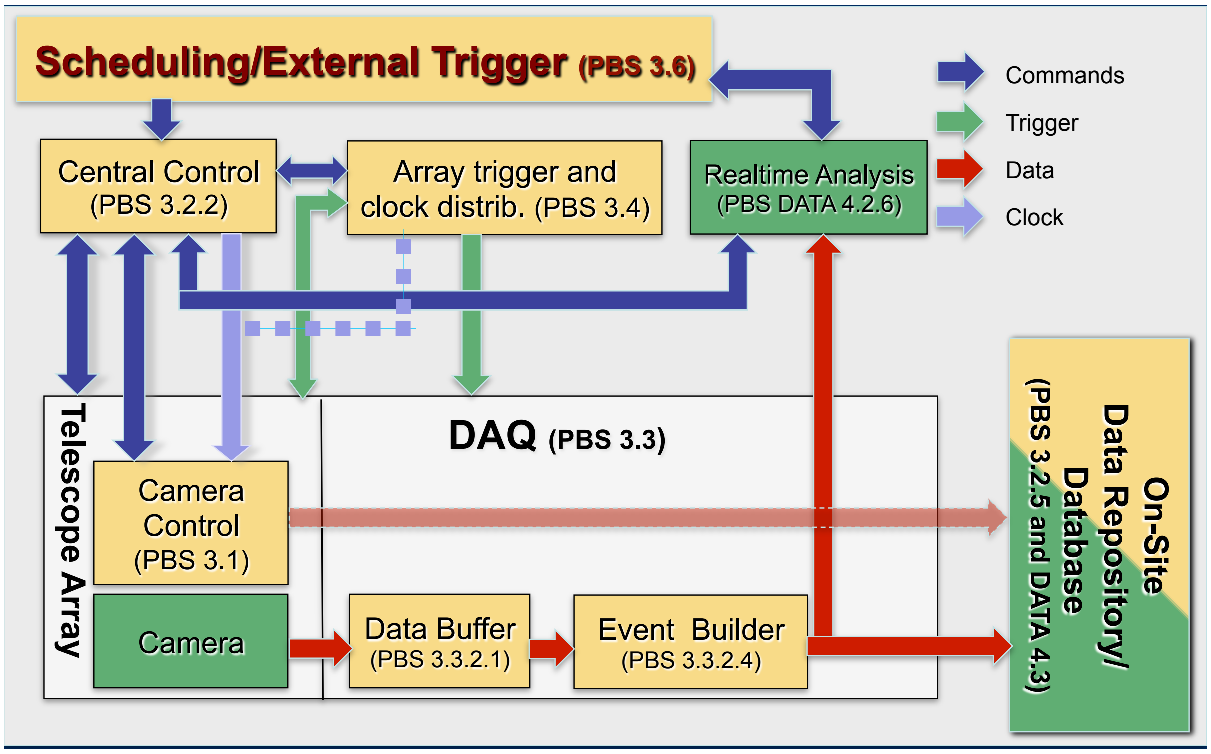

Scheduling/External Trigger (PBS 3.6)
Commands
Trigger
Data
Clock
Central Control (PBS 3.2.2)
Array trigger and clock distrib. (PBS 3.4)
Realtime Analysis (PBS DATA 4.2.6)
Telescope Array
DAQ (PBS 3.3)
Camera Control (PBS 3.1)
Camera
Data Buffer (PBS 3.3.2.1)
Event Builder (PBS 3.3.2.4)
On-Site Data Repository/ Database (PBS 3.2.5 and DATA 4.3)

Figure 3.1 – Overview of Data Acquisition and Central Array Control components (in yellow), including the main data flow as red arrows. In green, components external to ACTL but directly interfaced are indicated. PBS item identification numbers are indicated for each of the components.

Fig. 3.1 shows a schematic view of the main building blocks of the envisaged ACTL system along with the data and control flow. A flexible scheduler (see Sec. 3.2.6) will be used to automatically create observation schedules on different time scales (months, nights, hours) in order to ensure that observations can be performed without human intervention, at high efficiency and optimized for maximum scientific output of CTA. The off-site service of the scheduler will compute and select the best long-term plan for a complete season, which will be used afterwards at the observatories to compute the short-term schedule. The off-site service is a time consuming and complex task to be done prior to the proposals execution on-site. For this purpose, the off-site service (or LTP) will have to provide a specific GUI to let operational unit staff manage the decision complexity in an understandable way (which is known to be a major bottleneck for large infrastructures). The scheduler will be able to react to external and internal triggers and will be tightly coupled with the system to react to changes of the observation conditions and the state of the array.

The optimized observation schedules are executed by the central control system (Sec. 3.2.2), which prepares, starts and stops observations and handles error conditions. The central control interacts with all entities in the system, in particular with the array trigger (see Sec. 3.2.4) and the individual telescopes (see Sec. 3.2.1) to organize the acquisition of showers observed by more than one telescope. It also implements the monitoring of all hardware devices (see Sec. 3.2.2) independent of on-going observations and provides graphical displays and user interfaces. The central control system will also provide clock distribution mechanisms (see Sec. 3.2.4) for synchronization to a global clock, allowing all physics data, typically the Cherenkov camera events, to be timestamped with high precision. Data sources (typically monitoring data sources), for which less precision is required, will use times kept on local clocks.

The data streams generated by the cameras after local camera triggers dominate the overall data rate of CTA. This data acquisition (DAQ, described in Sec. 3.2.3) comprises the camera readout, the buffering of the read-out data, the processing of array trigger decisions, the building of camera-dependent events and filtering of interesting events to reduce the overall data volume. A compact and flexible data format is needed at this stage since the amount of data to be read out depends strongly on the camera type which in turn determines parameters like the camera trigger rate, the sampling rate and the decision as to whether waveform information is being included or not.

Acquired data will be stored in an on-line storage system, which will consist of a data repository and one or more databases mainly for control and monitoring data. A level-A analysis (pseudo real-time analysis) pipeline (see also the corresponding section in the DATA TDR) is used to monitor the science performance during data taking. A preliminary data analysis is applied to the data to detect high-states of sources and transient events. Based on the results of the level-A analysis, feedback will be sent to the scheduler which will generate appropriate actions.

Figure 3.2 depicts a schematic view of the overall flow of data in the ACTL system. Three streams are involved: the data from the photo-detector cameras, the slow control and monitoring data of other devices like the drive, alignment and camera control systems , and the telescope trigger data. The highest volume is due to the camera data but non-negligible volumes are involved in the instrument slow control/monitoring data and trigger systems. In the next sections further details of these streams are provided, including details on how the estimation of the data volumes is performed. The relevant subsamples of the data from these streams are merged by the analysis pipelines [8] for both the level-A and level-B analysis and the final, off-line level-C analysis.

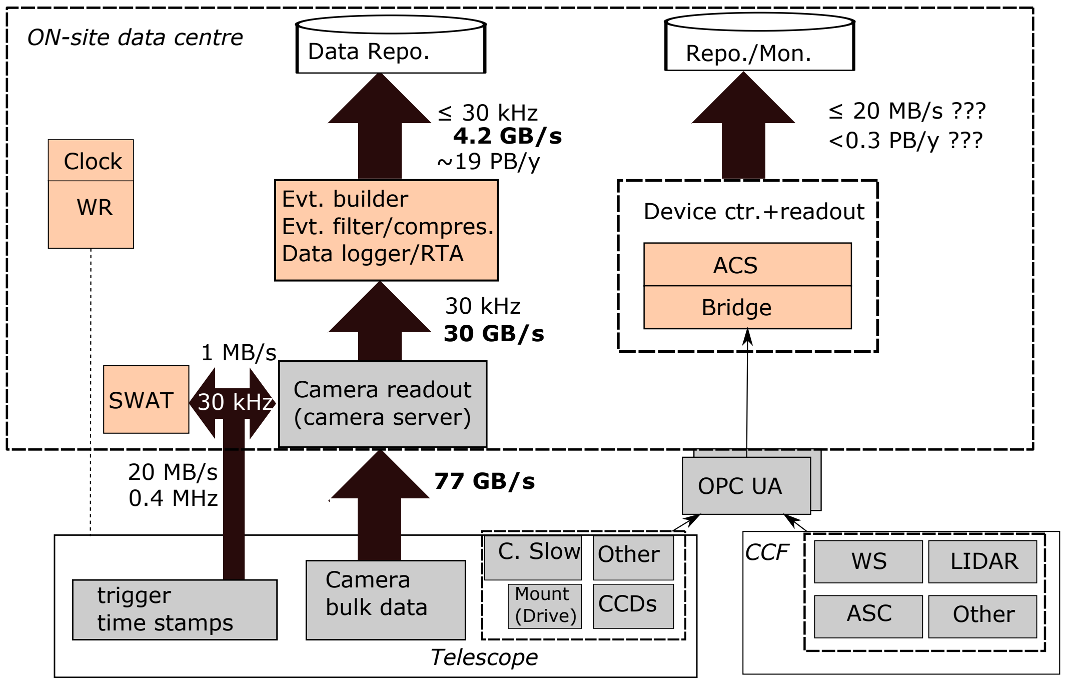


Figure 3.2 – Schematic view of the ACTL system overall data flow, including the direction of the flow and the estimated involved rates and volumes.

3.2 Design of ACTL

The implementation of the ACTL product is organized around six sub-work packages, each of them with an associated top-level product (see Fig. 1.3) and corresponding sub-products. In this section, a description of the design for those products is provided.

3.2.1 ACTL-SLOW: Instrument slow control software

In this section the product ACTL-SLOW (Instrument Slow Control Software) numbered *3.1* in the CTA PBS structure, is described. ACTL-SLOW provides software for the control of telescopes and central calibration facilities, and for their integration in the central control, configuration and monitoring systems (described later in Sec. 3.2.2).

Device Integration

The control of each individual telescope and additional auxiliary devices that will make up each of the CTA arrays will be integrated within the central array control software. The various interfaces and firmware required to access the devices within these assemblies force a setup with a relatively inhomogeneous environment, where some units will be controlled from different Programmable Logic Controller (PLC) types, others from specific Linux environments, others with embedded computers onboard the telescopes, and others under Microsoft Windows environments.

On the other hand, ACS, the software framework to control CTA (see Sec. 3.2.2), will be operated in the specific Scientific Linux [1] distribution that is also chosen for the CTA array control cluster. The ACS software must interface the low-level software layer of the devices, which in general may run in one of the various environments as described above. ACS provides a mechanism to encapsulate the communication with the processes corresponding to these devices with its DevIO classes (see Fig. 3.3). However, for each of the various environments and device firmware, specialized DevIO classes would be required, implying in many cases that intense programming activity would be required on both the client and server side for the communication with low-level software layer of devices.

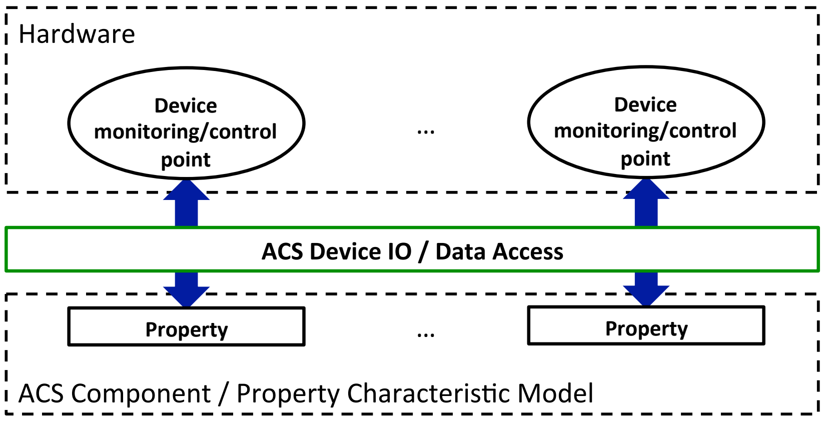

Hardware
Device monitoring/control point
...
Device monitoring/control point
ACS Device IO / Data Access
Property
...
Property
ACS Component / Property Characteristic Model

Figure 3.3 – The ACS DevIO concept. The DevIO bridges device monitoring (for example a sensor) and control points (for example an ON/OFF switch) to the ACS standard elements called “properties”.

In order to avoid complications and potential unstable results, the ACTL team proposed an alternative approach, namely using a standard software layer based on the OPC UA standard (see Sec. 3.2.2) as the access point to the devices. Given that OPC UA implementations are able to run in all the environments mentioned above, it is possible to use the OPC UA protocol to access such servers from

[1]ACS is also available for RedHat and adaptable for CentOS linux distributions. On-going initiatives exist within the ACS community to port ACS to other linux distributions such as Ubuntu.

the ACS environment, using just one DevIO library. The DevIO library was extended to cover communication aspects, e.g. method calls and alarm transmission, in a standard manner additional to those of the standard DevIO implementation. Following this procedure, the CTA assemblies will integrate into the CTA control system via OPC UA servers, which are created by the instrument teams that provide the hardware, whereas the ACTL team will create software bridges to the ACS-based central control framework. Alternatively, a direct ACS integration without OPC UA is also possible for a device team that elects to provide the required DevIO library and corresponding ACS bridge itself. For the case of the cameras of the telescopes the approach is the same in what respects to the slow control functionality: The configuration of the devices, alarms notification, transmission of commands from the central control of the array, etc are performed with the same mechanism: via ACS down to one or more OPC UA servers close to the hardware, or as an alternative, using a dedicated DevIO library instead of OPC UA. The exception to this is the transmission of the camera pixel data and trigger timestamps, as described in sections 3.2.4 and 3.2.3.

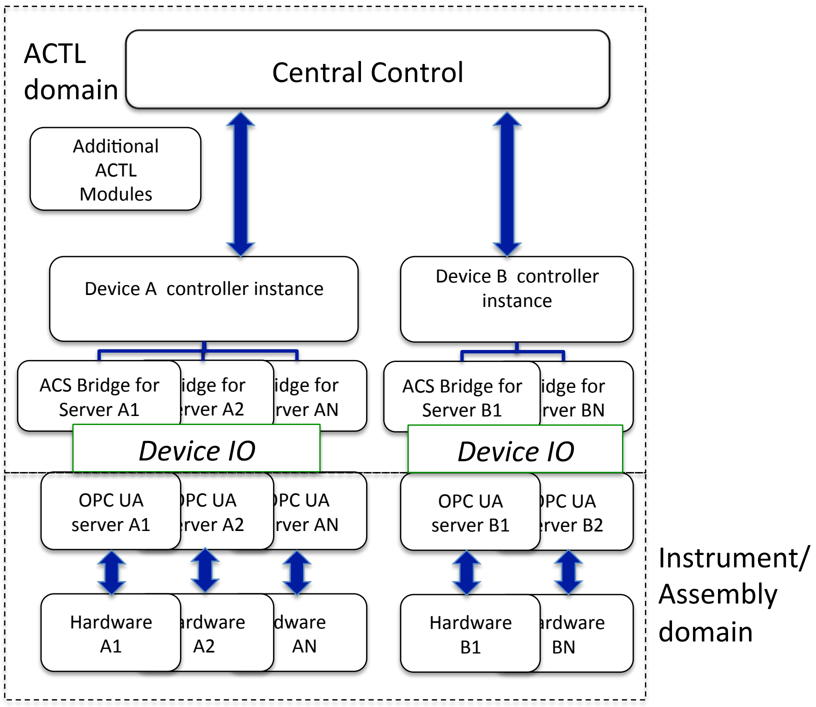


Figure 3.4 – Schematic view of the baseline integration of hardware devices and assemblies into the ACTL system. The domain of the instrument teams comprises the hardware and OPC UA servers, whereas ACTL is in charge of the DevIO and bridges.

Figure 3.4 shows as an example a schematic view of the general architecture for the integration of the device control into the ACTL structure. The following elements are depicted in the diagram:

- **OPC UA Server** The access to a hardware component is granted via OPC UA servers. One assembly may make use of more than one OPC UA server: for example in a telescope a PLC may govern the power supply of a device whereas the device itself is steered via another mechanism. Nevertheless, very often just one OPC UA server will be involved per assembly. These OPC UA servers provide device-specific functionality.
- **ACS Bridge** Shortcut for ACS to OPC UA bridge. The ACS bridge uses a standard library to map the OPC UA nodes into ACS standard elements: OPC UA nodes to ACS properties, methods, and alarms. A bridge will connect to a specific OPC UA server, providing device specific functionality. Sometimes several bridges will connect to the same OPC UA server (e.g. an OPC UA server managing the power of a whole CTA product such as a telescope). It is possible to configure the bridge to not link all OPC UA nodes to ACS elements, but only those which need to be controlled and monitored by ACTL. Some type conversion can be performed in the bridge when limited

functionality on the OPC UA side exists or if the particular OPC UA technology is not capable of implementing the full OPC UA standard, but only a fraction of it, for example, certain PLCs, where OPC UA data items are supported, but no methods or alarms.

- **Controller instance:** Controllers are running ACS component instances that implement a common interface definition for a particular device type. The ACTL team is currently discussing the possibility that controllers implement a state machine, either common for all the controllers, or specific to the assembly. These controllers translate the command from the central controller, including those originating from the automatic scheduler, to the assembly specific commands. A design based on that of ALMA Controller/Device Hierarchy and the ACS software to generate state machine code from a textual or graphical model is being considered in this context.
- **Additional ACTL modules:** Further ACS elements will be deployed in the ACTL system, among others: Master Component state machines that govern the software state of a subsystem, central logging and alarm tools and operator graphical user interfaces.

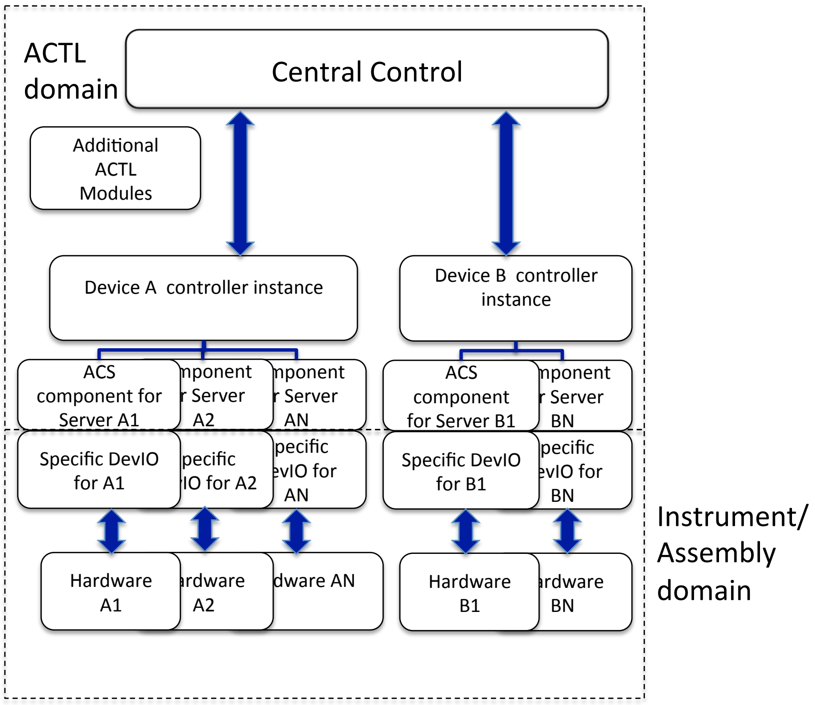


Figure 3.5 – Schematic view of the alternative integration of hardware devices and assemblies into the ACTL system using dedicated DevIOs, instead of using the baseline approach of using OPC UA servers as depicted in Figure 3.4. The domain of the instrument teams comprises the hardware and DevIO classes, whereas ACTL is in charge of ACS components.

The ACTL WP has provided a document with guidelines for the creation of OPC UA servers [5]. The current approach in CTA is to develop the OPC UA servers according to a well-defined interface description, either as dedicated code provided by the hardware teams or by purchasing server firmware along with the device. A complementary/alternative approach currently under evaluation is a tool that provides an environment to describe any device as a set of data points (i.e. public variables), control commands and device protocols in XML files. When combined with OPC UA, it enables hardware integration in an automatic way. In simple cases, this environment can enable the developer to avoid writing dedicated code for each device to integrate. In complex cases, it is possible to incorporate dedicated code, via a plug-in, without any knowledge of OPC UA internal mechanisms.

The OPC UA servers are then interfaced to the ACS bridge components implemented with the ACS Java API, however C++ and Python ACS implementations could be used as well[2]. As for OPC UA servers, the running environment will in general limit the choices at hand. For example, in the case of PLCs the

[2]Currently, only a Java DevIO OPC UA library exists.

tools provided by the hardware vendors will be used, whereas for the applications running in PCs, two options are considered in CTA: Java software development kits (SDKs) from Prosys and C/C++ SDKs from Unified Automation.

In some specific situations, e.g. when special external libraries or protocols are needed for communication to a device, hardware teams may want to implement the device control software directly in the ACS framework. In this case, the ACS bridges themselves provide the functionality that would be expected from the OPC UA servers.

Standard DevIO library for OPC UA servers

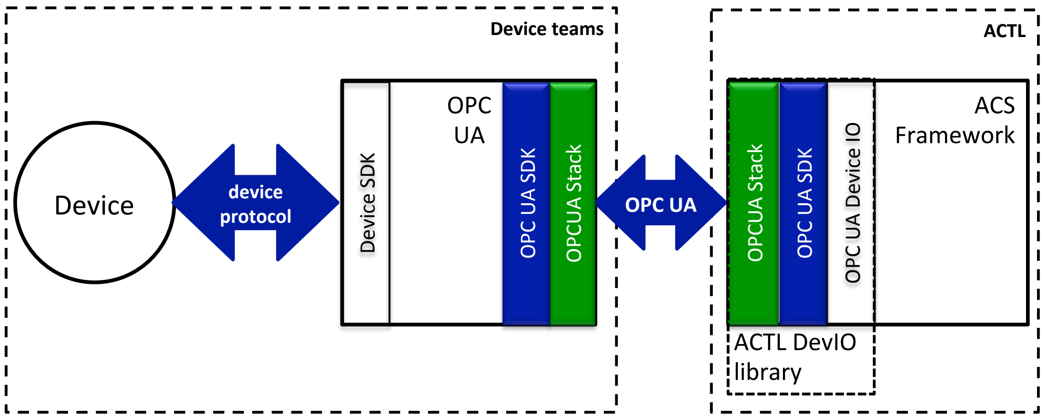


Figure 3.6 – The standard DevIO library for OPC UA and ACS communication. The device is accessed from the OPC UA server via any device specific protocol (for example, serial line communication). The OPC UA server communicates with the ACTL ACS based software via the OPC UA protocol thanks to the DevIO library, which internally uses the OPC UA libraries provided by *Prosys*. See text for further details.

The CTA DevIO library is a generalization of the original ACS concept based on the bridge design pattern. It makes use of the Prosys SDKs for Java OPC UA clients creation and this can be used as a generic means to communicate from ACS Java bridge components to any OPC UA server, as illustrated in Fig. 3.6. This library includes support for the following functionality:

- **Data Item and Property support:** Enables the association of a specific OPC UA data item node with an ACS property. The library uses the OPC UA subscription mechanism to keep the property value updated and to trigger data-change reactions, which allows the usage of value-change triggers in the instrument monitoring system (see Sec. 3.2.2), or custom created reactions to value changes (such as transmission of a new CCD image when it becomes available in the server). Support for specialized types such as enumerations, or special OPC UA data types (like UInt16, UInt32 etc.) is also provided. This *Data Item and Property support* would constitute the original scope of a specific realization of a DevIO.
- **Method support:** Supports both synchronous and asynchronous invocation of OPC UA methods.
- **Alarm/Notification support:** Allows the transmission of notifications and alarms from the OPC UA servers to the ACS bridges. OPC UA Alarms can be filtered according to their family, source, and other parameters. The alarms can thus be transmitted into the ACS alarm systems if required, depending on the bridge configuration.

ICD/Code generation

The ACTL external interfaces (see Section 3.3.2) are described in the Interface Control Documents (ICDs) database. These documents are 'contracts' between ACTL and the external device teams and include information about how to control and monitor the external resource hardware and software components by ACTL. This information if provided also in the form of tables can be easily used to automatically generate the ACTL software interface code (the ACS bridge, see previous section). This approach has been used in ALMA and was successfully tested in the ASTRI/CTA SST-2M prototype (see Sect. 3.4.7 and also [9]). The ICD automatic code generation approach should ease the implementation and debugging of thousands lines of mostly repetitive code.

Engineering GUIs

An engineering graphical user interface (GUI), not to be confused with the operator GUI described in Sec. 3.2.2, allows the display and modification of parameters the normal user/operator should not need to adjust (e.g. pointing model parameters, servo loop parameters). It is primarily intended for testing during the commissioning phase of the devices or of a single telescope before it is added to the CTA Array. Also, the engineering GUI could be useful in case of maintenance of a hardware component, even if dedicated GUIs are provided together with the devices (e.g. in case of PLCs all companies are providing dedicated SDK and Configuration tools). The design of the engineering GUI for each device should follow the schedule of the device fabrication to be ready to enable the engineers to perform complete testing and commissioning of the individual device control modules. Given that all the device parameters to be monitored and controlled by ACTL are detailed in the ICDs, the engineering GUIs can be developed in a uniform way, with most of the code being automatically generated. In Sec. 3.4.1, prototypes created by the ACTL team and tested in real operation are described. A general approach for all of the engineering GUIs will be provided based on the prototyping activity. A coherent approach to provide the same look and feel of engineering GUIs is indeed crucial for a later maitenance of the system.

3.2.2 ACTL-OPS: Instrument operation software

In this section the product ACTL-OPS (Instrument Operation Software), numbered *3.2* in the CTA PBS structure, is described. The main responsibility for this sub-work package is to provide support for the usage of software frameworks in ACTL, to produce the central control system software of the array and to provide the services for system monitoring and configuration, including the associated databases.

Software Frameworks

The nature of CTA as a worldwide collaboration of universities and research centres, with different software development cultures, implies that the creation of the ACTL software can become a complicated task. A way to reduce these complications from the beginning can be achieved by the selection of common software frameworks, that promote merging the disparate styles and approaches into a uniform software architecture and reduce the manpower needed for the development and maintenance of the ACTL code. Two software frameworks have been chosen as a basis for the ACTL software: the ALMA Common Software (ACS) as a higher-level software framework for supporting the application layer and OPC UA as lower-level software framework for standardised hardware access.

ACS ALMA Common Software (ACS, [10]) was developed as a common software framework for the Atacama Large Millimeter/submillimeter Array (ALMA, [2]), which is a joint project of a worldwide collaboration and an installation with many similarities in operation and complexity to those of CTA. ACS is a distributed middleware framework, developed for running under the RedHat Enterprise Linux operating system (used by ALMA), and its re-distributions Scientific Linux and CentOS[3], and is available under the GNU LGPL license. It provides a well-tested platform for ACTL that embeds standard design patterns, avoids duplication of effort and enforces use of common constructs. ACS is based on a distributed container component model, which implements the common patterns as Common Object Request Broker Architecture (CORBA) objects in any of the supported programming languages Java, C++, and Python.

[3]Some initiatives to port ACS to other Linux distributions like Ubuntu are ongoing within the “ACS community” [11]

The usage of standard communication software, which in particular enables the exchange of objects, ensures that the distribution of the control software services and the applications over different hosts can be realized in an easy and general way. Location transparency enables deployment of components to be specified at system startup time, and to be easily changed to free performance bottlenecks or make better use of underutilised hardware resources. The ACS components form the basis for all high-level control entities to be developed for CTA. In particular ACS already provides common CORBA-based services such as remote and distributed logging mechanisms, alarm and error messaging, a configuration database model, process control, discovery and access to remote resources, and support for a scripting language for control. Additionally ACS supports a mechanism to encapsulate the lower level access interfaces in an abstraction layer, which uses the "bridge" design pattern[4] and is realised via Device IO classes (see Sec. 3.2.1 for more details). This allows ACS components to be created without being bound to a special communication mechanism. This is of particular importance for interfacing the higher-level software layer of ACS with the lower-level software layer OPC UA, which provides standardised hardware access by allowing the use of a DevIO library for any communication between these two frameworks. See [1] for further details on the proposed usage of ACS for CTA.

OPC UA The CTA arrays will comprise a multitude of different hardware devices (CCD cameras, mirror actuators, drive systems, high-voltage systems, calibration units, weather stations, etc.) that have to be controlled and/or monitored. Given the many different hardware devices with different hardware buses, a well-defined software bus is chosen, which delivers a standard software interface to readout and control the devices in a common way and hides the details of the underlying hardware and operating system as well as the low-level programming details. In addition, possible hardware changes will not affect the higher layers of the communication and control software. For CTA, the Object Linking and Embedding for Process Control Unified Architecture (OPC UA) has been chosen. OPC UA is based on the OPC specifications, first released by the OPC Foundation in 2006, which provide a basis for a convenient and efficient linking of automation components with control hardware and field devices in a standardised way. OPC UA is a cohesive, secure and reliable cross platform framework for access to real time and historical data and events. This interoperability layer unifies information exchange and provides a common interface for controlling processes. OPC UA is an industrial standard, designed to be independent of the operating system, and comes already with many devices such as automated systems driven by PLCs (e.g. the telescope drive systems). Another advantage is that the most diverse OPC UA components of different manufacturers can work together with no additional programming needed for the adaptation of the interface between components. The main idea for using the OPC UA standard in CTA is to allow us to characterize a hardware device in an OPC UA server by means of the so called "data-point items", which are standard elements that describe the characteristics of a device like a sensor readout or a status flag in a standardised way. Thus, all the public information of the device is directly accessible via client applications. One such client application is implemented in ACS using the bridge design pattern and provides a layer to access the device information (see Sec. 3.2.1 for more details). In this way, the OPC UA software framework used as a standardised hardware interface layer complies with the main needs of CTA for operation and readout of the instruments while hiding their specifics. The OPC UA interface is the baseline solution for the slow control, monitoring and readout of all devices in CTA [5], with a few exceptions. The bulk data of the Cherenkov cameras is not transported via OPC UA, but transfered via mechanisms based on ZeroMQ protocols (see Sec. 3.2.3 for more details) due to the particular requirements on high throughput and performance. In order to assess the proposed OPC UA technology and to acquire experience well before the CTA software production phase, various CTA teams are using OPC UA with prototype systems to write server and client applications in simulated and real environments. See [3] for further details on the proposed usage of OPC UA for CTA.

Central Array Control System

The design of the central array control system is being created under the global ACTL Architectural Design document [12]. A preliminary version of that architecture is depicted in Fig. 3.7, where the main

[4]The bridge pattern is a design pattern used in software engineering which is meant to "decouple an abstraction from its implementation so that the two can vary independently"

[5]Direct ACS interfaces for devices are possible, but not the standard choice. Also in this case, the interface would appear transparent to the ACTL system using the same client applications as for OPC UA. See Sec. 3.2.1 for more details.

functional blocks are shown. The main components of this architecture affecting the central array control system are (associated item numbers are those used in the ACTL PBS):

- **Operator/Scientist GUI -** standard interface for CTA operator (Item 3.2.2.1), see later for details.
- **Coordinator -** coordinates the different subsystem operations (Item 3.2.2.2.1).
- **Sequencer -** interprets the scheduler and user interface commands and issues the orders to the low level software (Item 3.2.2.2.2). These orders are high level ACS Python scripts obtained from a *command library*.
- **Resource manager -** acquires, releases and monitors all main subsystem resources (e.g., telescopes, on site archive resources, computing resources). Each subsystem resource has its own resource manager that acquires, releases and monitors the assemblies that are parts of the subsystem. Thus the Central Resource Manager is at the top of a hierarchical tree of managers. It collects and aggregates information from all the subsystem resource managers making them available to the coordinator and scheduler packages that can use this information to plan system operations (Item 3.2.2.3). This item includes the management of sub-arrays (see later).
- **Monitoring -** retrieves monitoring data from instruments and stores it into the repository (Item 3.2.5.1.1). Also verifies the status of the environment and array hardware and provides feedback to the Scheduler.
- **Alarm -** categorizes, filters and transmits alarms from sources to the operator, and stores them for further reference. It provides automatic means and support for alarm recovery.
- **Logger -** manages the various level logs of the system, from filtering to automatic cleaning of the old data (Item 3.2.5.1.1).

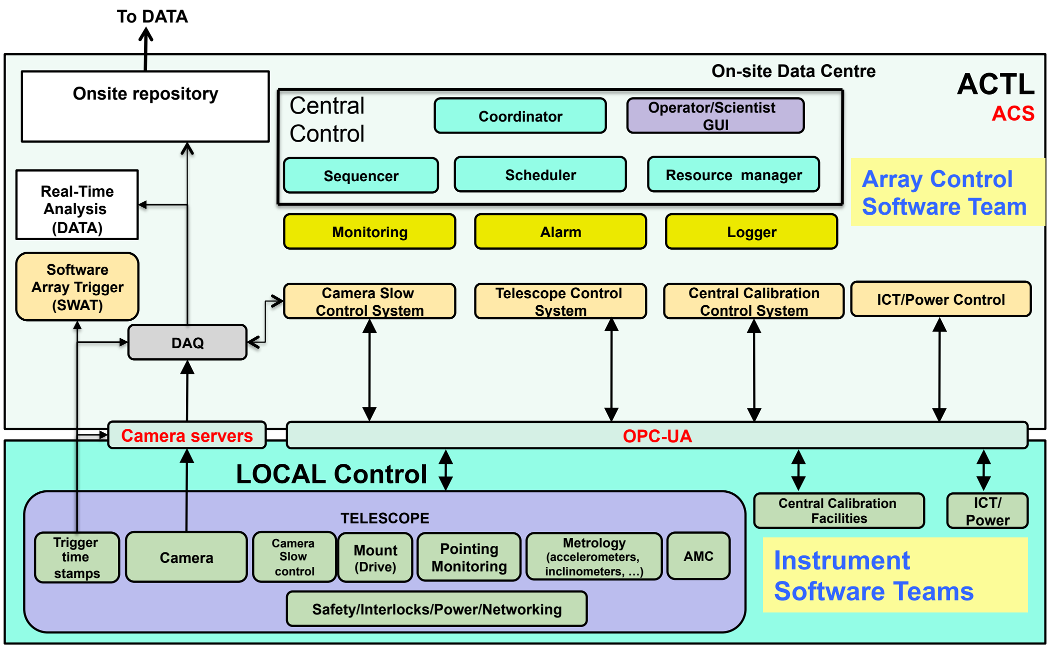


Figure 3.7 – Conceptual view of the overall ACTL architecture and its integration with the instrument software and ACTL main functional blocks. The upper half of the diagram includes the responsibilities of ACTL, and represents processes executed in the On-Site Data Centre. The lower part corresponds to the responsibilities of the instrument software teams (xST and COM).

Figure 3.8 illustrates the inter-relation of the components of the central control system with the services and other components of ACTL in a typical night operation sequence.

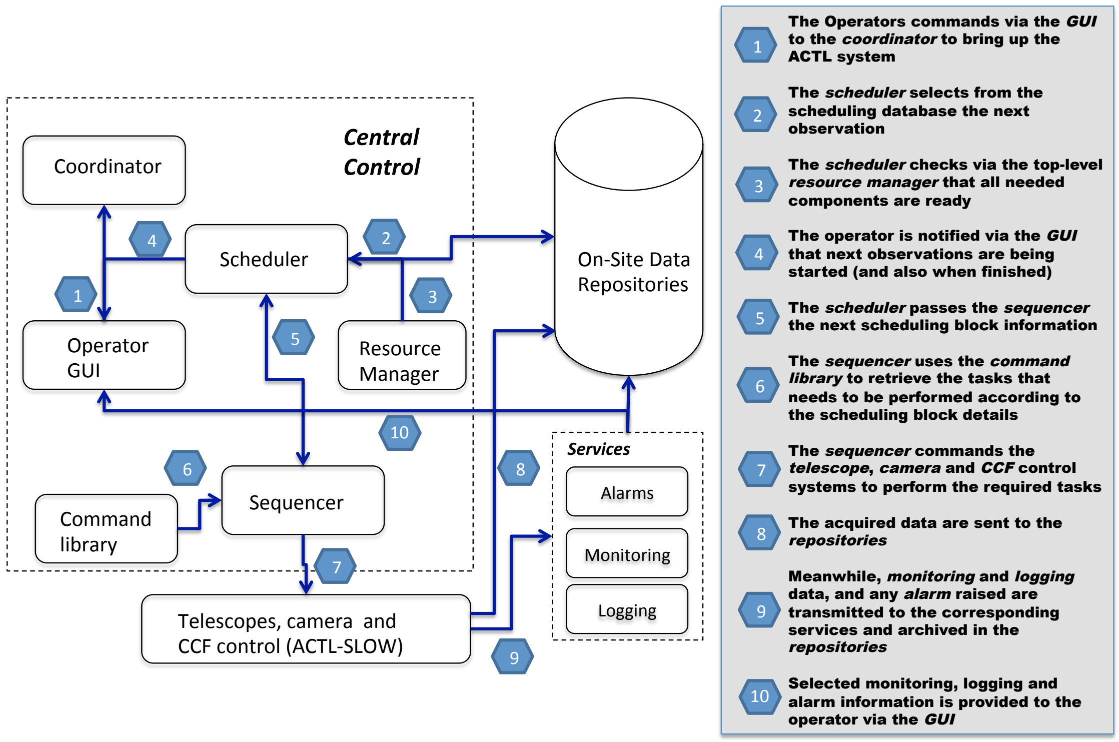

Central Control
Coordinator
Scheduler
Operator GUI
Resource Manager
Command library
Sequencer
On-Site Data Repositories
Services
Alarms
Monitoring
Logging
Telescopes, camera and CCF control (ACTL-SLOW)
1 The Operators commands via the GUI to the coordinator to bring up the ACTL system
2 The scheduler selects from the scheduling database the next observation
3 The scheduler checks via the top-level resource manager that all needed components are ready
4 The operator is notified via the GUI that next observations are being started (and also when finished)
5 The scheduler passes the sequencer the next scheduling block information
6 The sequencer uses the command library to retrieve the tasks that needs to be performed according to the scheduling block details
7 The sequencer commands the telescope, camera and CCF control systems to perform the required tasks
8 The acquired data are sent to the repositories
9 Meanwhile, monitoring and logging data, and any alarm raised are transmitted to the corresponding services and archived in the repositories
10 Selected monitoring, logging and alarm information is provided to the operator via the GUI

Figure 3.8 – Schematic view of the interaction of the central control components and other ACTL elements during a simplified operation sequence in an observation night.

In the recent past, the efforts of the ACTL work-package has been focused on determining the main building blocks and the main goals of the system. The requirements [13] and specifications [14] of ACTL provide some valuable input. However we recognized that for building the Architectural Designs we lacked a common knowledge and many specifications of the system, thus the ACTL *use cases* task (the WBS item 3.2.6.1 of ACTL) has elaborated a list of use cases that will constitute the input for this document. These use cases are the functional description of what needs to be done by the ACTL system (not how to do it). Each use case is a standardized text description of the interaction of an actor (operator, the automatic scheduler) and the system (ACTL as a whole, or a part of ACTL like the central control), using some predetermined categories and keywords. The use cases are stored in a wiki and a snapshot of the existing ones at the time of writing the TDR is provided in Ref. [15]. In addition to their role in the definition of the architecture, the use cases will support any RAMS analysis.

A simplified view of carrying the overall data flow of the ACTL system was presented in Fig. 3.2 of Section 3.1. In this section we present a more detailed view of the current design of the overall data flow (see Fig. 3.9).

Operator GUI One very special component of the central array control system is the operator interface. For its design, considerations of technology but more importantly usability by and interaction with the operator have to be taken into account (see also goals B-ACTL-2090 and B-ACTL-2100). The large scale of the array to be controlled (approximately 100 telescopes) will require a careful design of the system and choice of technology. For the wealth of accessible information and controllable parameters, *human factors* in the interaction of the operator with such a GUI have to be considered early in the design phase. In that regard, very specific knowledge is required. A brainstorming meeting with experts in Human-Computer Interaction (HCI) is planned to provide input for the subsequent development. This will be followed by a more detailed design, test and verification plan, which will involve professional operators of running experiments and observatories to help with the highest-level design of the Operator GUI and then to verify whether the selected implementation fulfills expectations. Currently, web-browser-based technologies are being investigated for the operator interface, which provide flexibility in operating the array. Some of these technologies are being tested on the telescope prototype projects (see Sec. 3.4.7).

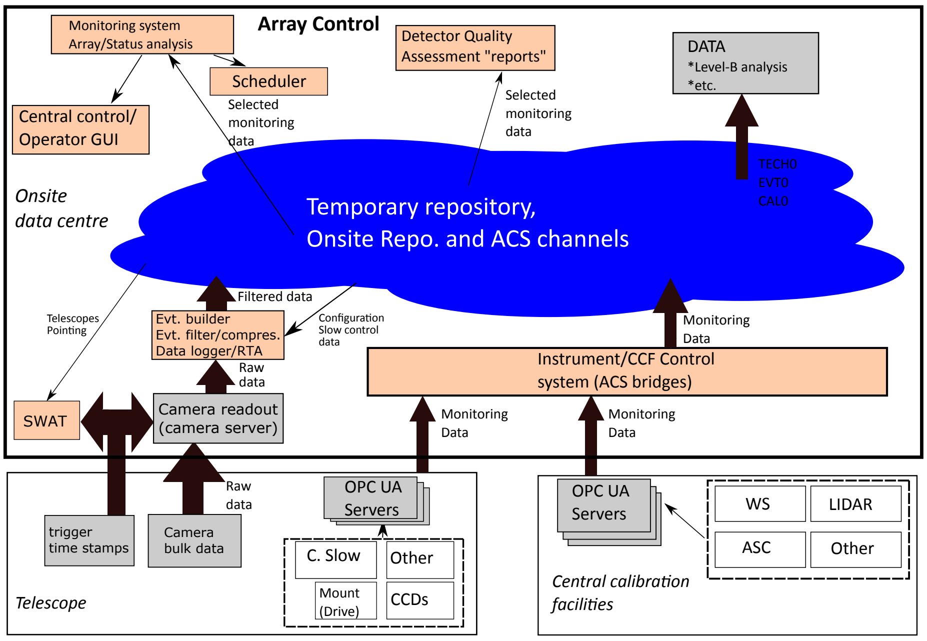

Array Control
Monitoring system Array/Status analysis
Scheduler
Selected monitoring data
Central control/ Operator GUI
Detector Quality Assessment "reports"
Selected monitoring data
DATA *Level-B analysis *etc.
TECH0 EVT0 CAL0
Onsite data centre
Temporary repository, Onsite Repo. and ACS channels
Filtered data
Telescopes Pointing
Evt. builder Evt. filter/compres. Data logger/RTA
Configuration Slow control data
Monitoring Data
Instrument/CCF Control system (ACS bridges)
Raw data
SWAT
Camera readout (camera server)
Monitoring Data
Monitoring Data
Raw data
OPC UA Servers
trigger time stamps
Camera bulk data
C. Slow
Other
Mount (Drive)
CCDs
Telescope
OPC UA Servers
WS
LIDAR
ASC
Other
Central calibration facilities

Figure 3.9 – View of the ACTL system's overall data flow. Data flow from the data acquisition (DAQ, Cherenkov data) and slow control systems (monitoring data) into the "cloud" that is composed of the ACS notification channels as well as the temporary and on-site repositories. Depending on the latency needed and volume of the data to be transmitted by affected systems, data are transmitted via the ACS notification channels (latencies on the scale of some seconds or less, low data volumes), or via dedicated mechanisms such as ZeroMQ for the DAQ and corresponding interfaces to the repositories (larger latencies, large volumes). The specific choices for the different system components will be determined from the specifications detailed in the interface description. SWAT stands for "software array trigger", CCF for "central calibration facilities" while WS for "weather station" and ASC for "all sky camera" are examples of CCF devices.

Sub-array management Sub-array management conceptually describes the task, provided by a software component, that allows the central control system to maintain a list of sub-arrays of telescopes which are configured and controlled as a unit. An identifier is given to each sub-array and can be used by the sub-array management component to get generalized telescope information or to apply a certain observation mode configuration to a given sub-array. Sub-arrays are created at run-time following the scheduling requests by the short-term scheduler or via the operator GUI. After a sub-array is defined, all subsequent configuration and control commands for the observation refer to the sub-array identifier.

Electronic logbook The electronic logbook will allow the operator to record observing and operation activities and events. This includes a detailed list of science observations, information about the environmental conditions, logs of technical operations carried out on the telescopes, the list of calibrations performed for a given night, and specific technical problems faced during the observation. Such a logbook is a very important tool for keeping information for observing statistics, telescope mode usage, downtime breakdown, etc. The ACTL team is evaluating the possible usage of a modified version of the electronic logbook from ALMA for such purpose (see Sec. 3.4.2)

ACTL scripting language

The ACTL scripting language is used to define observing scripts, as well as to serve as a suite of interactive commands to be used by hardware engineers for testing or debugging the equipment, or by operators for developing new observation procedures. ACTL is currently evaluating whether the ALMA Control Command Language (CCL), based on python, can be adapted for such a role.

System Monitoring and Configuration

A *monitor point* defines the value provided by a read-only property over time, while a *control point* defines the value provided by a read-write property. As an example, the wind speed (measured value and corresponding timestamp) from an individual weather station constitutes a monitoring point. A control point, on the other hand, could be the voltage value set in a power-supply. Monitor and control points (just monitoring points hereafter) are usually sampled either with a regular rate, usually of the order of 1 Hz or slower, or whenever a value changes more than a predefined threshold. Monitoring is not restricted to devices but is also applied to logical monitoring points, for example to the status of software elements such as the components of the DAQ and the RTA, and of services such as databases. The final number of monitor points for CTA has not yet been determined, but it is expected that around 10^5 monitoring points must be sampled at rates on the order of 1 Hz.

The effective analysis and interpretation of this large quantity of data depends on maintaining links between a device's monitor points and the configurations, both current and historical, that determine where and during what periods the devices was installed and operating in the observatory. If, for example, a device is raising alarms or issuing anomalous values, it is important to know what part of the system it is affecting. If a telescope's pointing should become unsatisfactory, it is essential to know what pointing model was in use at that time. For reasons such as these, we treat the two subjects, configuration and monitoring, together.

The ALMA Observatory has created the Telescope Monitoring and Configuration Database (TMCDB) to address this problem. The TMCDB is built atop the Hibernate Object-Relational Mapping (ORM) framework, which, among its many advantages, enables the change of the underlying database implementation without requiring changes to the application code. ALMA supports both the *Oracle* and *hsqldb* databases. The CTA ACTL team has implemented support for MySQL as a prototype and applied it in the context of the MST prototype project at DESY and HUB. The ACS development group at ESO has incorporated this enhancement into the latest release of ACS (version 2015.2 at time of writing). Furthermore, there is even a branch of Hibernate that supports NoSQL databases such as MongoDB. Graphical tools for easy population and modification of the configurations are being evaluated (see Sec. 3.4.2).

The problem of gathering and storing data for the large expected number of monitoring points has been successfully resolved in the case of ALMA, with the help of a large and expensive Oracle implementation. However, the ability to query these data in any but the most trivial manner has proven much more difficult, in spite of many attempts to resolve it in ALMA. Both the ALMA and ACTL experts believe[6] that this is because relational databases are ill-suited to the manipulation and retrieval of large quantities of time-series data. Implementations based on NoSQL databases are currently being tested, Cassandra in the case of ALMA, and MongoDB by both ACTL and ALMA. The ACTL team is scheduling a rich test program to evaluate the performance of these technologies (see Sec. 3.4.2) in order to reduce the risk of choosing the wrong approach. Because the full complement of devices, and hence monitor points, will only be realized quite late in the construction of CTA, we plan to evaluate evolving technologies in this area, but nevertheless find a well-suited setup well in advance via detailed tests and simulations. In addition, CTA will profit from the ALMA effort, which since all of its 66 antennas and about 200,000 monitoring points have been delivered to the operations site, has a much more pressing schedule.

[6]ACTL team members have participated in meetings to exchange ideas and solutions with the ALMA experts for the common problem of handling the instrument monitoring data.

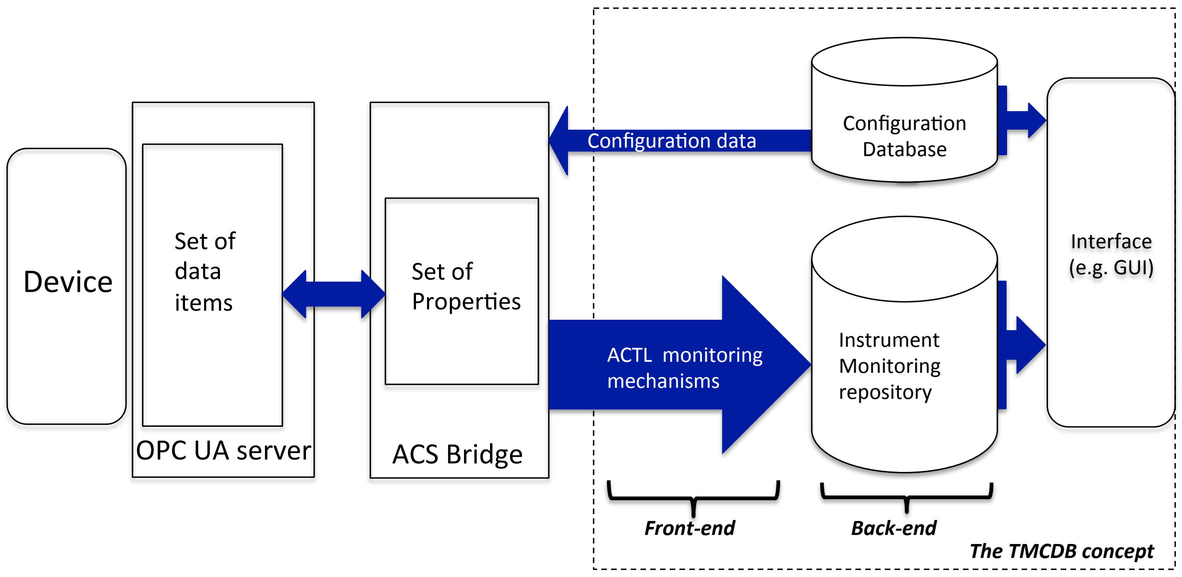


Figure 3.10 – Illustration of the TMCDB concept, which contains the configuration database and the instrument monitoring repository. The task of the front-end mechanism is to gather the monitoring data (instrument monitoring) or transmit the configuration data (obtained from the configuration database) through the ACS-OPC UA bridge to the device, and the task of the backend is to handle the communication with the database.

The functionality of the instrument monitoring can be divided in two items, the front-end and the back-end (see also Fig. 3.10):

Front-end. The front end extracts the data from the monitoring point and transmits it to the backend. The ACS framework provides the basic element *monitor* that permits the possibility to trigger by *time* or *value* change, or by both. The ACTL team has at hand two possible ACS based solutions for the front-end, the *Blobber-Collector* and the *Property Recorder*.

- **Blobber-Collector:** This is the baseline solution for the CTA monitoring front-end. In order to provide a scalable solution to the problem (raised in the ALMA installation) of gathering and storing data from about 200,000 monitor points from many telescopes, ALMA has developed the monitor collector and blobber mechanism in ACS. Each antenna (in the case of ALMA) or telescope (in the case of CTA) contains tens of sub-devices, each with numerous sensors. A single process, the monitor collector, operates on each telescope, collecting the monitor point data from each of these sensors over some specified period, typically one minute. Each monitor point can be sampled at a rate determined at deployment time; in general, rates above 2 Hz are rarely needed. It is also possible to specify that a monitor point be re-sampled only when its value changes by an amount that exceeds some threshold. Responsibility for acquiring the data from the collectors and storing them in the database is shared by a set of centrally located processes, the blobbers. Each blobber is assigned approximately 10 telescopes; in round-robin fashion, the blobber downloads the data acquired by each of the monitor collectors on the telescopes in its group, calculates basic statistics over the 1-minute period for each monitor point, organizes and formats the data into multiple records, and then executes a single transaction to finally store these records in the database. For ALMA, a system of 50-60 telescopes, this resulted in the very manageable total of about 100 transactions per minute to the Oracle database at the ALMA operations site, routinely storing 900,000 such time-tagged values every minute. The blobber design is quite flexible, allowing the replacement of the relational database with, for example, a NoSQL database (MongoDB has already been used successfully in tests at the ALMA site) or even text files.
- **Property Recorder:** The *Property Recorder* is a generic ACS property monitoring system, implemented with the ACS Python API, that exploits ACS standard mechanisms like *monitors*. It consists of a front-end, which automatically detects any ACS component and continuously acquires its property data and then transfers it to a database back-end. The *Property Recorder* is

designed to monitor data at fixed frequencies around a few Hz maximum, and it is also able to monitor data triggered by on-value-changes (relative or absolute changes). Both fixed rate and value-change triggers can be combined and specified for each individual ACS property separately. The front-end of the *Property Recorder* is pluggable to different back-ends to insert data into different database technologies. Currently, the back-end uses MongoDB. Alternative back-ends using MySQL and HDF5 files are planned. The main benefit of this tool is the automatic detection of the deployed setup and individual property configuration from the configuration database, including value-change triggered monitoring, and thus it is ready to be used in any early prototype and early installation on the CTA sites.

The current approach is to employ the *property recorder* for some of the telescope/hardware prototype setups and also in very early stages of the telescope deployment in the arrays due to its easier setup and to eventually replace it with the *blobber-collector*, which requires a somewhat more complicated deployment process, to provide the required performance and scaling process needed when several telescopes are deployed.

Back-end The back-end takes care of storing the gathered data into the CTA instrument monitoring repository. The *Tech0* data are stored in the on-site data centre within the instrument monitoring repository and constitute the sub-sample of the stored monitoring data that is relevant for the DATA work-package. For a proper selection of a back-end one needs to consider the structure and volume of the data to be stored, and the support for the usage of the stored data. Both the *Blobber-Collector* and the *Property Recorder* use different pluggable back-ends. Given the associated data rates and the lessons learned from the ALMA experience, back-ends based on non-relational database technologies are being considered for this task and preferred to SQL database technologies (MySQL, Oracle etc.). Currently, the MongoDB and Cassandra back-ends are being evaluated as the best candidates for this task (see Sec. 3.4.2 for details).

Detector quality assessment tools

These tools provide means for both automatic and on-demand assessment of the performance of telescopes and other devices. In order to allow for on-demand requests, a user friendly web-based GUI tool will be provided for the CTA technical staff. The GUI will provide access to the monitoring data stored in a data base and plotting functionality to display:

- any parameter as a time series, which can be zoomed, filtered, cut
- correlation between parameters in the database.

The input data for these tools will be time-averaged instrument monitoring data (for example daily statistics as average, minimum, maximum, root mean square values for each of the monitoring points), extracted from the much larger monitoring repository (see Sec. 3.2.2). The time-averaged data will allow for easier correlation studies, either automatic or on-demand. It is important to note that in case a more detailed study is required, e.g., when a suspicious behaviour of a device is found by the automatic routines, the fine time-binned instrument monitoring repository will be used. In a similar way as with the instrument monitoring repository, MongoDB and Cassandra database technologies are being investigated for this task. First prototyping activities have started with data collected by the CTA MST prototype and by MAGIC.

Other repositories

The ACTL-OPS is responsible for providing support for further ACTL repositories in addition to those described above. The need for these additional repositories is being investigated by the ACTL team. In particular, the need for a repository for storing instrument calibration and engineering data not directly associated with the Cherenkov data has been identified. This calibration and engineering data can be for example CCD images for instrument pointing refinement and telescope bending model determination and metrology measurements from accelerometers and inclinometers for the study of the telescope structure. The ACTL team is working to produce the specifications for such a repository.

Finally, the design for the temporary Cherenkov data on-site repository, is formally included as one of the ACTL-OPS sub-products. This repository constitutes the main usage of the storage system in the on-site data centre (see Sec. 3.2.5). Being directly affected by the DAQ architecture (see Sec. 3.2.3) and the needs of the DATA group, the ongoing work is coordinated with these groups. Further work on this repository is still required to integrate all the requirements and in particular further effort around prototypes and tests, as mentioned in Sec. 3.4.8, is needed.

3.2.3 ACTL-DAQ: Data acquisition software

In this section the product ACTL-DAQ (Data acquisition Software), numbered *3.3* in the CTA PBS structure, is described.

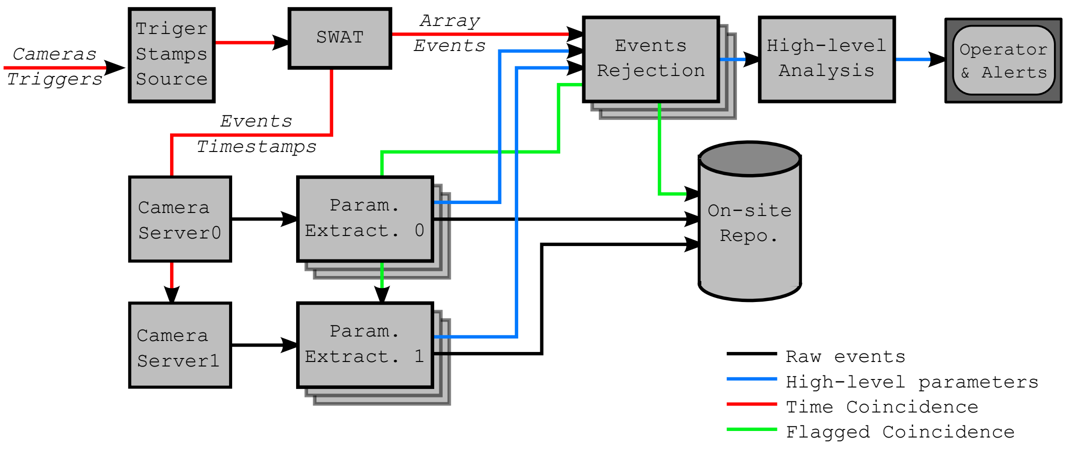


Figure 3.11 – Foreseen real-time data reduction configuration. Single-camera triggers are time-stamped before being passed to the software array trigger (SWAT). Timestamps passing the SWAT criteria are sent to the camera servers which in turn send the associated event data to the parameter extraction nodes. There the data are calibrated, a first analysis performed and image parameters extracted. From there, event parameters are delivered to the event rejection module (blue lines) where they are assembled into array-events and further filtered. The result of this analysis is fed back to the parameter extraction nodes (green lines) and forward to the high-level analysis. Events that remain of interest are then compressed and written to the repository (black lines), while the result of the high-level analysis is presented to the operator. The parameter extraction nodes must provide enough buffering capabilities to wait until the analysis is complete before deciding whether to discard an event or not. All boxes and connecting lines in this diagram are logical entities. More than one box can run on a single physical server, and a given box can be load-distributed across several servers. The lines are also logical connections, which will be implemented via the ZeroMQ middleware for the data transfer and ACS remote calls for the configuration. In order to improve fault tolerance, information from the event rejection could be routed in parallel to an algorithm that detects and corrects faults in the information used to associate events. The results of that algorithm would then be provided to the off-line analysis.

Design Overview

Due to the Cherenkov data volume of the order of several tens of PB per year, a sophisticated data acquisition (DAQ) pipeline as illustrated on Fig. 3.11 is required. This architecture will be deployed on top of the ACS framework to allow for an efficient transfer of the bulk data. The DAQ design is created as a modular architecture to allow heterogeneous pipelines to coexist. A modular architecture has several advantages compared to a monolithic approach. Complex processing can be decomposed into simple tasks, thus easing the development and validation procedures. It allows for easier load-balancing across several physical nodes and helps to achieve the high availability required by allowing the reconfiguration of the system at any time. Besides Cherenkov observations, CTA is foreseen to also operate in optical interferometry mode. This extra feature would continuously record the observed value of some pixels, sampled at 100 MHz or more. The DAQ system is thus designed to be modular enough to allow for such reconfiguration of the acquisition pipeline on a short timescale. The DAQ will be composed of components running inside ACS and connected via ZeroMQ data streams. Each component has a simple task to perform while more complex tasks are created by plugging several components together. Control and monitoring will be made through ACS while specific master components are responsible for monitoring the system and taking appropriate actions if necessary. A short description of foreseen components is given below while further details about the architecture are provided in [16].

- *Data Buffer* components will store events and/or event parameters in memory for some time until a decision has been taken regarding their fate. They will provide flexibility in the pipeline by allowing the array event builders to retrieve event parameters only when the time coincidence is known. They will also help the self-assembling data model to operate with fewer time-constraints.

- *Repository Writer* components will be responsible for committing incoming data to the on-site repository. If the data are provided raw then it will be compressed on-the-fly. Each repository writer will deal with one single stream of data, opening new files once the size or event number limit is reached or after a new run is started. The target storage node will remain the same at least during a single run. Changing the storage node from run to run would allow for level-B analysis to start while data taking is still ongoing, possibly providing more accurate results earlier.

- *Processing Nodes* are meant to be all-purpose skeleton components so that any kind of computation can easily be introduced to the pipeline. Their primary goal is to embed real-time processing into the DAQ pipeline. For instance, extracting the image parameters from the raw event data are desirable early in the pipeline to help reduce the total network throughput. The data reduction algorithms are provided by DATA.

- *Array Event Builder* will operate either on raw event data or parameters only. It will retrieve individual camera data from *data buffers* or raw data files and merge them into array-events. The main goal of this component is to deliver complete array events to the analysis pipelines.

- *Time Coincidence Splitter* will generate sub-array specific trigger data from full array triggers.

- *DAQ Master* will monitor ongoing acquisitions and is responsible for automatic error detection and recovery. This component will be introduced only after the other components have been commissioned to improve the array reliability.

Data Volume Reduction Scenarios

The general concept for camera data capture involves transmission from each camera to a camera server process running at a central location. In most cases a stream of all available raw data (i.e. locally triggered events) can be continuously delivered, without any event pre-selection (or array-level triggering). This approach minimises the complexity on the camera side and delivers maximum flexibility for data collection. Sub-system-level hardware triggers are envisaged internally by the camera groups of the LST and SCT. Under these assumptions, the volume of data delivered by cameras is likely too high for complete permanent storage. Estimates suggest a data volume of several tens of PB per year. Reducing this large data set should not discard scientifically useful data, thus each new data trimming process should be carefully evaluated and tested off-line before it is introduced in the real-time pipeline. This can be achieved by the proposed modular architecture by storing the full dataset during commissioning and gradually introducing new data trimming algorithms in the real-time pipeline once they are ready.

Various data reduction strategies are under consideration, including obvious choices such as waveform reduction and zero-suppression, but also more ambitious ones such as background suppression. This latter item aims at discarding proton or helium nuclei initiated cascades as they are of no interest for science. Tagging such events in a reliable way is a computation intensive process which will be challenging to implement in real-time. Further details on these approaches, as well as the implication on the Data pipelines and calibration are provided in Ref. [17].

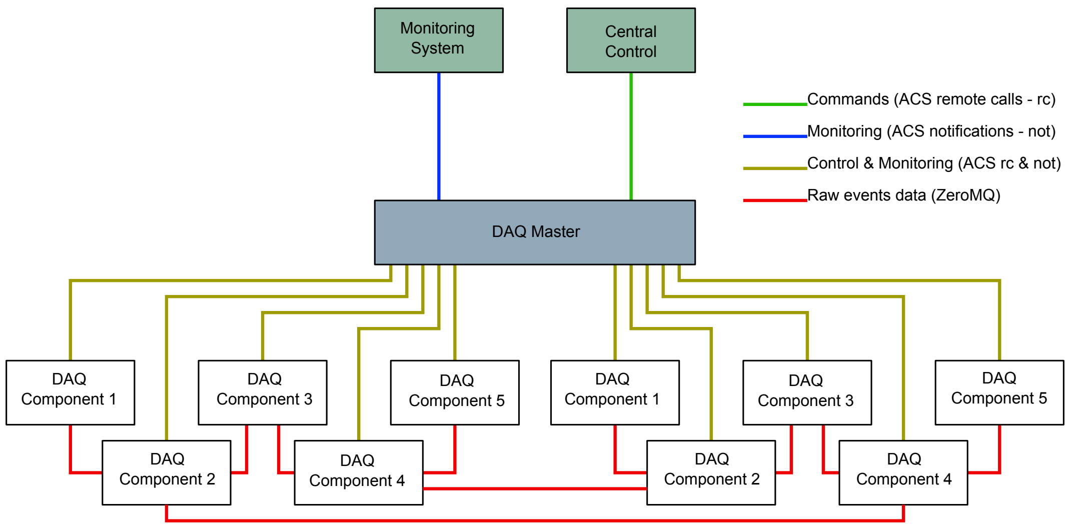


Figure 3.12 – Conceptual view of the DAQ data flow with an arbitrary low-level component inter-connection. Commands are provided by the central control, directly to the DAQ master. The master then takes appropriate action to reconfigure the relevant components accordingly, for instance by connecting two low-level components with a raw event stream. Each low-level component is monitored by the DAQ master which in turn forwards summary information to the monitoring system. If necessary, for instance for debugging purposes, it is possible that the monitoring system directly connects to the low-level component notification channels. The vertical data flow is entirely handled by ACS (either through remote procedure calls or notification channels) while the horizontal, high-throughput data flow is entirely handled by ZeroMQ streams. This architecture reduces the complexity for the central control, and lets the DAQ master handle errors.

Data Streams and format

All control and monitoring will be implemented using native ACS, while the bulk data transfer will be implemented using ZeroMQ as shown in Fig. 3.12. ZeroMQ implements many interesting features such as automatic connect/reconnect of new peers, automatic load-balancing, message encapsulation and several connection paradigms and media. Because zeroMQ is also capable of connecting different processes or threads running on a single machine, it would be very easy to build single large components from several smaller ones if the communication performance requirements are not met by network protocols. Connection paradigms will be chosen individually for each connection item. The possible cases are request-reply, push-pull and publish-subscribe.

The internal DAQ format will rely on protocol buffers. Protocol buffers are a standardised way of handling the serialization problem. Developed by Google and widely used within their facilities, structures are defined via a high-level language similar to an interface description language (IDL). A compiler then generates the run-time code. The serialized version of the structure is highly optimized, with values being packed as efficiently as possible to reduce the network throughput or left untouched to minimize the serialization overload. Moreover, the built-in forward and backward compatibility of the format allows modification of the structure content without breaking the pipeline. As an option that can be selected, the built-in reflection of the structures allows reading and writing any data structure to/from persistent storage without having to write a single line of code specific to any data structure.

The latest DAQ prototype has been reviewed by the DATA management sub-work package Pipelines to evaluate whether the proposed framework can be re-used for both the real-time and off-line analysis pipelines. Following this evaluation the decision was taken to merge both the DAQ and RTA prototypes while the offline analysis will remain separate and most likely implemented in python. Moving algorithms from one pipeline to the other will be possible though via a simple interface built on top of zeroMQ and protocol buffers.

3.2.4 ACTL-TRIG: Array Trigger and Clock Distribution

In this section the product ACTL-TRIG (Array Trigger and Clock Distribution, PBS 3.4) is described. ACTL-TRIG provides two main products, which are the Clock Distribution & Trigger Timestamping (PBS

3.4.2) and the software array trigger (PBS 3.4.1).

Clock Distribution & Trigger Timestamping

Data from a wide variety of sources (e.g. telescope cameras, weather stations) distributed throughout the CTA array must be correlated in time. Successful association of these data requires sufficiently precise timestamping mechanisms. For quantities such as weather data that need only be known on time scales larger than typical network latency times ($\approx$ milliseconds, see B-ACTL-2330), adequate timestamps can be obtained by timestamping after the data are read out, using system clocks synchronized with the Network Time Protocol (NTP). For these cases, round-trip times in the array (typically $10\,\mu\mathrm{s}$ for a distance of $2\,\mathrm{km}$ at fibre speed) are negligible.

The requirements are far more stringent for the Cherenkov event data read from the individual telescope cameras. Here relative telescope-to-telescope timing accuracy of the order of $10\,\mathrm{ns}$ is required in order to correctly associate events which are seen by several telescopes in coincidence and to efficiently reject non-coincident events. If Cherenkov wavefront-timing is used for direction-fitting and background rejection then relative timing accuracy of $2\,\mathrm{ns}$ is desirable. Therefore, a goal requirement of $2\,\mathrm{ns}$ for the relative timing accuracy (A-PERF-0610) has been set, with a requirement that the rms precision of the individual telescope trigger times be $1\,\mathrm{ns}$ (B-ACTL-0270). Studies of pulsars, active galactic nuclei (AGN), and other periodic and transient phenomena also require that an absolute time be attached to each reconstructed photon candidate. This is most easily done via the same mechanism that provides the timestamps to the telescopes, and imposes an additional requirement that these timestamps provide an absolute time good to $1\,\mu\mathrm{s}$ (B-ACTL-4010).

These more stringent requirements mean that a separate solution must be found to distribute a clock and timestamp information to the telescope cameras. This clock distribution and trigger timestamping (CDTS) system will:

1. Receive a trigger signal from a device and timestamp it with nanosecond precision relative to all other relevant hardware components in the array (e.g. the other CTA cameras).
2. Forward trigger timestamp information, together with additional information about the nature of the trigger and the status of the telescope, to the software array trigger (SWAT).
3. Deliver precision clock signals to all devices (e.g. the Cherenkov cameras) in the array as needed. Clock signals will consist of both a $1\,\mathrm{PPS}$ signal and a configurable signal $> 10\,\mathrm{MHz}$ (the latter value may vary between component designs).
4. Provide an absolute timestamp good to $1\,\mu\mathrm{s}$.
5. Fulfill additional service and time calibration functionalities.

The backbone of the CDTS system is a White Rabbit network [18, 19]. The White Rabbit transmission protocol is based on the BASE1000-BX10 Ethernet standard (see IEEE 802.3-2008) and uses single-mode simplex fibres with different wavelengths for data transmission in each direction ($1310\,\mathrm{nm}$ and $1490\,\mathrm{nm}$). The network itself consists of a single master switch that provides the time and frequency reference for the entire network, connected to a set of slave switches as shown in Fig. 3.13. For the CTA application, the time and frequency references provided by the master switch are disciplined by a single external GPS clock which provides the absolute timing information. A standardised interface card built on White Rabbit technology will provide the interface between all telescope cameras and the CDTS. This card is the responsibility of ACTL and will be provided to the telescopes.

White Rabbit: White Rabbit technology provides nanosecond accuracy and sub-nanosecond precision of clock synchronization for large distributed systems. Its aim is to combine the real-time performance of industrial networks and the flexibility of the Ethernet protocol with the accuracy of dedicated timing systems. White Rabbit is capable of synchronizing more than 1000 nodes, connected by fibres of up to $10\,\mathrm{km}$ in length, with sub-nanosecond accuracy. It uses a robust, prioritized message delivery system. White Rabbit time synchronization packets are routinely flagged as high priority, but other critical Ethernet packets may be priority-flagged as well.

This technology has a number of advantages. First, it provides a scalable and modular platform with simple configuration and low maintenance requirements. Second, because the White Rabbit Project is open source rather than proprietary, use of this technology also avoids hardware and software vendor lock-in. This is reinforced by increasing adoption of this technology in recent years. White Rabbit is currently being implemented as the next version of the IEEE 1588 standard, the Precision Time Protocol (PTP) and there is a large and active community of White Rabbit users and contributors (for example CERN, GSI, DESY and Nikhef)[7]. Several other modern experiments (LHAASO,km3Net, IC-Gen2) have also decided on the use of hardware based on the common and open White Rabbit solution.

Non-White Rabbit devices may be connected to the White Rabbit network via a standard network interface card and operated within the White Rabbit network. White Rabbit is therefore compatible with modern computing centre strategies (such as NAGIOS) for monitoring the status of complex systems, making it easy to use the same technology to monitor both White Rabbit and non-White Rabbit CTA devices. White Rabbit's reliability has been assessed both in the laboratory and the field [18, 20, 21, 22, 23], including a field application very similar to that proposed for CTA [24, 25, 26]. Further discussion of both component reliability and the reliability of the overall timing distribution system is given in section 4.3.

Fibre Network and Network Components: As noted earlier, the timing distribution system must be supported by a dedicated White Rabbit optical fibre network (simplex BASE1000-BX10). The fibre network is part of the overall fibre network planned and deployed by ACTL-ONSITE. A star-like architecture is required with a single White Rabbit slave switch in the ACTL centre directly connecting via individual fibres to UCTS boards in up to 17 telescopes. It is planned that these fibres be co-located with those used for other purposes. Given knowledge of White Rabbit and tested small form-factor pluggable transceiver (SFP) capabilities, fibre lengths of up to 10 km can be used without affecting performance.

All fibres inside a cable to a given telescope used for the White Rabbit system must meet the following requirements: they must be roughly equal in length, not contain any active components, and be of identical type and optical properties (this includes any short patch cables used in the data centre). It is optimal if the fibres to a given telescope come not only from the same producer but the same production batch. These rules (which are more stringent than the requirements for normal Ethernet connections) must be followed anywhere in CTA where White Rabbit nodes or switches may be installed.

A two-layer White Rabbit switch network is required, with six White Rabbit switches plus a spare. The number of SFP connector pairs is equal to the number of telescopes plus the number of switches. The CDTS needs 1$(+1$ spare) CTA fibers (i.e. 2$(+2)$ physical fibers) per telescope. Two long (multi-kilometer-length) fibres from different production batches are also needed for lab testing pre-commissioning. A White Rabbit-capable optical fibre from the pool of available free optical fibres will be used for calibration and debugging purposes during commissioning.

UCTS: A unified clock distribution and trigger timestamping (UCTS) board, built upon White Rabbit technology, will provide a common, dedicated interface between the CDTS system and each of the telescope cameras. A baseline realization of the UCTS board has been developed on the basis of the White Rabbit simple PCIE FMC carrier (SPEC) card, equipped with a 5-channel digital I/O FMC card [24, 25, 26]. It has been tested long-term in a laboratory and is part of the HiSCORE prototype array, reconstructing air shower events with sub nanosecond precision, see Sec. 3.4.4. The details and precise implementation of the UCTS interface are currently under negotiation between the various camera groups and ACTL and will be detailed in a forthcoming ICD.

GPS Clock: The GPS clock disciplining the master switch should consist of a highly stable oscillator (e.g. a double-ovenized or rubidium oscillator) disciplined by a GPS receiver. We have considered the need for one spare and a reference clock that could operate in parallel with the master clock for error-checking purposes.

CDTS Server: A dedicated computer server monitors all UCTS boards and White Rabbit switches, as well as the GPS-clock. The CDTS server tracks performance statistics (packet rate, dropped packets, etc.) and hardware monitoring information from the switches, UCTS boards, and GPS clock. It reports the system status to the instrument monitoring system (ACTL-OPS).

[7]`http://www.ohwr.org/projects/white-rabbit/wiki/WRUsers`

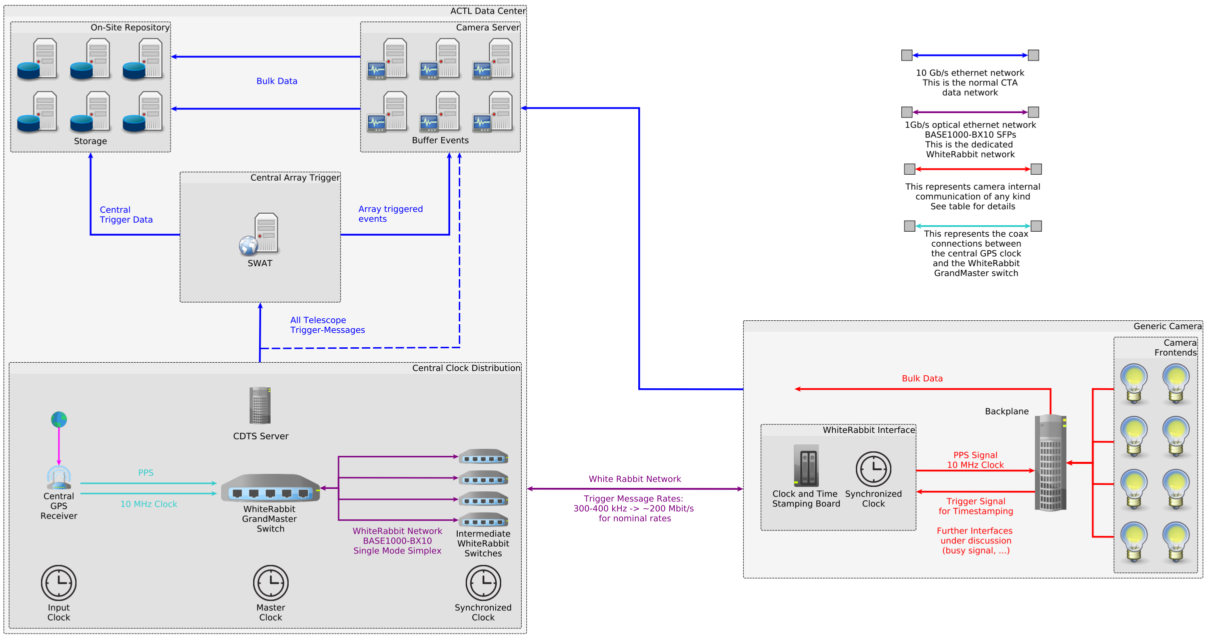

ACTL Data Center
On-Site Repository
Storage
Camera Server
Buffer Events
Bulk Data
Central Array Trigger
SWAT
Central Trigger Data
Array triggered events
All Telescope Trigger-Messages
Central Clock Distribution
CDTS Server
Central GPS Receiver
PPS
10 MHz Clock
WhiteRabbit GrandMaster Switch
WhiteRabbit Network BASE1000-BX10 Single Mode Simplex
Intermediate WhiteRabbit Switches
Input Clock
Master Clock
Synchronized Clock
10 Gb/s ethernet network This is the normal CTA data network
1Gb/s optical ethernet network BASE1000-BX10 SFPs This is the dedicated WhiteRabbit network
This represents camera internal communication of any kind See table for details
This represents the coax connections between the central GPS clock and the WhiteRabbit GrandMaster switch
White Rabbit Network
Trigger Message Rates: 300-400 kHz -> ~200 Mbit/s for nominal rates
Generic Camera
Camera Frontends
Bulk Data
Backplane
WhiteRabbit Interface
Clock and Time Stamping Board
Synchronized Clock
PPS Signal 10 MHz Clock
Trigger Signal for Timestamping
Further Interfaces under discussion (busy signal, ...)

Figure 3.13 – Scheme of the proposed White Rabbit optical fibre network and data flow for the CDTS system showing the common UCTS board in a generic Cherenkov camera.

Timestamp information flow: The baseline scheme under consideration is sketched in Fig. 3.13, which shows the flow of data between the cameras, camera servers, UCTS, White Rabbit switches, and SWAT over both the White Rabbit and normal Ethernet networks. Triggers are sent from the camera to the UCTS, which sends on event trigger timestamps and associated auxiliary information (e.g. trigger type) via the White Rabbit network to the ACTL data centre. From there a dedicated server (the CDTS-Server) sends on these timestamp packets to both the SWAT (solid blue line in Fig. 3.13) and the camera server (dashed blue line). This parallel forwarding has advantages from the point of view of latency (timestamp information reaches the camera servers sooner, permitting early merging). It also ensures that all events, regardless of their trigger type (e.g. array trigger events, muon or calibration events), are timestamped in the same fashion.

The SWAT uses the timestamps to determine whether an array event has taken place and distributes the appropriate timestamp packet to each camera server as part of an instruction to the ccamera to transfer the event farther along the data acquisition pipeline. In addition to events selected on the basis of timestamp coincidence (array trigger events), SWAT can pass through single-telescope events on the basis of its trigger type (e.g. muon or calibration events).

Array Trigger

An array trigger that preferentially selects shower signatures will be relied upon to stabilize data read-out rates, reduce data volume, and recognize background processes (e.g. muon rings and obvious hadrons). The array trigger will also provide redundant tags (e.g. an event number) to individual components of an event.

Trigger Concept Overview The cameras in the CTA array will generally trigger locally (i.e. independently of other telescopes) with an algorithm that suppresses night-sky background light. In the case of some cameras (i.e. the LST projects), information from neighbouring telescopes may also be incorporated into the camera trigger. The array trigger acts on information sent by the telescope triggers to select shower signatures and distinguish background processes (obvious hadrons and muons). The full trigger system, including the array trigger, must be flexible enough to preserve background processes such as muon rings that are useful for calibration while keeping the overall volume of recorded data under control.

A stereoscopic time coincidence imposed by a central real-time hardware array trigger (such as that used by H.E.S.S., MAGIC, and VERITAS) would place additional demands on the camera electronics. In particular, given that we expect relatively large round-trip signal propagation times (≈ 10 microseconds), a central array-level hardware trigger with a coincidence window for telescope triggers demands sampling electronics with deep ($\gg 4$ microsecond) buffers. Since it has been demonstrated by Monte Carlo studies that imposing the stereoscopy requirement only reduces the array trigger rate by a factor of a few, it was felt that the demands imposed by a hardware array trigger were not well-justified.

However, as noted in Sec. 2.2.1, steps must be taken to reduce the overall volume of data stored. This reduction can be achieved at the level of an individual telescope by compressing or otherwise restricting the volume and type of data recorded. Examples include muon ring identification, the suppression of empty pixels (zero-suppression), and the extraction of pulse shapes. This can be done by the data acquisition system using the appropriate computing hardware, or it can be done at the trigger level, where special care has to be taken for pass-throughs of events of interest for calibration. The factor of few reduction gained by imposing stereoscopy can provide greater flexibility by permitting these constraints on the volume of data recorded by the individual telescopes (e.g. zero suppression) to be relaxed. This is expected to be particularly advantageous during the commissioning period. Therefore, an alternative, software-based concept has been adopted by CTA.

Software Array Trigger The Software Array Trigger (SWAT) is a program running on a dedicated, central processor that works in the near real-time regime. The SWAT receives input record streams from each telescope. The input record includes a trigger timestamp plus other identifying information (at a minimum, the type of the trigger and the status of the telescope at the time the trigger was produced; the baseline timestamp-distribution scheme also requires additional counters to be included). The SWAT then collects all the input records from all the telescopes and merges them in a common buffer. At this point the array coincidence algorithm works as follows:

1. Records are grouped by sub-array.
2. Information about the telescope pointing at the time of the trigger is used to correct the timestamps for differences in the shower arrival time between the telescopes.
3. Records are sorted by time and coincidences searched for on the basis of the corrected timestamps.
4. A unique identifier is assigned to each detected coincidence.
5. The coincidence window is programmable individually for each telescope pair. In general, the size of the coincidence window will be $1/c$ times the distance between the two telescopes along the axis perpendicular to the detection plane, ± 20 nanoseconds. The maximum size of the coincidence window (assuming a maximum of 1 km distance between each telescope pair) is 3.3 microseconds.
6. A packet of information is then sent back to each camera server that includes this identifier, plus all the information included in the input record. This information is also delivered to ACTL-DAQ, which uses it to verify the completeness of data sent to the archive.

There is also a mechanism to pass through single-telescope calibration events of various types (including muon rings). All triggers sent to the SWAT must be tagged by the cameras with a trigger type that clearly identifies them as either a physics trigger or a calibration trigger (e.g. muon ring, injected calibration event). For any appropriately-designated calibration event type, SWAT will pass through (and if necessary pre-scale) single telescope events.

The SWAT algorithm operates on a timescale of order 1 second, which gives ample time for any array trigger detection algorithms and compensates for any network induced packet jitters (common during high volume network traffic). This requires that the camera data (whether continuously digitized or read out after a camera trigger) be kept in a secondary buffer on the camera server for ~ 1 second. When a camera server receives notification from the SWAT that an array coincidence has been detected, it uses the information provided by the SWAT to access and permanently record the relevant camera bulk data.

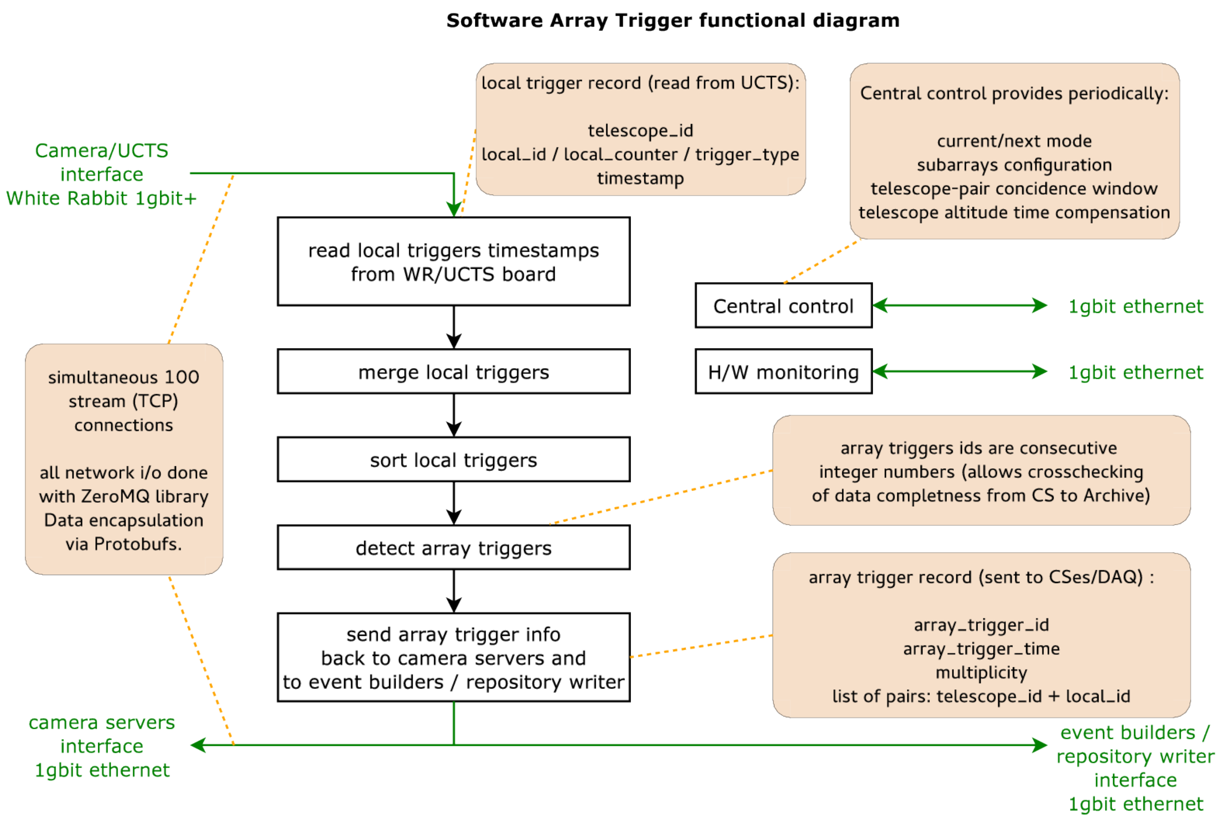

Software Array Trigger functional diagram
Camera/UCTS interface White Rabbit 1gbit+
local trigger record (read from UCTS): telescope_id local_id / local_counter / trigger_type timestamp
Central control provides periodically: current/next mode subarrays configuration telescope-pair concidence window telescope altitude time compensation
read local triggers timestamps from WR/UCTS board
merge local triggers
sort local triggers
detect array triggers
send array trigger info back to camera servers and to event builders / repository writer
Central control
H/W monitoring
1gbit ethernet
1gbit ethernet
simultaneous 100 stream (TCP) connections
all network i/o done with ZeroMQ library Data encapsulation via Protobufs.
array triggers ids are consecutive integer numbers (allows crosschecking of data completness from CS to Archive)
array trigger record (sent to CSes/DAQ) : array_trigger_id array_trigger_time multiplicity list of pairs: telescope_id + local_id
camera servers interface 1gbit ethernet
event builders / repository writer interface 1gbit ethernet

Figure 3.14 – SWAT functional diagram.

Demonstrator systems based on this concept show promising performance [27]. Figure 3.14 shows the functional diagram of the proposed system.

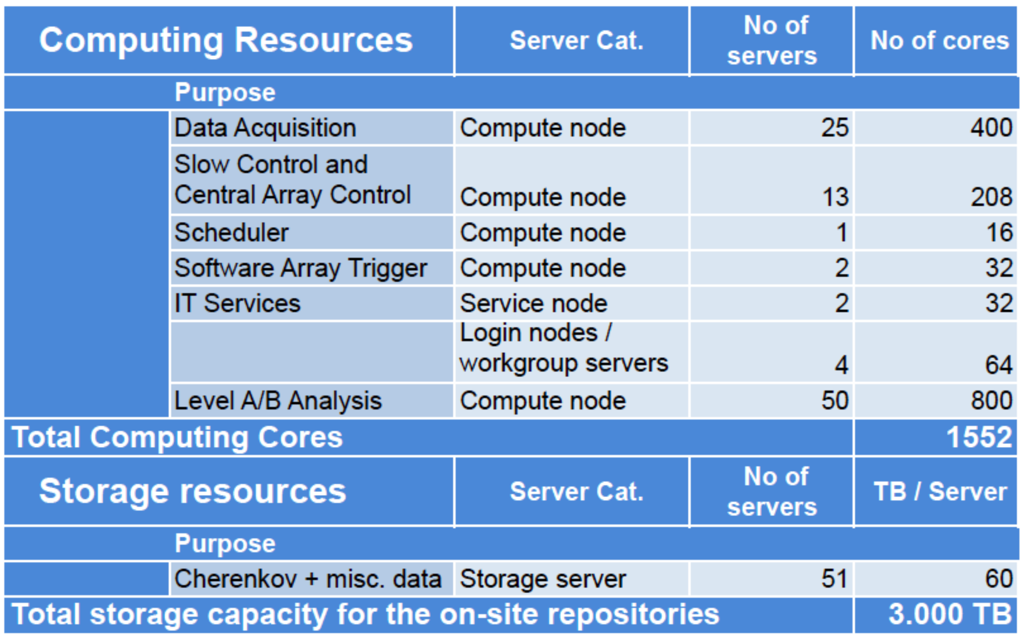
3.2.5 ACTL-ONSITE: On-site ICT Infrastructure

In this section the product ACTL-ONSITE (On-site ICT Infrastructure), numbered *3.5* in the CTA PBS structure, is described. ACTL-ONSITE comprises all computing/storage hardware, the overall networking infrastructure (including cabling and switches) and all system services (operating system handling, networking services, name services, etc.) needed to drive on-site ICT. In addition to the southern and northern on-site data centres, the remote array control centre is handled by ACTL-ONSITE as well.

On-Site Computing and Storage Servers

The design of the on-site computing cluster, including the hardware and software specifications of the individual computing nodes and storage servers, is covered here. CTA on-site computing is characterized by high data rates (high bandwidth networks, high throughput computing) and a CPU intensive data processing and analysis, which are accompanied by a high availability required for the entire system. The on-site data centre will host different classes of servers and nodes for computing and storage, according to different functionalities divided into the following components:

- Camera servers (under responsibility of the telescope projects)
- Computing nodes
- Storage servers
- Service nodes



Computing Resources		Server Cat.	No of servers	No of cores
	Purpose			
	Data Acquisition	Compute node	25	400
	Slow Control and Central Array Control	Compute node	13	208
	Scheduler	Compute node	1	16
	Software Array Trigger	Compute node	2	32
	IT Services	Service node	2	32
		Login nodes / workgroup servers	4	64
	Level A/B Analysis	Compute node	50	800
Total Computing Cores				1552
Storage resources		Server Cat.	No of servers	TB / Server
	Purpose			
	Cherenkov + misc. data	Storage server	51	60
Total storage capacity for the on-site repositories				3.000 TB

Table 3.1 – Summary of the ACTL storage and computing farm that will be located in the ACTL on-site computer centre. All numbers are approximate and refer to the southern CTA site only.

- Login nodes / workgroup servers
- Control room PCs.

Except for the control room PCs and, possibly, the camera servers, all components will consist of commodity compute servers in standard 19-inch racks. They are to be installed in the on-site central computer room. The camera servers, one per telescope (so about 100 for the CTA southern array), are in charge of the Cherenkov camera readout and a first selection of the raw data by using the array trigger signal and are provided by the telescope groups. The main components of the ACTL computing cluster are the computing nodes for data acquisition, event processing and running the ACTL software. Other relevant classes of servers are storage head nodes and servers, used to save and manage the payload data (triggered and compressed data), login nodes, workgroup servers and service nodes. All these different classes (excluding the camera servers) will be implemented by ACTL-ONSITE with a minimal set of standardised hardware configurations. The servers and computing nodes will be based on two-socket x86 CPU architecture. CPU performance, memory and local disk size are adapted to the needs required by the functions of the different server types (see below). Due to the fact that all servers are following the same basic architecture the choice of the specific hardware and the procurement procedures including initial testing are common for all components and ensure easy spares management.

On-Site Computing The computing nodes are used for data acquisition (ACTL-DAQ) and data analysis (DATA management work package, level-A and level-B analysis) as well as for running all ACTL software products responsible for the control and monitoring of the telescope array. A total need of about 1500 cores of the x86 architecture for the CTA South array is estimated from current software prototypes (see Tab. 3.1 and Sec. C).

On-Site Storage The on-site storage system will be entirely assembled from basic storage units consisting of one head node server equipped with internally connected disks. Currently the space of a single (Nearline SAS) disk is 6TB and the number of disks connected to a head node is 12. Therefore, using RAID 6 for best redundancy a basic storage system today will provide a net capacity of 60TB. The logical design for the storage is based on the requirements of the camera readout and the data processing. The approach using flexible storage units as described above makes it possible to decide which kind

Number of lines	Purpose	Rate per line (Gbit/s)
2–4	Cherenkov Camera Data	10
1	Telescope Drive System	1
1	Array-level Trigger	1
1	AUX Devices/Network	10
1	Slow control	1
1	Management Network	1
5–3	Spares	10

Table 3.2 – Summary of the ethernet lines connecting a telescope with the control building and the array-level trigger. It is assumed that some telescope types (e.g. SSTs) will use fewer than four 10 Gbit/s-lines for the transfer of camera data. Note that the number of spare lines depends on the number of lines used for transfer of Cherenkov data for a specific telescope type.

of storage systems finally will be implemented, a hierarchical storage model, flat data distribution, fast cluster file systems etc. Based on these basic principles, the requirements for the density, amount, reliability, availability of the storage and required speed (intra- and inter-machines) of all required storages types will be specified. Specification of the needed lifetime, the replacement at the end of lifetime and the accompanying feature evolution has to be taken into account as well as the portability of part of the storage if necessary. A total need of about 3 PB are estimated for the CTA South array (see Tab. 3.1 and Sec. 2.2.1).

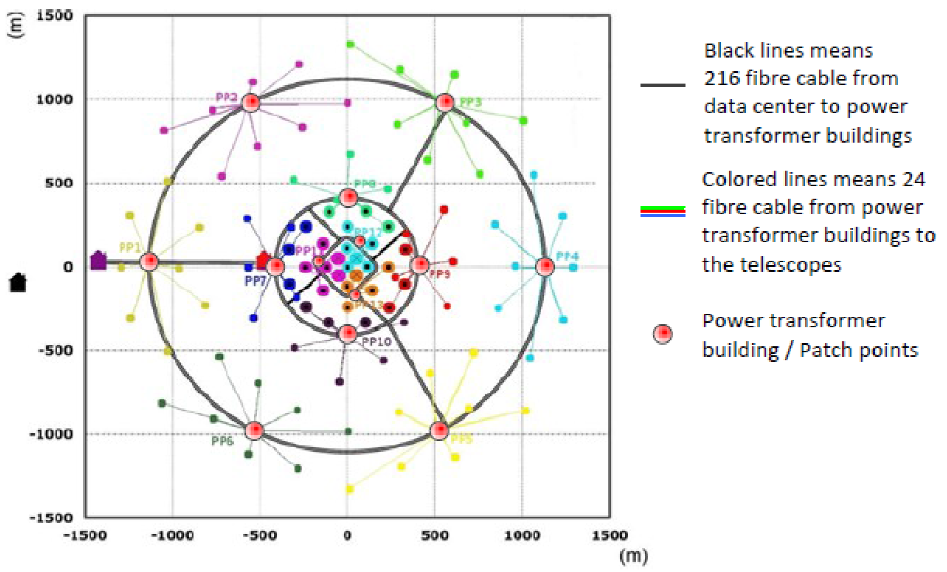
Network Infrastructure The on-site network infrastructure is composed of three main parts, the array network topology, the data centre network topology and the office/accommodation network topology.

The array topology includes the cable paths from the data centre to the telescopes and the appropriate interface connection plugs in the telescopes, the data centre and power transformer buildings.

As illustrated in Fig. 3.15, eight to twenty telescopes form a group that is connected to one patch point or power transformer building on the array. Each telescope is connected via a fibre cable holding 24 single-mode optical fibres, which allows for 12 single connections (two fibres are used for one connection in RX/TX mode) of up to 10 Gbit/s. In Figs. 3.16 and 3.17, the network topology at the telescopes, their direct connections to the data centre and the data centre network topology are shown. The following direct connections between the telescopes and the data centre are planned and recommended: two connections for the camera bulk data (which will be extended to up to four connections if the expected data rate requires it), one for the drive system, one for the trigger and timing distribution system, one for auxiliary devices (via a switch in the telescope), one for the management network, one for slow control and one connection for the wireless backbone as shown in Tab. 3.2. All connections could be used at 10 Gbit/s if necessary. This leaves up to five connections (three in the worst case) for redundancy, upgrades and/or temporary usage (e.g. during the commissioning phase).

The number and specifications of the connections between the data centre and the telescopes regarding drive system, trigger system, auxiliary devices, management network and slow control will be the same for the SSTs, MSTs and LSTs. However, the trigger and timing distribution system's need for White-Rabbit-compatibility does place additional constraints on the choice of fibre cables, especially in terms of uniformity of the fibre types (see Sec. 3.2.4 for further details). According to the different data rates the camera bulk data connections will be also different. For the NectarCam cameras on the LSTs and the MSTs four optical fibre connections with 10 Gbit/s will be installed. The MSTs' FlashCam will be connected with two optical fibre connections, each with capacity 10 Gbit/s. Also, the SSTs will be connected with two optical fibre connections, but with 1 Gbit/s only. Because of availability in general it is important to connect the telescopes with a redundant optical fibre. There will be only one type of optical fibre on site. That means that the optical fibre specifications will be the same for all connections. The type of the fibre will be ITU-T G.652.D. The difference in the data rate is a result of the different types of SFPs. It is planned to use 10 Gbit/s SFPs and 1 Gbit/s SFPs.

By means of a splice sleeve in the patch points the 24-fibre cables of each telescope are connected to a 216 fibre cable which connects the power transformer building with the data centre (red house



(m)
1500
1000
500
0
-500
-1000
-1500
-1500
-1000
-500
0
500
1000
1500
(m)
PP1
PP2
PP3
PP4
PP5
PP6
PP7
PP8
PP9
PP10
PP11
PP12
PP13
Black lines means 216 fibre cable from data center to power transformer buildings
Colored lines means 24 fibre cable from power transformer buildings to the telescopes
Power transformer building / Patch points

Figure 3.15 – Schematic view of a possible array network topology.

in Fig. 3.15). In the telescopes as well as in the data centre the optical fibre ends at a patch panel. The patch panels in the telescope are directly connected to network endpoints, which allows for direct connections of e.g. the camera to the camera server or the 24-port switch at the telescope, which is collecting the data from all auxiliary devices and the wireless LAN access point, to a point in the data centre. A detailed view is shown in Fig. 3.17.

The data centre topology includes the network wiring in the computer room and the network connections to the control room. In the data centre, a central switch stack is responsible for the data distribution from the patch panels (telescopes) and camera servers to the computing and storage nodes.

The office and accommodation topology includes all connections from the data centre to the offices and to the accommodations. If the offices and the data centre are in the same building, they will be directly connected. The accommodation or other buildings will have a small technical room with a switch, which is connected to the data centre and which will supply buildings with Ethernet connections.

The connection between the camera internal network inside the telescopes and the Camera Server in the on-site central computing centre is based on optical fibre single mode technology. Physically the internal switches inside the telescopes and the Camera Servers are connected via locally installed patch panels (PPTs, PPDCs) which are installed at all telescopes and the central computer centre as well. The PPTs and the PPD are also the endpoints of the fibres for the slow control, the drive systems, auxiliary devices and the time synchronization network, but are connected directly to the central camera server or the White Rabbit network respectively. All connectors in the patch panels as well as the optical patch cables are based on LC plugs. For the connection between the Camera Servers and the central switch 2x10Gbit/s connections are planned for each Camera Server.

Furthermore an on-site wireless network is planned. All technical buildings will be covered as well as a small area around each telescope. The latter will allow for easy access to steer the telescopes locally during commissioning and maintenance. Especially, during the commissioning, but at any time as well it is important to have a further possibility to get access to the telescope. It could be that the Ethernet switch or the telescope tower is not accessible and so a wireless network provides one means of getting easy access from outside the tower. For the southern array, a wireless network of about 160 access points and a wireless control system will be necessary.

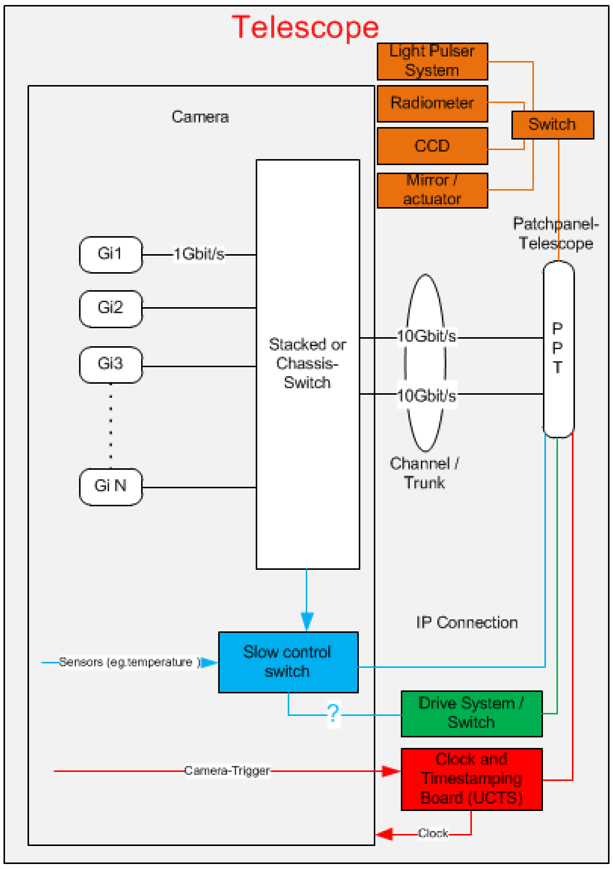


Figure 3.16 – Detailed view of the network topology of a generic telescope.

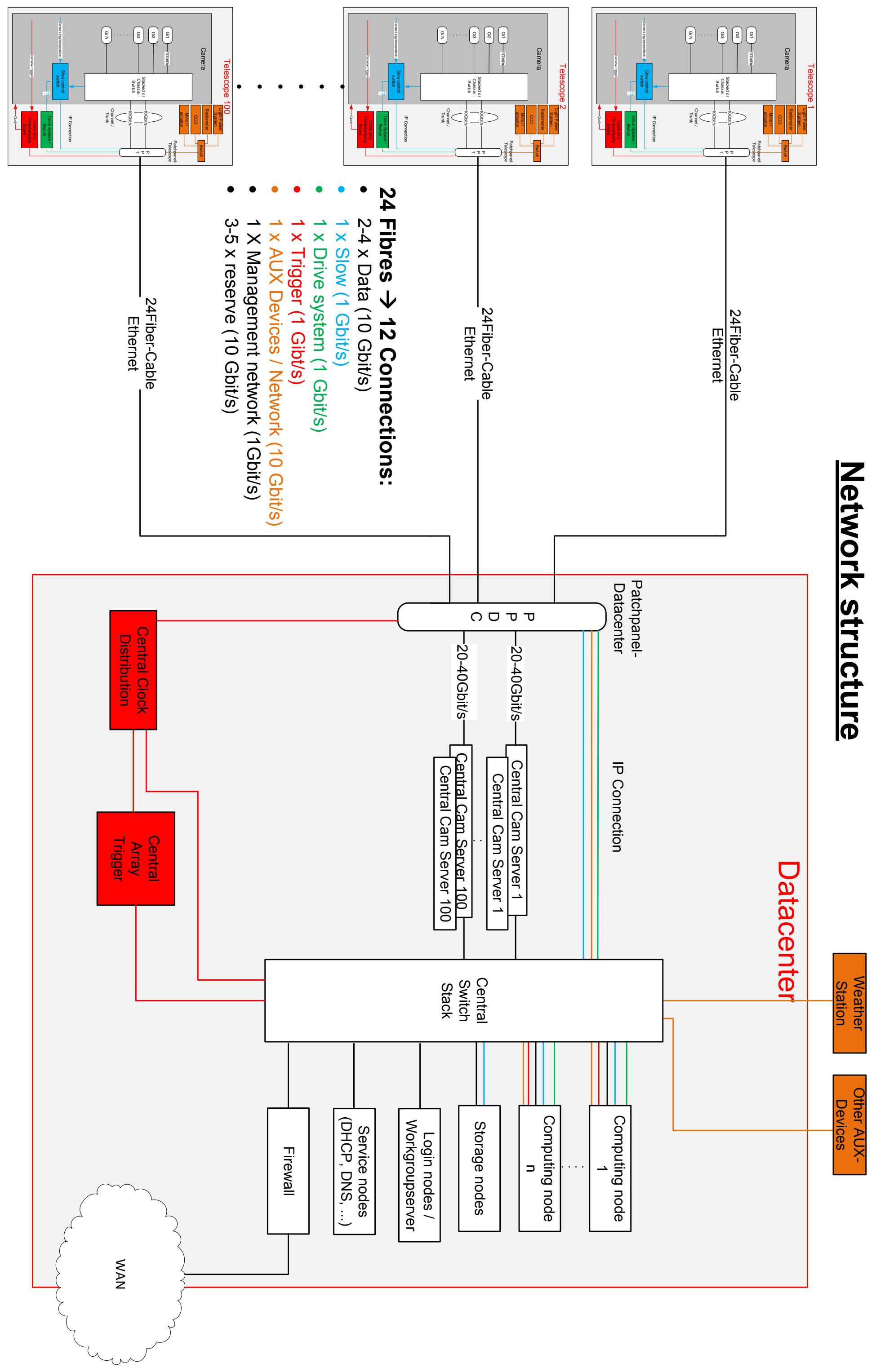


Figure 3.17 – Detailed view of the array network and data centre topology.

On-Site Computing Software

On-site operating system and system services The operating system for all servers will be based on Linux RedHat Enterprise (RHEL) re-compiled distributions (e.g. Scientific Linux, CentOS). RHEL has been developed for long-term supported commercial usage and Scientific Linux as RHEL re-compilation has successfully been used inside Particle (LHC) and Astroparticle Physics projects for many years. Inside CTA, important parts of the on-line and off-line software is already running under SL, e.g. the Grid middleware and the Alma Common Software (ACS). On-site services will include the required set of usual services needed for running a standard computing centre, like DNS, DHCP, TFTP for the deployment of the operating system on bare metal. In addition, services like printing, mail, instant messaging and video conferencing may be required. Backups of system configuration data, manually installed software, personal data from development environments, electronic logbooks and other logs will be needed and should be duplicated off-site. In general, the strategy will be to avoid running services unless absolutely needed. In most cases, the canonical choice for the software to use will be the one readily provided with the chosen operating system (Scientific Linux, or another clone of Red Hat Enterprise Linux, or RHEL itself see above). External access is a special kind of service due to its high impact on site security. User authentication and authorization for on-site user login will follow the CTA wide Single Sign On (SSO) approach defined by the DATA project. For this purpose a CTA SSO client will be provided to allow user access. For security reasons OS administrator rights (root privileges) will be handled differently under the responsibility of ACTL.

On-Site Security The site has to be protected against unauthorized access over the network. The most significant risks to consider are vulnerabilities in externally accessible services (in particular, login) and compromised credentials of authorized users. To minimize these risks, general best practices will be followed, like limiting network traffic to what's actually needed, keeping systems patched against remotely exploitable vulnerabilities, keeping systems with user access patched against locally exploitable issues, making use of hardening features like SELinux, running all software with the minimal required privileges, limiting any kind of access to the required minimum, minimizing the number of services to run, minimizing the amount of software installed on systems, optimizing all configuration for security, and using automated central maintenance and monitoring for all on-site systems to guarantee all this. Network security will be achieved by a firewall between the site and the internet, router ACLs between logical security zones (typically, Layer 2 subnets) and the stateful local firewall on each system. For remote access, multi-factor authentication options will be evaluated. It has to be decided which appointments and committees will be required, such as a security officer and deputy, a computer incident/emergency response team and a computer security council.

System administration On-site systems will be maintained using automated mechanisms bringing systems into the state defined in a central configuration database. It is planned to re-use a lightweight framework that has been in use for 5 years to run a compute facility of similar scale to the final CTA sites, which uses standard tools like RPM, YUM, Apache and git. A preliminary, not yet complete version is already used to manage the three Linux systems at the MST prototype project. It will be completed and further adapted to CTA needs for deploying and managing the testbed cluster that has been deployed early 2015. Change management procedures will have to be defined and implemented in order to maximize instrument availability while enabling progress and maintaining an adequate security level.

System monitoring The monitoring system is a system that constantly monitors every ICT component (e.g. servers, workstations, network equipment, software pipeline, UPS, cooling system, environmental parameters) in order to identify and notify the administrator of any anomaly due to overload or failure. An appropriate systems monitoring is neccessary to ensure the required reliability of the ICT in terms of continuity of service by preventing malfunctions and providing quick identification of faulted components. The CTA ICT monitoring will be implemented by integrating traditional monitoring tools used in the computer centre with specific custom tools which interface the ACS via OPC UA. The former will be based on SNMP (Simple Network Management Protocol) and standard software (e.g. Nagios and MRTG) and features embedded in the devices themselves, and will generate alerts by e-mail or SMS. In the latter case, the SNMP protocol will be converted into the OPC UA protocol in order to implement an OPC UA server that will interact with an ACS OPC UA client and, through the ACS Notification Channel, send monitor data and alerts to the central console of the CTA observatory.

3.2.6 ACTL-SCHED: Central Scheduler

In this section the product ACTL-SCHED (Central scheduler), numbered *3.6* in the CTA PBS structure, is described. This product includes both the long- and short-term planning.

The ACTL-SCHED design presented hereafter is based on some assumptions related to the CTA operational design (see Sect. 2.3). It might suffer require after the official approval of the observation strategy and the overall data flow. However, important changes are only expected on those aspects affecting the algorithm design, so that the overall software architecture should remain.

The main purpose of the Scheduler for CTA (SCTA) is the allocation of multiple tasks to one single array or to multiple sub-arrays of telescopes, while maximizing the use of the observatory operational time and maximizing the scientific return of the CTA proposals. This is an example of a multi-objective problem, since different factors must be optimized. An important limitation for the operational scheduler of CTA and for scheduling similar large Earth-based astronomical infrastructures is the fast reaction time required to respond to changing conditions or to high priority transient events (i.e., Targets of Opportunity and science alerts triggered by Level-A Analysis). A suitable long-term planner (LTP) that computes a complete season schedule is required to work together with a dynamic (or short-term) planner (STP) that provides the new schedule in a very short time, but keeping the long-term perspective to ensure the overall efficiency of the telescope operation. This process becomes impractical for manual planning due to the complexity in computing the enormous amount of possible combinations in the search for a near-optimal solution. Although these kinds of automatic planning and scheduling problems are considered NP-hard because they are impossible to solve in polynomial time, there are many mathematical tools to solve them: from simple heuristics to more complex Artificial Intelligence (AI) approaches [28].

Scheduler in the operational workflow

The scheduler is responsible for the proposal task planning and it is used in two different steps of the operational workflow: the off-line and the on-line planning activities. This two-step process is devoted to translating the scientific prioritization, based on the proposal excellence and on observatory access policies, into a feasible and optimized real-time schedule. The latter is responsible for delivering tasks to the CTA array control software. Different actors participate in these two steps.

- **Off-line planning:** The off-line activity produces a plan of objects to be observed in a period of time according to the CTA constraints that can be computed beforehand. It contains the LTP that is devoted to schedule object observations on timescales from several months to single nights. Long-term scheduling is mainly focused on filtering, prediction and simulation processes, and is based on real-world models. The LTP is not time-critical. A solution can be based on a computationally expensive AI approach. The solution obtained with this algorithm highly optimizes the objectives defined in the problem. In CTA, these objectives are related to fulfil the constraints, maximize the scientific return and to reduce the downtime between observations. Off-line planning is used by the CTA unit in charge of compiling the list of accepted proposals for a full season and by the array control software for the on-site operation.

- **On-line planning:** The on-line strategy contains a long-term and short-term scheduler. The latter reacts to unexpected situations by adapting the previously computed long-term plan. On-line planning is executed on-site providing the tasks that are to be executed by the array control software and taking into account those conditions that change dynamically (e.g., weather). This is a time-critical process devoted to fit a near real-time response with global defined objectives. Long-term results must be used in the short-term calculation to accomplish these global objectives.

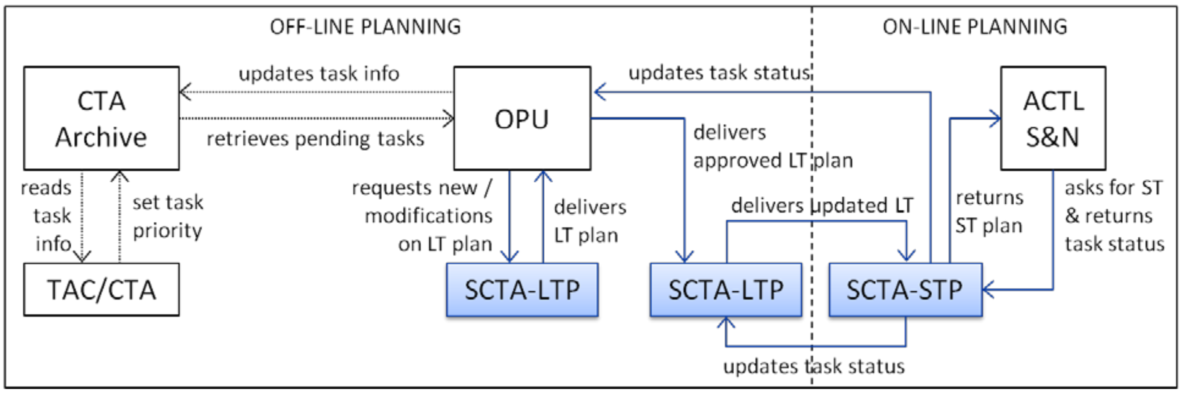


Figure 3.18 – High-level design for the CTA scheduler.

Figure 3.18 represents the current design of the interaction of the scheduler with the different control modules in the CTA operational work flow. Several steps are necessary to handle proposals from their definition to their execution and tracking.

- **Proposal and task definition:** CTA observations will be performed in response to scientific proposals submitted by the CTA consortium or by other users. This step will be based on a proposal handling tool and an evaluation process that will end with a set of proposals approved and prioritized by a time allocation committee (TAC). The proposals will be stored in an archive that will be used as an input data provider by the operations planning unit (OPU). The OPU is the entity devoted to provide the seasonal planning to the observatory and will use the SCTA to compute an optimum long-term plan.
- **Long-term plan:** A long-term plan will be computed by the LTP at the beginning of the observing period and will be optimized to maximize the number of completed proposals and the scientific return. The optimization process will be devoted also to minimize the time overheads and the gaps between the tasks to be executed by the observatory. This long-term plan will be updated periodically to address deviations from the real case. The CTA OPU will be responsible for supervising the process to obtain the long-term plan and adjust the target priorities, constraints and the scheduler configuration when needed. This process will be carried out off-line: real-time feedback from the observatory will not be taken into account to compute the plan. The resulting long-term plan will be transferred to the SCTA on-site in order to drive the task selection process carried out on the observatory. An additional long-term plan will then be computed on-site on a periodic basis, in order to update the initial schedule using the tasks executed in previous nights. This will provide the night schedule that will be taken as a reference by the STP.
- **Short-term planning:** The task selection during the night-time will be done in almost real-time and will take into account the changing conditions of the environment and system. This on-line planning will be carried out by the STP and will consider also transient phenomena when re-computing the operation night-plan. Observatory operators will have the capability to override the task selected by the scheduler. They will use a support tool to check the impact overriding this task would have on the observatory data production and efficiency levels and will be used to minimize any adverse effects caused by this human intervention.
- **Data-taking control:** Data-taking control performed by ACTL will include the monitoring and control capabilities for all subsystems that take part in the data acquisition process in the Southern and Northern observatories (ACTL S&N in Fig. 3.18).
- **Task tracking:** The execution of tasks will be tracked by the SCTA in order to update the schedule.

SCTA: the Scheduler tool for CTA

The SCTA uses two scheduling strategies, the off-line and the on-line strategy. The first one obtains a plan of objects to be observed in a period of time according to the CTA constraints (see Table 3.3) that can be computed beforehand, whereas the second one is devoted to schedule tasks in almost real-time, considering both the long-term perspective and the unpredictable changing conditions. Two different types of planning are then considered: the LTP for the off-line schedule of tasks for a complete observing period (e.g., six months) and a combination of the LTP and STP, which are suited to create schedules on-line for one night in quasi real time. Fig. 3.19 shows the activity diagram of the SCTA. The first process (1 in Fig. 3.19) is used by the CTA operations planning unit (OPU in Fig 3.18) as a support tool to compute the long-term plan. It corresponds to the off-line computation and initially provides a set of solutions based on different objectives. One of the computed solutions is selected by the team in charge and is delivered to the observatories to be used as a reference to drive the on-line scheduler. This first process is required only at the beginning of the observatory operational season, although it can be carried out at any time to introduce corrections to the initial plan.

The second process (2 in Fig. 3.19) recomputes the long-term plan every day using the same algorithm as in process 1 and considering the tasks already executed (Executed Blocks) and the status of the proposals. The long-term solution is then updated by the STP, that is, configured to deal with operational events and create schedules in quasi real time responding to changing conditions. Targets scheduled by the LTP for the following night are considered with highest priority by the STP. The on-line procedure can be executed through the ACTL interface or automatically. In the automatic mode, the LTP will be executed before the night starts, and the STP will be executed when a new observation is required at a specific point in time. The LTP has a flexible configuration and its execution frequency and temporal scope can change. In terms of communications, ACTL will interact with SCTA by means of three operations:

- Informs about housekeeping: ACTL sends information about the housekeeping to SCTA. This information will contain the environmental (e.g., weather) and the system conditions.
- Requests an observation: a new observation is required from SCTA (on-line module) by ACTL. The STP will consider the system conditions when computing the new observation.
- Informs about the observation status: the ACTL sends the status of the last observation executed by the observatory. The status will indicate the observation duration and whether it has been completed or stopped. In the other direction, SCTA communicates with ACTL to deliver an observation: SCTA (on-line module) sends a new task to be executed to ACTL.

The main difference between LTP and STP algorithms is that the short-term algorithm can deal with events during the computation of new schedules. These events are useful in order to let the scheduler know when an important change occurs such as the activation/deactivation of one sub-array, the weather conditions or the current status of a scheduled block which is under execution.

Type	**Constraints**	**Description**	**Computation**	**Scope**
Hard	Night	Objects must be observed in night conditions, measured in terms of Sun altitude	In advance	LT / ST
Hard	Sky brightness	Night sky brightness is defined as dark depending on different Moon variables: altitude, phase and distance of the object to the Moon. Dark, grey and bright conditions are considered and must be fulfilled according to the specified task execution pattern	In advance	LT / ST
Hard	Target visibility	Objects must be observed when they are above the horizon	In advance	LT / ST
Hard / Soft	Zenith angle distance	Objects zenith angle must be minimized to increase the observation quality. A maximum value is given and must be taken as a limit	In advance	LT / ST

Hard	Overlapping	A telescope can only do one task at a time, so the operation tasks must be scheduled avoiding overlapping between them, including the time to reconfigure the system (i.e., slew time). CTA can operate in sub-arrays or in full array configuration, where the same constraint must be applied	In advance	LT / ST
Hard	Avoidance areas	The contaminating source of a specific region must be considered in order to avoid distortion in the target observations (e.g., Moon, planets, bright stars)	In advance	LT / ST
Soft	Task priority	Number of priority tasks observed during the season or during a specific period of time must be maximized	In advance	LT / ST
Hard	Slewing	Pointing to a particular target and acquiring data requires a specific configuration. Thus, time to transfer to a new configuration must be taken into account when computing the mission planning. This reconfiguration time mainly depends on the slewing speed of the satellite	In advance	LT / ST
Soft	Observing time	Fraction of the operational time that the observatory is carrying out science tasks must be maximized	In advance	LT / ST
Hard	Time constraints / Monitoring	Minimum observable time in a day, monitoring conditions for periodic observations, or fixed times windows have to be considered when scheduling a task	In advance	LT / ST
Soft	Task Completeness	Number of tasks completed during a specific period of time has to be maximized. Incomplete tasks may not produce any scientific return. A task with a significant level of completion must be promoted	In advance	LT / ST
Hard	Environment and system conditions	The task scheduling has to be adapted to the environment and the system conditions (i.e., weather, availability of telescopes)	On the fly	ST

Table 3.3 – Hard and soft constraints for the SCTA (Long-term and Short-term labelled LT and ST, respectively). The Scheduler must comply with a pre-defined set of constraints. Some of them can be computed in advance to check how they impact the schedule of tasks.

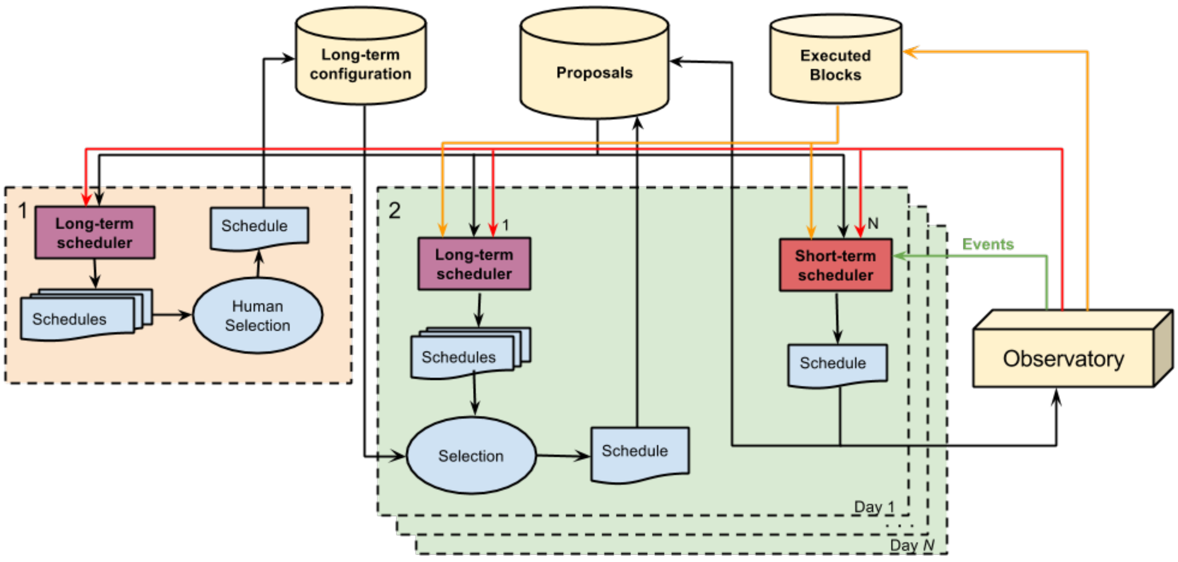


Figure 3.19 – SCTA activity diagram. Two separated blocks describe off- and on-line planning activities and are labeled with 1 and 2, respectively.

3.2.7 Beyond ACS: Reuse of High-Level ALMA Software

Because the scale and topology of ALMA and CTA are similar, it is natural to ask whether there might not be higher-level software from ALMA whose reuse could reduce ACTL's development effort and provide modules that have already been proven in the field. The use of ACS as the software framework for ACTL, over and above its value in its own right, also reduces the cost of reusing such high-level ALMA software, which itself is built to conform to the ACS paradigm and exploit ACS services. A non-exclusive list of such items are listed here:

- Controller/Device Hierarchy.
- Control Command Language.
- TMCDB-Explorer.
- Shift Log Tool.

Further details on the strategy to adapt these items are given later in the corresponding sections of the text.

Collaboration with ALMA

Although individuals within the ALMA project have been generous with their time and often available to answer questions or offer explanations of the characteristics and behaviour of individual items of ALMA software, the groups developing software for the MST and ASTRI prototypes decided to draft a formal Software Sharing Agreement with the representatives of the the ALMA Executives, namely, ESO, AUI, NAOJ and the Joint ALMA Office (JAO) in Chile. The agreement, signed by the Directors of the participating institutes in 2013, gives the MST and ASTRI projects access to all of ALMA's open-source software and "will permit the Institutes to share this ALMA software with other collaborating institutes within the CTA consortium, provided that such institutes adhere to the terms of this agreement." Although the software is provided "as-is," without warranty, the possibility of arranging some support for its use is mentioned, as well as the possibility that feedback from the MST and ASTRI projects will be followed by fixes or enhancements to the software.

Since the signing of this agreement, there was a joint ESO/ALMA-MST-ASTRI meeting in June 2013, to discuss a shared approach to finding a satisfactory technical solution to the problem of retrieving information from the large quantity of monitor data produced by ALMA and expected to be produced by CTA.

In December 2014, a three-day ACS Workshop prepared by the ALMA personel was held on the ESO premises; it included both a basic course for developers from ALMA, CTA and other projects, as well as an advanced track that addressed questions and issues brought by the ACTL groups, e.g. working with the MST and ASTRI prototype projects, that are more advanced in their use of ACS.

3.3 Interfaces

3.3.1 Overview

The ACTL work package is the project within CTA that has the highest number of interfaces with other CTA work packages. A well-described set of interfaces, both hardware and software related, are key for CTA to reach the required high standard of availability and reliability. In addition, the interface descriptions allow setting boundaries between the different work packages and distributing the responsibilities for the products interfacing with ACTL.

All external interface descriptions will be detailed in a dedicated interface control document (ICD) database as defined in the CTA interface management plan [29]. The ICDs will contain information on all logical, physical and responsibility interfaces between two projects. In addition, ACTL also considers the *internal* interfaces between pairs of ACTL sub-work packages, part of the internal documentation of ACTL, to be detailed in dedicated ICDs. The philosophy as described in [29] is to develop and iterate the interfaces in a joint approach between the involved projects aiming for a shared understanding and agreement on the description. To work out these ICDs is a major task for ACTL. A dedicated team of software developers inside ACTL, composed in a way that expertise covering all ACTL sub-tasks is present, will develop the interface description together with the respective project teams in an iterative process including writing and reviewing stages.

Whenever there are similar interfaces to be described for different projects interfacing ACTL, a uniform interface description will be found for all projects. This will allow simplifying the ACTL system and sharing of software among the projects, thereby reducing development costs. This is of particular importance for easier development and sharing of the low-level and high-level software for the integration of the telescopes and auxiliary devices into the ACTL system. The importance of uniform interfaces for the maintainability and reliability of CTA should not be underestimated.

An overview on the interfaces to be described is given in this section.

3.3.2 External Interfaces

ACTL has external interfaces with the following CTA work packages: all telescope work packages xST, COM, DATA, OBS, and INFRA. A short summary of the main interfaces will be given in this section.

ACTL-xST interface

In the context of CTA, *assemblies* are primary elements of a telescope. The control, monitoring and readout of each individual telescope's assemblies and additional auxiliary devices attached to the telescope must be integrated into the central array control software. The interface description is the basis for the integration. The list of assemblies to be interfaced for each telescope is (not complete list):

- drive system
- Cherenkov cameras
- CCD cameras
- active mirror control
- calibration devices
- structure monitoring sensors
- clock distribution
- trigger system

Whenever possible, the interface description should be uniform for all telescope projects to allow for easier software development and sharing of software. The interfaces can be tested by the telescope prototype projects.

Device Integration based on interface descriptions The device integration scheme is presented in Sec. 3.2.1 and in [4]. In this paradigm, the lower-level instrument software is created by the telescope teams and consists of OPC UA servers providing instrument access while hiding the hardware specifics. The OPC UA servers are then integrated into the ACS-based ACTL software framework via software *bridges*, which are the responsibility of the ACTL team. Here, the interface between the two projects is located. The interface description to be created for each assembly will contain all the control, monitoring and readout information in addition to connection type and protocol.

Naming convention For reasons of simplified software development and uniformity, ACTL will propose a uniform naming space convention for device integration to name each monitoring point and method in the associated OPC UA server uniquely. Each named object in the OPC UA namespace should be uniquely identified by its name. This implies providing a naming convention where the full pathname will illustrate the device's dependencies and hierarchy in an assembly. Each device included in the array control system will be registered and then associated with its root pathname. Device integration tests will be based on this naming convention. The same devices being used on telescopes of different types will only differ by their root pathnames, but not on the software to be used.

Analysis and calibration algorithms: In addition, any algorithm used for e.g. calibration of devices (Cherenkov cameras, auxiliary devices) and which should be executed centrally within the ACTL framework will be described as an interface. The respective algorithm will be made available from the instrument teams to ACTL as a code fragment to be implemented in the ACTL software framework or as an external library.

Readout of Cherenkov cameras: The ACTL-DAQ system must retrieve data from different kinds of telescopes and write the event data in a unified format to the central repository. Discussions are ongoing to reach an agreement on a common raw data format and readout API. Some differences may remain, both in terms of format and protocol due to high throughput and specific constraints on the camera server sides, but must be minimized. The proposed API is written in C++ and based on the protocol buffers and ZeroMQ. Specific implementations are under consideration to embed this interface into the camera server's environment, often in C. If some fundamental differences remain and cannot be resolved by the camera servers, on-the-fly format conversion has to be performed by the ACTL-DAQ receiver, reducing the overall performance of the ACTL-DAQ system.

ACTL-TRIG also plays a role in this process, as the SWAT sends array trigger messages to initiate transfer of bulk data from the camera server buffer to more permanent storage and to the level-A analysis pipeline (DATA-PIPELINE-RTA). The full format of these messages must accommodate the needs of all telescope cameras and will be detailed in a forthcoming ICD.

Clock distribution and trigger timestamping: One of the most critical ACTL-xST interfaces is the common interface between the UCTS board mentioned in Fig. 3.2.4 and the various Cherenkov cameras. The complete definition will be the subject of upcoming interface workshops.

The current baseline interface and UCTS design require the following:

1. The UCTS must deliver both a $1\,\mathrm{PPS}$ and a configurable input clock of at least $10\,\mathrm{MHz}$ to the corresponding camera (most likely these are LVDS signals). The camera clock is synchronized to these signals, i.e. the UCTS acts as a master camera clock.
2. The UCTS receives a trigger signal from each camera. It timestamps the rising edge of this signal with nanosecond precision relative to all other CTA cameras. Trigger signals may also be received from other sources (e.g. LIDAR events acting as veto events in the SWAT).
3. A still to be defined serial two way communication interface will be available between the UCTS board and the attached hardware. This will be used to transfer a dedicated trigger type provided by the cameras for non-cosmic events (muons, calibration) and can be used to transfer a reset command to the cameras (explained below).
4. The UCTS will receive a status level from the camera when the camera is occupied with reading out, tag triggers received during this period appropriately, and make a measurement of the telescope deadtime.
5. The UCTS should be capable of delivering centrally-generated calibration triggers back to the cameras.

A dedicated server (CDTS-Server) forwards the UCTS timestamp packets (White Rabbit timestamp plus auxiliary information) to both the SWAT and the camera servers. The SWAT will use the timestamp packets to reach an array trigger decision and forward readout instructions, including timestamp information, to the camera servers.

A “double-clock/double-timestamping” setup, illustrated in Fig. 3.20, will be used to correlate the timestamp information produced by the UCTS with the camera bulk data for each camera. The UCTS produces two pieces of (meta)data for every camera trigger: the precision WR timestamp and an event counter that increments each time a trigger is received. The cameras are also expected two produced two parallel pieces of information: a timestamp counter based off of the camera clock (which is slaved to the UCTS clock signals) and an identical event counter that increments each time a trigger is produced. Matching and merging of the camera bulk data and the UCTS timestamp information can be done (by default) on the basis of the event counters. This scheme provides a high level of redundancy and reliability, since the secondary (redundant) timestamp counter produced by the cameras permits software located at the camera servers to monitor for a variety of error conditions (see 4.5.2 for further details). When appropriate (start of run, error conditions) a reset signal can be issued by the UCTS to allow all counters to be reset to zero. It should be noted that the resolution of the camera timestamp counter may be coarser than that of the timestamp generated by the UCTS and will vary from camera to camera (e.g. 4 ns for FlashCam, 100 ns for NectarCam). The overall impact of the interface design on CDTS reliability is also discussed in sections 4.3 and 4.5.2.

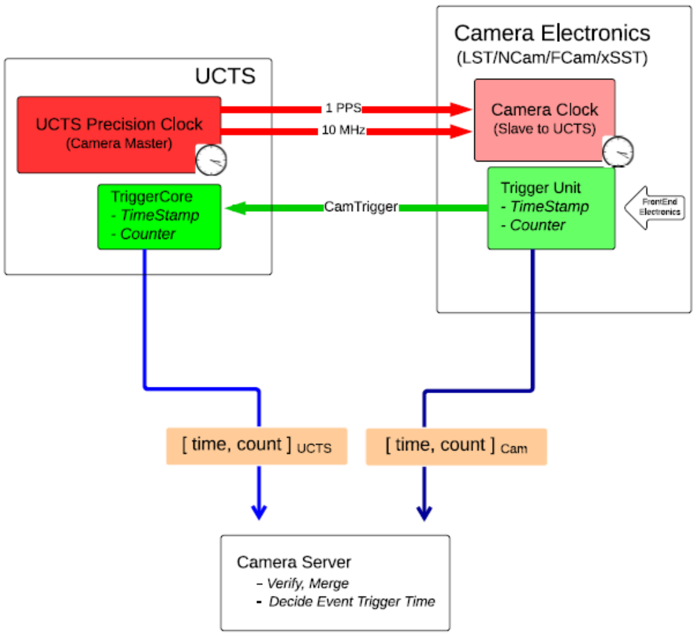


Figure 3.20 – Layout of the double-clockdouble-timestamping archiecture. Both UCTS and Camera generate per trigger a trigger-message $[t, count]$ with a trigger counter and a timestamp. The camera server merges the data and verifies integrity of the merge based on the parallel timestamps and counters. Monitoring software at the camera server will alarm when mismatches (indicative of problems with either the UCTS, camera, or camera-UCTS interface) are detected.

Inter-telescope triggers: For certain telescope designs, namely the large-size telescopes (LSTs) and Schwarzschild-Couder mid-sized telescopes (SCT-MSTs) some type of real-time hardware inter-telescope trigger is envisaged to reduce telescope dead time and to permit these telescopes to operate at a lower energy threshold. The details of the inter-telescope trigger design, implementation, and algorithms are the responsibility of the individual telescope groups and will be detailed in the corresponding sections of the telescope TDRs; any additional demands these designs impose on the fibre network are likewise the responsibility of these groups and ACTL-ONSITE. From the point of view of ACTL and the array trigger system, only the following features are relevant and specified by ACTL as required for any inter-telescope trigger system:

1. The inter-telescope trigger operates between a small patch of telescopes only (no more than 9-15 telescopes), in order to keep signal round trip times below 5 microseconds.
2. The inter-telescope trigger at each telescope operates autonomously, i.e. each telescope decides on its own whether or not to trigger, but uses information from nearby telescopes to do so.

3. The cameras with inter-telescope triggers must provide a measurement of the additional trigger latency for each telescope (i.e. the time between the initial camera trigger and the inter-telescope trigger decision) and store these values in a database accessible to the SWAT.

From the point of view of the array trigger system, the operation of the inter-telescope triggers is essentially transparent; the inter-telescope trigger appears to the array trigger system to be another camera trigger. The only caveat is that additional information, related to the operation of the inter-telescope trigger, may be forwarded along with the camera trigger information for inclusion in the bulk camera data. The exact data format will be specified in forthcoming ICDs.

ACTL-COM

The external interfaces to the COM work package include the integration of all devices for calibration and environment monitoring using the same device integration paradigm as presented in the previous paragraph for integration of telescope assemblies. Uniform interfaces for devices of the same type shall be used to allow for sharing of software.

The interface description will cover, in addition to monitoring and control, the communication type and protocol, the computing and software setup as well as configuration parameters and data analysis algorithms, which shall be available at a central place. The list of devices to be integrated into the ACTL framework is:

- LIDARs
- All-Sky-Camera
- FRAM ((Ph)Fotometric Atmospheric Robotic Monitor)
- Ceilometer
- Anemometers
- Weather Stations
- Rain Sensors
- Sun/Moon Photometers
- Octocopter
- Central Laser Facilities
- UVScope
- Illuminator

Environmental impact on observation planning: The measurements performed by the central calibration and monitoring devices will be provided to ACTL-OPS to assess the environmental conditions and forward this information to ACTL-SCHED to adjust the observation schedule accordingly. The corresponding interface will be defined in coordination with the proposal handling sub-task of DATA as it impacts the science operation planning of CTA.

Impact of laser operation on observation planning: Laser beams in the field of view of observatories might impact their data-taking. Therefore, any laser operated on the CTA site has to register its current pointing position at the laser control traffic system to inform other observatories nearby the CTA site. Vice versa, the CTA array should not be pointed into the direction of any operating laser of nearby astronomical facilities. An interface will describe this scenario, its definition will depend on the actual setup of nearby observatories.

ACTL-DATA

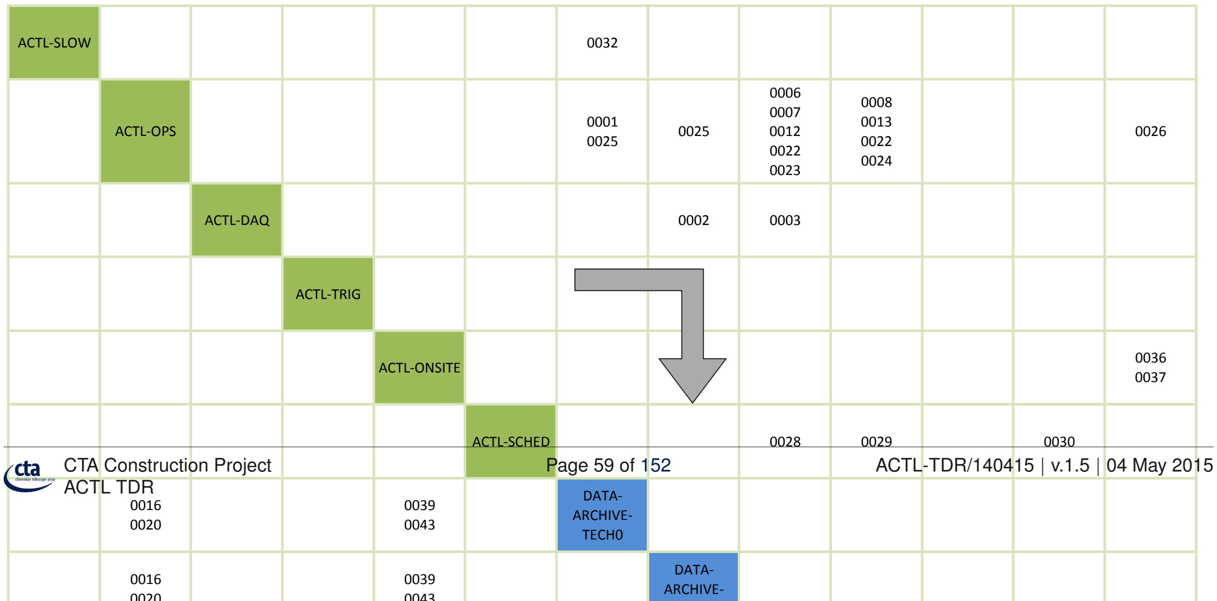
For CTA to reach its goals in science operation, it is key that the two work packages ACTL and DATA work closely together in the process of defining the interfaces. In addition to the integration of the DATA tasks of data quality assessment, calibration and data analysis performed on-site, the DATA work package is a stakeholder for the interface descriptions ACTL-xST and ACTL-COM. The ACTL work package thereby constitutes the bridge between the instruments and engineers on one side and the scientists on the other side, to ensure that all information that is provided is complete and available.

Most of the interfaces ACTL-DATA are software-related (logical interfaces), but topics such as the on-site ICT infrastructure or sharing of the outgoing internet line are also covered. A summary list of the identified interfaces is given in Tab. 3.4 and as a N2 diagram in Fig. 3.21 where the following acronyms refer to DATA PBS items:

- DATA-ARCHIVE: Archive software
- DATA-PIPELINE-RTA: Level-A Analysis software (Real-Time Analysis)
- DATA-PIPELINE-PRA: Level-B Analysis software (Preliminary Analysis)
- DATA-OBS-PH: Proposal Handling Software
- DATA-ICINFRA-DTU: File Transfer Unit Software
- DATA-ICINFRA-MONITOR: Monitoring Tools Software
- DATA-ICINFRA-TICKET: Ticketing Software

Table 3.4 – Overview of the external interfaces ACTL-DATA (not complete list).

ACTL to DATA Interfaces		
Interface ID	Type	Functional interface
I-ACTL-DATA-0001	Logical	ACTL-OPS provides TECH data from CTA assemblies to DATA-ARCHIVE-TECH0
I-ACTL-DATA-0002	Logical	ACTL-DAQ provides EVT0/CAL0 data to DATA-ARCHIVE-EVT0/CAL0
I-ACTL-DATA-0003	Logical	ACTL-DAQ provides EVT0/CAL0 data to DATA-PIPELINE-RTA
I-ACTL-DATA-0004	Logical	RTA provides Data Volume Reduction algorithms to ACTL-DAQ
I-ACTL-DATA-0005	Information	EVT0, CAL0 data format
I-ACTL-DATA-0006	Logical	ACTL-OPS provides a sample of TECH0 data to DATA-PIPELINE-RTA
I-ACTL-DATA-0007	Logical	ACTL-OPS sends TECH0 alarms to DATA-PIPELINE-RTA
I-ACTL-DATA-0008	Logical	ACTL-OPS sends TECH0 alarms to DATA-PIPELINE-PRA
I-ACTL-DATA-0009	Logical	DATA-PIPELINE-RTA sends Data Quality Alerts to ACTL-OPS
I-ACTL-DATA-0010	Logical	DATA-PIPELINE-RTA sends Science Alerts to ACTL-SCHED
I-ACTL-DATA-0011	Logical	DATA-PIPELINE-RTA sends Science Alerts to ACTL-OPS
I-ACTL-DATA-0012	Logical	ACTL-OPS provides display system for the RTA monitoring system
I-ACTL-DATA-0013	Logical	ACTL-OPS provides display system for the PRA monitoring system
I-ACTL-DATA-0014	Logical	ACTL-OPS monitors DATA-PIPELINE-RTA
I-ACTL-DATA-0015	Logical	ACTL-OPS monitors DATA-PIPELINE-PRA
I-ACTL-DATA-0016	Logical	ACTL-OPS monitors DATA-ARCHIVE
I-ACTL-DATA-0017	Logical	ACTL-OPS monitors DATA-ICINFRA-DTU
I-ACTL-DATA-0018	Logical	DATA-PIPELINE-RTA sends logging info to ACTL-OPS and display status
I-ACTL-DATA-0019	Logical	DATA-PIPELINE-PRA sends logging info to ACTL-OPS and display status
I-ACTL-DATA-0020	Logical	DATA-ARCHIVE sends logging info to ACTL-OPS and display status
I-ACTL-DATA-0021	Logical	DATA-ICINFRA-DTU sends logging info to ACTL-OPS and display status
I-ACTL-DATA-0022	Logical	RTA and PRA use the Alarm Condition Database
I-ACTL-DATA-0023	Logical	ACTL-OPS controls DATA-PIPELINE-RTA
I-ACTL-DATA-0024	Logical	ACTL-OPS controls DATA-PIPELINE-PRA
I-ACTL-DATA-0025	Logical	CTL-OPS controls DATA-ARCHIVE
I-ACTL-DATA-0026	Logical	ACTL-OPS controls DATA-ICINFRA-DTU
I-ACTL-DATA-0027	Information	TECH0 data format
I-ACTL-DATA-0028	Logical	ACTL-SCHED provides short-term schedule to DATA-PIPELINE-RTA
I-ACTL-DATA-0029	Logical	ACTL-SCHED provides short-term schedule to DATA-PIPELINE-PRA
I-ACTL-DATA-0030	Logical	ACTL-SCHED provides planned and executed observations to DATA-OBS-PH
I-ACTL-DATA-0031	Logical	DATA-OBS-PH provides the long-term scheduling to ACTL-SCHED
I-ACTL-DATA-0032	Logical	ACTL-SLOW provides a subset of the Instrument Configuration to ARCHIVE
I-ACTL-DATA-0033	Logical	ACTL and DATA share outside internet connection
I-ACTL-DATA-0034	Logical	ACTL and DATA share software management tools on-site
I-ACTL-DATA-0035	Logical	ACTL and DATA share test-bed for software verification/validation
I-ACTL-DATA-0036	Logical	ACTL-ONSITE sends information to DATA-ICINFRA-MONITOR
I-ACTL-DATA-0037	Logical	ACTL-ONSITE sends tickets to DATA-ICINFRA-TICKET
I-ACTL-DATA-0038	Logical	DATA-ICINFRA-TICKET sends tickets to ACTL-ONSITE
I-ACTL-DATA-0039	Resp. int.	DATA subsystems query ACTL-ONSITE for user authentication
I-ACTL-DATA-0040	Resp. int.	DATA provides specification for the running environment to ACTL-ONSITE
I-ACTL-DATA-0041	Resp. int.	ACTL provides a running environment to DATA
I-ACTL-DATA-0042	Logical	DATA-PIPELINE-OSA is deployed on ACTL-ONSITE
I-ACTL-DATA-0043	Logical	DATA-ARCHIVE is deployed on ACTL-ONSITE
I-ACTL-DATA-0044	Logical	DATA-ICINFRA-DTU is deployed on ACTL-ONSITE



Integration of DATA into ACTL: The defined interfaces make sure that DATA software applications running on-site (level-A and level-B analysis, on-site archive and data transfer) can be fully integrated into the ACTL system, which includes their execution (start, stop, ..), monitoring (usage, alarms, logging, ..) and display (operator GUIs, monitoring GUIs, ..).

Usage of ACTL on-site ICT infrastructure by DATA: In addition, ACTL-ONSITE provides the computing and networking infrastructure for DATA to perform the calibration and analyses (level-A, level-B) tasks as well as the storage infrastructure to hold the acquired data including calibration and analyses results available for the required amount of time to transfer them to the off-site data centres and check their integrity. Therefore, the data and computing model proposed by DATA influences the requirements for the on-site computing and storage capabilities. In addition, the performance of the DATA level-A and level-B analyses have influence on the chosen computing capabilities. All the inputs need to be defined and, at the end of the implementation phase, tested on the ACTL test bed cluster during full system integration and performance tests.

Sharing of outgoing internet access line: The amount of raw Cherenkov data produced on-site is large when compared to the available bandwidth of the order of 1 Gbit/s for transfer to the off-site data centres. While high-performance data reduction and event filtering scenarios are envisaged to reduce the overall data volume to be transfered, the sharing of the available bandwidth needs to be well organized between ACTL (tasks include e.g. software updates, remote access for monitoring, control and debugging, transfer of logging information, detector status reports and observation schedules), DATA (primarily transfer of data) and the local observatory crew (e.g. internet access, video/audio conferencing, mailing). An appropriate interface description will cover this topic.

Data transfer: A reliable system needs to be implemented for data transfer to the off-site data centres, which will allow the transfer and subsequent removal of data available on-site. A data transfer unit will be developed by DATA, which can be fully integrated into the ACTL system.

Remote access: For security reasons, remote access to the on-site computing and storage cluster will be highly restricted and most, if not all, data analysis tasks fully automated. Scenarios that depart from this general rule will be covered by a dedicated interface description.

ACTL-OBS

The interface of the ACTL work package to the OBS work package mainly affects the scheduler sub-task of ACTL. A (not complete) list of identified interfaces is given in Tab. 3.5.

Table 3.5 – Overview of the external interfaces ACTL-OBS.

ACTL to OBS Interfaces		
Interface ID	Type	Functional interface
I-ACTL-OBS-0001	logical	OBS requires a long-term schedule from ACTL-SCHED
I-ACTL-OBS-0002	logical	OBS requires an updated version of the long-term plan from ACTL-SCHED when the it is modified
I-ACTL-OBS-0003	logical	ACTL-SCHED when computing again the long-term plan
I-ACTL-OBS-0004	logical	OBS transfers the approved long-term plan to ACTL-SCHED
I-ACTL-OBS-0005	logical	ACTL-SCHED requires a well defined set of proposals from OBS
I-ACTL-OBS-0006	logical	ACTL-SCHED computes an offline long-term plan and sends it to OBS
I-ACTL-OBS-0007	logical	ACTL-SCHED sends metrics to check quality and suitability of the long-term plan to OBS
I-ACTL-OBS-0008	logical	ACTL-SCHED updates task status in OBS

ACTL-INFRA

The external interfaces to the INFRA work package affect the ACTL-ONSITE sub-task and include all required infrastructure to operate the on-site data centre and connect all devices with the network infrastructure, which includes cable trenches, power connections, buildings (incl. power, cooling, ..) as well as the outside gateway infrastructure. Detailed information is currently being worked out and parameters will be fixed once the overall scale of the network and data centre infrastructure is known.

3.3.3 Internal Interfaces

Table 3.6 – Overview of the software-related functional internal interfaces (not complete).

ACTL Internal Interfaces		
Interface ID	Type	Functional interface
I-ACTL-INTERNAL-0001	Logical	OPS requests status/summary information to DAQ for the operator interface
I-ACTL-INTERNAL-0002	Logical	OPS transmits command to start DAQ processe
I-ACTL-INTERNAL-0003	Logical	DAQ transmits alarms to OPS
I-ACTL-INTERNAL-0004	Logical	DAQ gets configuration data from the CDB (part of OPS)
I-ACTL-INTERNAL-0005	Logical	OPS request status/summary information to TRIG for the operator interface
I-ACTL-INTERNAL-0006	Logical	OPS transmits command to start TRIG processes
I-ACTL-INTERNAL-0007	Logical	TRIG transmits alarms to OPS
I-ACTL-INTERNAL-0008	Logical	TRIG gets configuration data from the CDB (part of OPS)
I-ACTL-INTERNAL-0009	Logical	OPS queries SCHED for the short-term schedule and next tasks to be execute
I-ACTL-INTERNAL-0010	Logical	SCHED provides the short-term schedule, including task execution constraints
I-ACTL-INTERNAL-0011	Logical	OPS provides status of the task execution and status of system and environmental conditions to SCHED to update the initial variables for re-calculation of the short-term schedule
I-ACTL-INTERNAL-0012	Logical	OPS requests status/summary information about ICT status from ONSITE for the operator interface
I-ACTL-INTERNAL-0013	Logical	ONSITE transmits ICT related alarms to OPS
I-ACTL-INTERNAL-0014	Logical	OPS initiates SLOW to (re-)configure devices for a certain task
I-ACTL-INTERNAL-0015	Logical	SLOW transmits instrument monitoring information to OPS
I-ACTL-INTERNAL-0016	Logical	OPS reacts to instrument alarms according to the alarm register defined in SLOW
I-ACTL-INTERNAL-0017	Logical	SLOW transmits pointing information to TRIG for trigger setup
I-ACTL-INTERNAL-0018	Logical	TRIG provides trigger and time stamp information for the event building of the DAQ
I-ACTL-INTERNAL-0019	Logical	TRIG-CDTS provides local trigger time stamp information to TRIG-SWAT for array trigger detection

The overall design as presented in Sec. 3.2 presents the main building blocks of the ACTL system for which internal interfaces need to be defined. A detailed description cannot be presented at this stage of the project as the detailed architectural design of the ACTL system is not yet fixed. The definition of use cases [15], which is one of the steps towards the detailed architectural design, is an important input both for the architectural design and the functional definition of interfaces. All software-related building blocks will be integrated into the higher-level ACS-based software framework of ACTL and the internal interface descriptions should reflect this paradigm. A summary (not complete) of the internal interfaces is given in Table 3.6 and in Fig. 3.22.

In addition to the functional internal interfaces between the different (software) building block of the ACTL system, the software systems need to run under the computing and network environment provided by the ACTL-ONSITE sub-work package. A detailed specification of the computing and networking infrastructure, which will be designed to meet the requirements of the different building blocks, is currently being worked out. During the design and implementation phase (pre-construction phase) of CTA ACTL, a test bed cluster will be available for a system integration and performance tests of the full ACTL system. In this way, compatibility can be assured. In addition, the internal interfaces of the ACTL system will be tested at the prototype telescope projects.

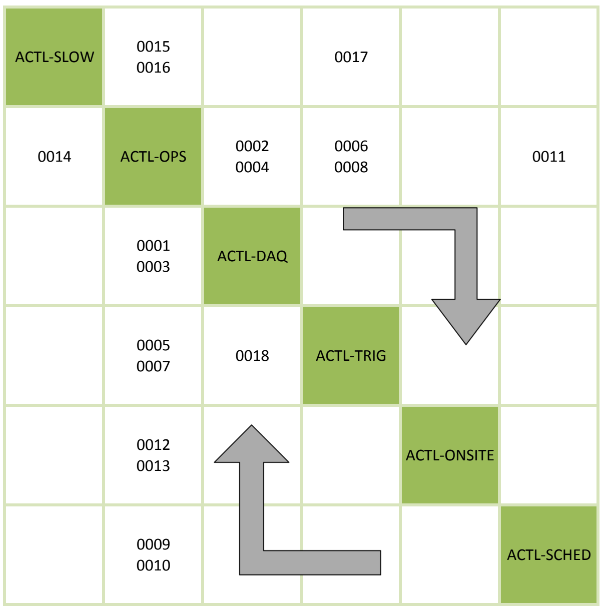


Figure 3.22 – N2 diagram illustrating the interfaces between ACTL sub-products corresponding to the highest-level PBS item for each sub-work package. Data flows in a clockwise direction between the producs as illustrated by the gray arrows. The numbers in the cells refer to the interface ID as shown in Table 3.4 (e.g. 0001 corresponds to I-ACTL-INTERNAL-0001).

3.4 Prototypes and Tests

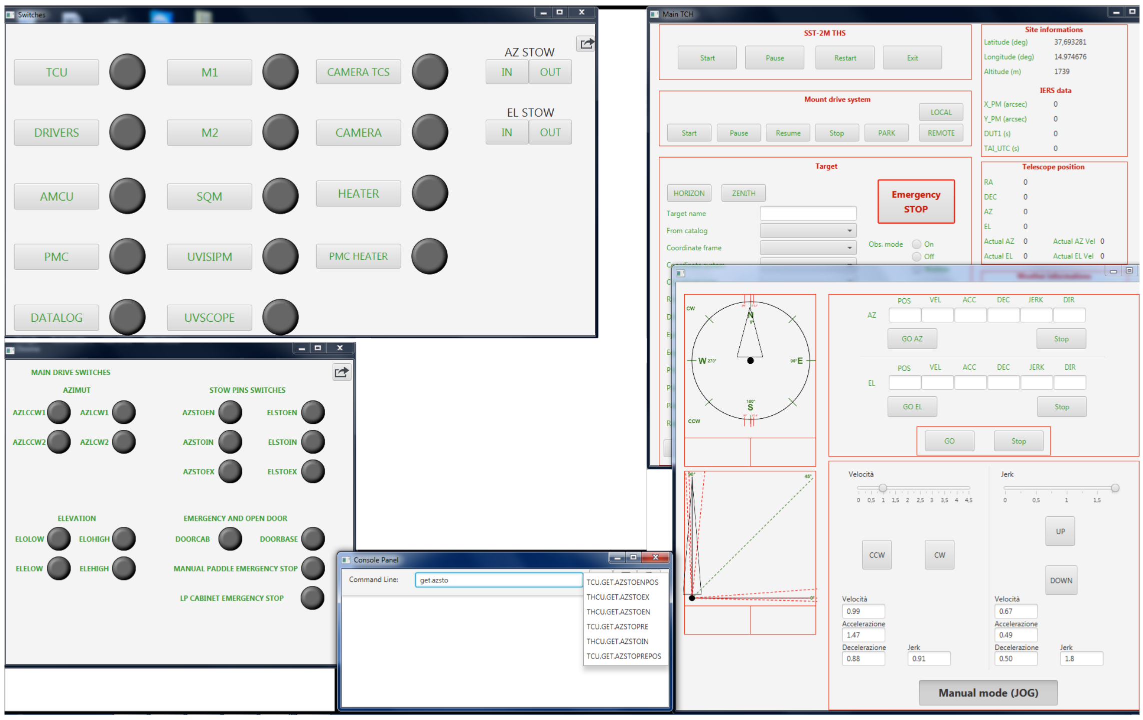
Prototyping activities have played a major role in the ACTL project during the last few years. They have provided a crucial input to assess the proposed technologies and frameworks, and allowed starting the development of tools that will be deployed in the operational CTA arrays. Generally speaking, two kind of prototyping activities have been carried out: those constituting prototypes of final ACTL products and those activities to steer telescope prototypes using the technologies and concepts proposed by the ACTL team.

3.4.1 ACTL-SLOW: Instrument slow control software

In this section the propotypes associated with ACTL-SLOW (Instrument Slow Control Software, see Sec. 3.2.1) numbered *3.1* in the CTA PBS structure, are described. See also later Section 3.4.7, that describes telescope prototypes where the ACTL-SLOW concepts are being tried.

Engineering GUI prototypes

The ACTL team has prototyped Java based Engineering GUI technologies within the ACS environment. These GUIs have been applied in real operation in both the MST and ASTRI prototype control subsystems.



```
Command Line: get.azsto
TCU.GET.AZSTOENPOS
THCU.GET.AZSTOEX
THCU.GET.AZSTOEN
TCU.GET.AZSTOPRE
THCU.GET.AZSTOIN
TCU.GET.AZSTOPREPOS
```

Site informations

Latitude (deg)	37,693281
Longitude (deg)	14.974676
Altitude (m)	1739

IERS data

X_PM (arcsec)	0
Y_PM (arcsec)	0
DUT1 (s)	0
TAI_UTC (s)	0

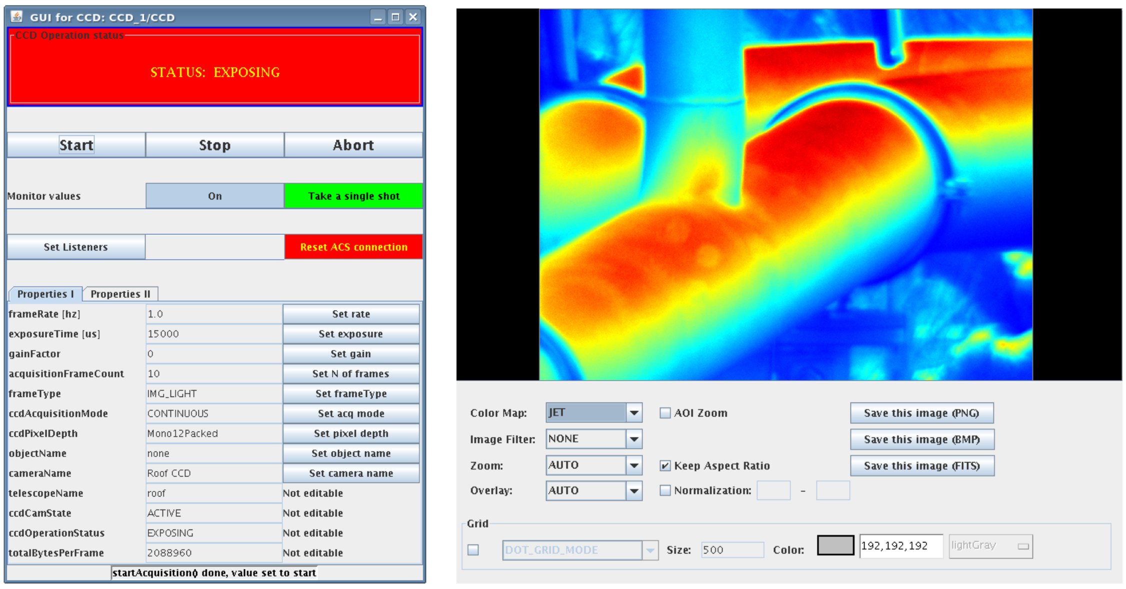
Telescope position

RA	0		
DEC	0		
AZ	0		
EL	0		
Actual AZ	0	Actual AZ Vel	0
Actual EL	0	Actual EL Vel	0

Velocità	0.99	Velocità	0.67
Accelerazione	1.47	Accelerazione	0.49
Decelerazione	0.88	Decelerazione	0.50
Jerk	0.91	Jerk	1.8

Figure 3.23 – Java FX based engineering GUIs, used to control the ASTRI prototype. The figure shows, from top left and counterclock wise, the subsystem switches, interlock switch statuses, command line interface and log console, mount (drive system) motion control and main panel GUIs.

JavaFX based prototypes A set of GUIs based on JavaFX has been developed and tested in the ASTRI telescope prototype (see Fig. 3.23). These GUIs are composed of three separate layers that follow the Model-View-Controller (MVC) paradigm. In order to reflect the subsystems and to simplify user operations the application is split into panels that provide input and output controllers (e.g. monitoring current phases in one panel or turn on and set the motion speed of the drivers). The GUI can monitor automatically required information and it can display charts from numerical values on request. An integrated Command Line Interface (CLI) gives access to all the commands described in the device ICDs. These GUIs use code that has been automatically generated from the device ICDs.



frameRate [hz]	1.0	Set rate
exposureTime [us]	15000	Set exposure
gainFactor	0	Set gain
acquisitionFrameCount	10	Set N of frames
frameType	IMG_LIGHT	Set frameType
ccdAcquisitionMode	CONTINUOUS	Set acq mode
ccdPixelDepth	Mono12Packed	Set pixel depth
objectName	none	Set object name
cameraName	Roof CCD	Set camera name
telescopeName	roof	Not editable
ccdCamState	ACTIVE	Not editable
ccdOperationStatus	EXPOSING	Not editable
totalBytesPerFrame	2088960	Not editable

startAcquisition() done, value set to start

Color Map: JET
Image Filter: NONE
Zoom: AUTO
Overlay: AUTO
AOI Zoom
[✓] Keep Aspect Ratio
Normalization: –
Grid
DOT_GRID_MODE Size: 500 Color: 192,192,192 lightGray

Figure 3.24 – Java Swing based engineering GUIs for the control of AVT CCDs . *Left:* Panel to control the configuration of and acquisition of data from the device.*Right:* CCD image display including image processing and storage options. In this example the image of a nearby pipe has been acquired and a false color scale applied.

Java Swing based prototypes An engineering GUI has been developed to simultaneously operate Allied Vision Technologies (AVT) CCD cameras such as those used in the MST prototype (3 cameras are used). As shown in Fig. 3.24, one panel provides full control of the configuration of and acquisition of data from the CCD cameras and in another it allows the display and filtering in real-time of the acquired images, as well the selection and storage of images. An integrated console provides feedback of ongoing operation and operation errors. This tool is also being used by LST team members to evaluate a CCD camera model for their telescope.

3.4.2 ACTL-OPS: Instrument operation software

In this section the prototypes associated with ACTL-OPS (Instrument Operation Software, see Sec. 3.2.2) numbered *3.2* in the CTA PBS structure, are described. See also Sec. 3.4.7, where telescope prototypes, the main testing ground of ACTL-OPS, are described.

Instrument monitoring data

With the large number of instrument sensors, status flags and operation setup values (*monitor* and *control points*, see Sec. 3.2.2), it is important to select the correct tools to acquire, store and later efficiently retrieve such data. Therefore prototyping and testing activity require both the selection and development of the tools to perform the acquisition of the data (front-end), and also the assessment of the adequate back-end database (technology and organization of the data) to efficiently store and recover the data. It is important to highlight that one uncertainty affecting this task (see Sect. 5.7) is that there are not yet clear specifications of the data to be monitored (type, volume, rate, monitoring trigger conditions etc.), thus a best guess, but conservative approach, is being followed in the ongoing tests.

Two tools are being prototyped and tested for the front-end (data gathering) mechanism:

Blobber-Collector This is the mechanism provided by ACS and used by ALMA to collect and store data from about order 200,000 monitor points that are sampled at typical rates of 1-2 Hz. Adaptations of the blobber are currently being used for the ASTRI CTA prototype. A plugin has been created to take a blob of data, transform it into a two column array with time and value of the corresponding property and write it into a mysql database. Current activity is focused in prototyping a way to extract and write all the monitoring information needed by the analysis pipeline into a suitable FITS file. In addition, large scale tests will be performed in the ACTL test cluster (see Sec. 3.4.5), including the integration of various back-ends such as MongoDB and Cassandra.

Property Recorder The tool has been tested in real operation with the MST telescope prototype, and in a simple emulated environment resembling what could be expected from several telescopes, thus proving its capabilities during the operation of the CTA prototypes or initial operation of a reduced array. Large scale tests will be performed with the ACTL test cluster (see Sec. 3.4.5), however the Property Recorder is kept as a backup solution for the Blobber-Collector.

MongoDB and Cassandra are currently considered as the most promising candidates for a back-end (storage) mechanism for the instrument monitoring archive.

MongoDB is being used as the standard back-end for the *property recorder* in the MST Prototype, and is also planned to be tested as one of the back-ends for the *blobber-collector*. A test program is scheduled to assess the performance of MongoDB as a shared database/collection, which is explained schematically in Fig. 3.25.

The first phase of the test program will create a CTA-like setup, which will generate data from 150.000 monitoring points with sampling rates from 1 to 0.05 Hz. Two data models will be tested (so-called *chunk-*

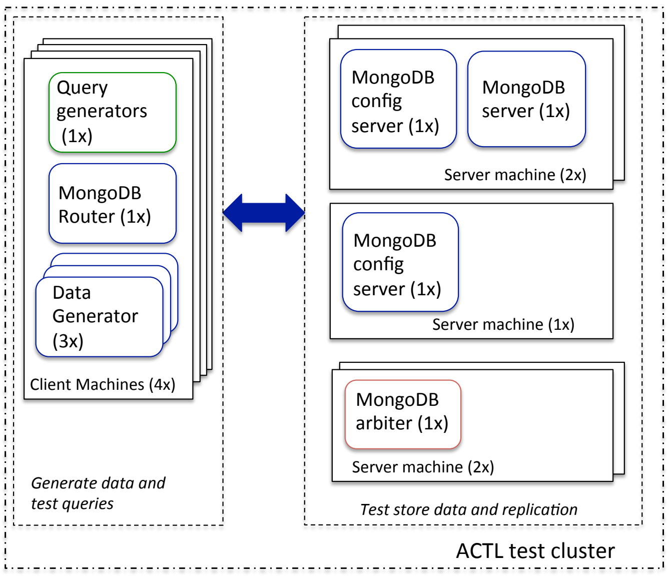


Figure 3.25 – Diagram showing a schematic view of the test program to assess the performance of MongoDB as monitoring repository backend technology. In the first phase of the test, mock monitoring data are generated in the *test machines* and stored in the MongoDB servers, testing the data ingestion capability of the technology. The usage of *arbiters* allows the testing of the replication capabilities of the technology. In a second phase of the test more realistic behaviour will be added by using simultaneous *query generators* that will access the database while data are being stored.

oriented and *day-oriented* models). Data volumes equivalent to 30 and 60 days of full CTA operation will be generated in order to test the data ingest and the post-facto querying performance. The second phase of the test program will comprise the generation of mock data and the execution of stress tests under more realistic conditions (e.g. adding load to the database by performing concurrent queries).

Cassandra A similar test program to the one described above for MongoDB will be performed for the Cassandra technology by the ALMA team in cooperation with ACTL personnel, in order to see whether a common approach can be used.

Interface for the On-site Configuration database

Testing the TMCDB-Explorer for CTA This application provides a graphical user interface for the population and modification of configurations in the ALMA TMCDB (see Sec. 3.2.2), and could be well-suited for the *CTA On-site Configuration database*. Written in Java, it is built upon the open-source Eclipse Rich Client Platform (RCP). Part of it is usable as-is, without further modification, for modifying software configurations (*e.g.*, deployment of ACS components and containers and specification of device characteristics such as alarms and monitoring rates). The hardware part, being ALMA-specific, needs to be cleanly separated from the software configuration-related code and then replaced by its CTA analog. The plug-in structure of an RCP application is designed to facilitate this kind of separation, and the ASTRI ACTL team is working with the help of the ALMA Control Subsystem leader to implemement and test it.

Operator logbook prototypes

The ALMA Shift Log Tool software application, currently being adapted to evaluate its usage in CTA by using it in the ASTRI prototype, is described in its manual as follows:

> The Shift Log Tool (SLT) allows its users to record all day- and nighttime observing activities. That includes a detailed list of all science observations, information about the environmental conditions, logs of technical operations carried out on the [telescopes] ..., and the

list of calibrations performed for a given day. The SLT is the main repository of information for observing statistics, telescope mode usage, downtime breakdown, etc. The tool can also produce a series of reports for any given time interval (e.g. 24 hours) or project, or more.

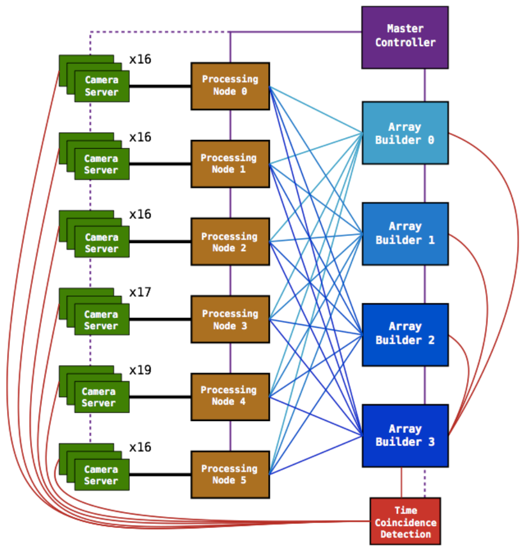
The tool's GUI provides access to its functionalities in an intuitive and user- friendly way. In particular, it provides (whenever possible) meaningful default values for entry fields; the user has, however, the possibility to override such defaults.

This tool is implemented in both desktop and Web-application versions. Its design has been refined in response to years of use by and feedback from ALMA astronomers and operators, and we have receive some help from its author in understanding the steps we need to take to adapt it to the needs of ACTL.

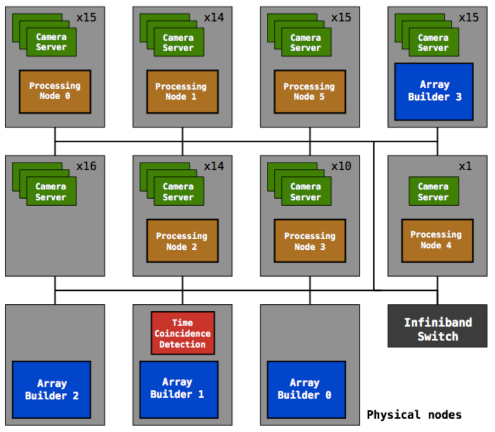
3.4.3 ACTL-DAQ: Data Acquisition Software

In this section the propotypes associated with ACTL-DAQ (Data Acquisition Software, see Sec. 3.2.3) numbered *3.3* in the CTA PBS structure, are described.

To validate the DAQ architecture concept (see Sec. 3.2.3 and [16]), early prototyping tested the distribution and merging of data streams coming from 100 simulated camera servers. The camera servers produced events at a predefined rate, which were sent to a dummy processing node. The dummy processing only converted the output format of the camera servers to the on-line, unified event format. Array builders polled the processing nodes based on the time coincidence information to reconstruct array events. The time coincidence module simply triggered every event generated by the camera servers. The logical setup can be seen in Fig. 3.26, left.



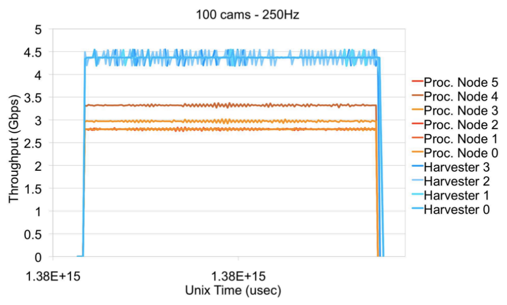
(a) Figure A



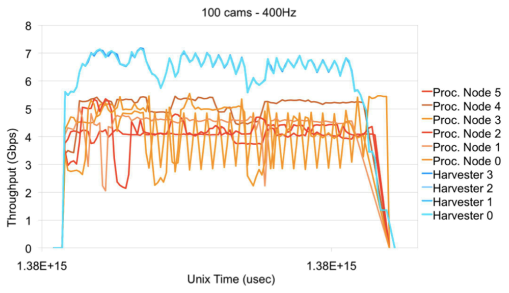
(b) Figure B

Figure 3.26 – *Left:* Performances test logical setup. 100 camera servers generate dummy events which are triggered by the dummy time-coincidence detection. The events are forwarded to dummy processing nodes, which buffer them until they are polled by the array-builders based on time coincidence information.*Right:* Physical layout of the components on the physical compute nodes at ISDC. Each node has 16 cores, 64 GB of RAM and they are interconnected through an Infiniband switch with a maximum throughput of 56 Gbps.

This setup was physically deployed on the ISDC cluster, as seen on Fig. 3.26. The production rate of events was limited to 250 and 400 Hz due the throughput limitation of the cluster. Higher data rates would be no problem, as long as the network supports them. The test required that each camera generate 200k events to be assembled into array events. A total of 1747.4 GB of data was successfully assembled into array events with no event loss. The results can be seen on Fig. 3.27.
This approach seems to be suitable for CTA as 90% of the network capability was utilized. The use of TCP/IP sockets limits the throughput per stream to approx. 10 Gbps. This comes from the time-of-



(a) Figure A



(b) Figure B

Figure 3.27 – *Left:* Throughput per component for an event generation rate of 250 Hz. The total data throughput is 18 Gbps which translates to 36 Gbps through the network as the events must be sent twice: First from camera servers to processing nodes, and then from processing nodes to harvesters. The rates are very stable and represent 63% of the max. theoretical throughput of the network. *Right:* Throughput per component for an event generation rate of 400 Hz. The total throughput is 58 Gbps which represents 90% of the theoretical network capability. The rates are not stable any longer due to a higher event production rate than what the network can really transport. This is not an issue as long as enough buffer space is available on the compute nodes. In this stress test, the event production would have blocked if the buffering capabilities of the nodes had been exceeded.

flight of the network packets between two nodes, and can vary depending on the packet size and latency between two peers. In case more than 10 Gbps is required for a single stream, several TCP connections will be used to load-balance the traffic.

A second prototype was developed, this time using both ZeroMQ and the protocol buffers. A monitoring tool was also created to enable the user to monitor the pipeline's activity, as seen on figure 3.28. Due to time constraints, this prototype focused on load-balancing and automatic failure recovery. The maximum throughput was tested between two nodes only and reached approximately 8 Gbps. This figure is between 10 and 20% lower than when using raw tcp sockets, which should not be an issue as more than one stream can easily be created to accommodate higher throughput. Moreover, the prototype used sub-optimal code which will be improved by removing intermediate copies of the data. This approach is currently being discussed with the camera work packages to estimate whether it would be suitable to readout the data from the camera servers, and if so what the required optimisations and specific constraints would be. The small-size telescopes will be no problem as the data rate remains low, however for MST and LST cameras the high throughput will most likely prevent the DAQ API from performing on-the-fly data conversion, thus delaying the unification of the data format to a later stage in the acquisition pipeline.

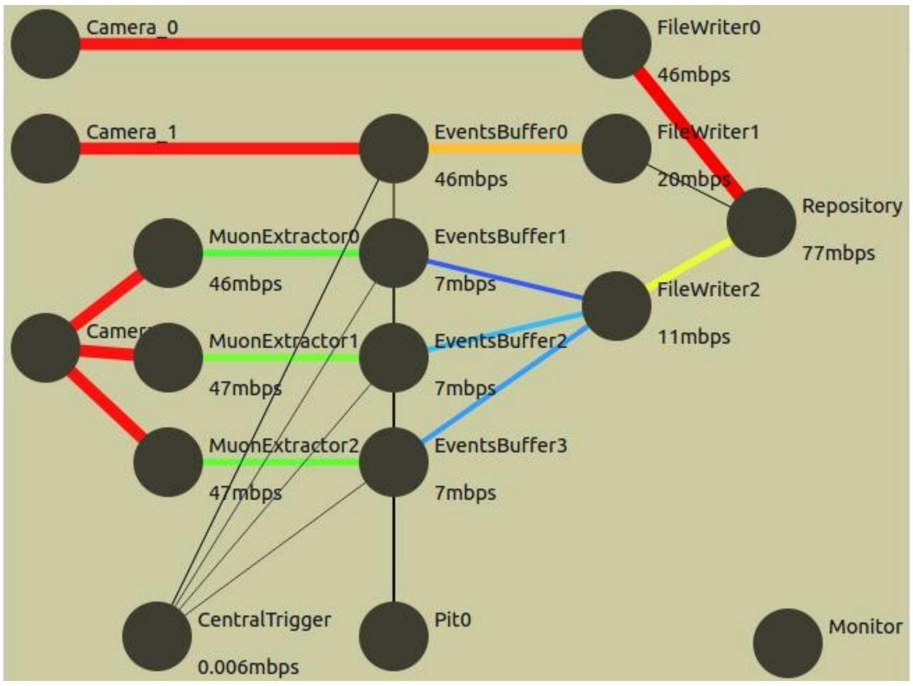


Figure 3.28 – Screenshot of the monitor display from the latest DAQ prototype. Each circle represents a DAQ component while the lines connecting them are actual ZeroMQ connections. The color of the line reflects the average throughput for each connection. Due to the use of ZeroMQ and the protocol buffers, new components can be added or removed without stopping operations. This particular view shows the data taking from 3 cameras. The first one has all its events written to disk, the second camera pipeline discards events based on their stereo coincidence while the third camera pipeline has an image cleaner and muon extractor which analyses the data in real-time. The EventsBuffer nodes retain the events until their stereo coincidence is known, while the FileWriters compress and write the events to disk in compressed FITS format. The node called Pit0 is a proxy that only receives events classified as muons for monitoring purposes.

3.4.4 ACTL-TRIG: Array Trigger and Clock Distribution

In this section the propotypes associated with ACTL-TRIG (Array Trigger and Clock Distribution, see Sec. 3.2.4) numbered *3.4* in the CTA PBS structure, are described. These include prototypes for both Clock Distribution (ACTL-TRIG-CDTS, PBS item *3.4.2*) and Array Trigger products (ACTL-TRIG-AT PBS item *ID 3.4.1*).

Clock Distribution Prototypes

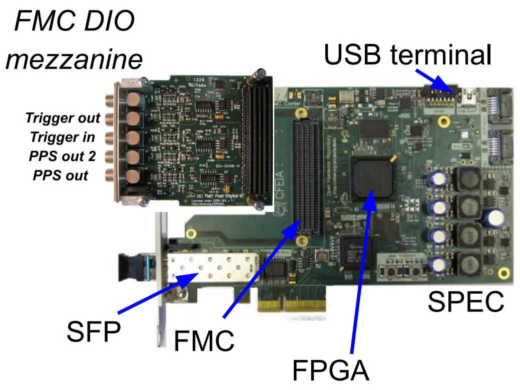
The HiSCORE [24, 25, 26] experiment is already using White Rabbit for time stamping of cosmic ray events and a prototype setup for the CTA implementation has been tested extensively and is described below together with a prototype setup used for the CHEC-M prototype camera.

HiSCORE A non-imaging Cherenkov air shower experiment to detect gamma-ray and cosmic-ray shower, HiSCORE, is under construction at the Tunka-site in Siberia. Since directional air shower reconstruction in HiSCORE is based on precision timing of the Cherenkov light front at the detector stations distributed over several square kilometers, the experiment has essentially the same trigger timestamping requirements as CTA and provides an excellent testbed for White Rabbit technology. HiSCORE uses a White Rabbit network architecture similar to that proposed for CTA—two White Rabbit switch layers to connect up to 50-100 nodes. The switches were sourced from the company Seven Solutions [30]. HiSCORE's choice of the White Rabbit node is a commercial device that is the default choice for most White Rabbit applications,the White Rabbit simple PCIE FMC carrier (SPEC) card. The SPEC card is interfaced with the detector components by the commercially-sourced 5-channel digital Input/Output mezzanine card. Both the mezzanine card and the SPEC card were sourced from two vendors, Seven Solutions and the Polish company Creotec [31]. The two vendors provided devices with comparable performance. The firmware on the SPEC card was extended by the DESY-Zeuthen group to allow:

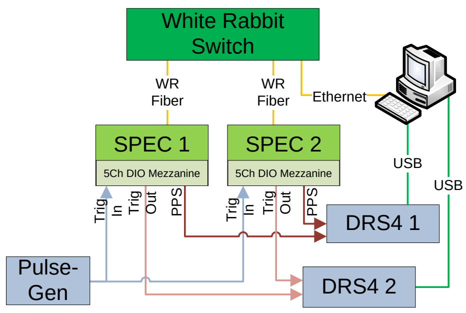
- Nanosecond precision trigger timestamping
- Generation and transfer of a UDP trigger message, e.g. counter, timestamps, flags
- Specific trigger functionality, e.g. busy signals and readout control.

Continuous monitoring of the White Rabbit synchronization status was done by EtherBone-driven readout of White Rabbit status registers, and via the SPEC USB-terminal port [26, 25]. The combination of a customized FMC card plus the extended functionality of the White Rabbit SPEC card used by HiSCORE would meet all CTA requirements for a UCTS board.

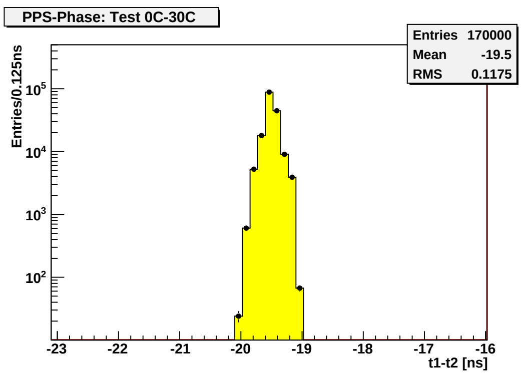
Data from HiSCORE includes initial White Rabbit laboratory and field tests and data from a 9 station prototype array that has been operating since Q3/4 2013 [26]. Laboratory tests with a dedicated climate chamber setup were done to check the relative White Rabbit node clock stability, i.e. the clock jitter, by monitoring the PPS-outputs of two SPEC cards with a $5\,\mathrm{GHz}$ sampling rate and the performance of the nanosecond precision timestamping of trigger signals as shown in Fig. 3.29. The two SPEC cards are connected to a PC via a White Rabbit switch. The PC receives the trigger messages generated by the SPEC cards which timestamp an incoming trigger pulse of a signal generator. Moreover, the status of the SPEC cards is logged by the PC as well. The clock jitter of the White Rabbit nodes is measured by a DRS4 $5\,\mathrm{GHz}$ sampling board. The results of a representative two-day climate chamber test, in which one of the SPEC cards and its $520\,\mathrm{m}$ optical fibre were exposed to temperature changes between $0\,^{\circ}\mathrm{C}$ and $30\,^{\circ}\mathrm{C}$, are also shown in Fig. 3.29. The resulting RMS of the clock jitter is $120\,\mathrm{ps}$, a value in excellent agreement with the known White Rabbit capabilities. Furthermore, the difference between the trigger timestamps generated by the two SPEC cards does not exceed $1\,\mathrm{ns}$ which is well within CTA requirements for inter-telescope timing. Note that the resolution of the measurement was only $1\,\mathrm{ns}$ and that the offset between the different clocks is due to different cable lengths in the setup. The laboratory results were confirmed by field measurements at the Tunka array [25].



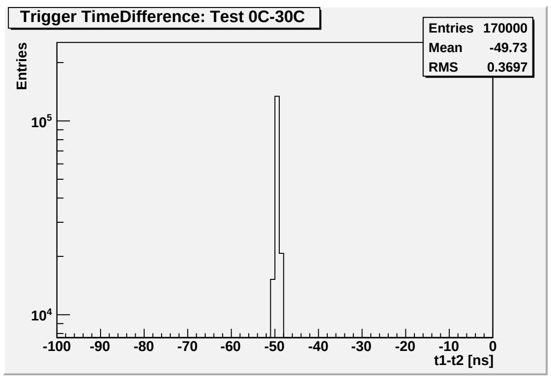
(a) White Rabbit SPEC & FMC cards



(b) HiSCORE White Rabbit test setup



(c) Clock jitter measurement



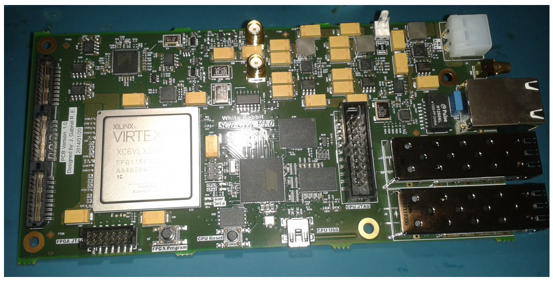
(d) Trigger timestamp difference measurement

Figure 3.29 – On the upper left, the White Rabbit SPEC card with a 5 channel DIO FMC extension card, which is used by HiSCORE in stand alone mode, is shown. On the upper right, the HiSCORE measurement setup to verify the stability of the PPS-phase and trigger timestamping for two SPEC cards connected via $20\,\mathrm{m}$ and $520\,\mathrm{m}$ of optical fibre is depicted. The tests discussed here lasted for $48\,\mathrm{h}$ and were done with SPEC card 2 and its corresponding $520\,\mathrm{m}$ located inside a climate chamber. They were subject to a series of temperature variations between $0\,^{\circ}\mathrm{C}$ and $30\,^{\circ}\mathrm{C}$. In the lower left the distribution of the jitter between the PPS-clock-pulses from the two SPEC cards is shown and on the lower right the distribution of time differences between measured trigger time stamps with a $1\,\mathrm{ns}$ resolution.

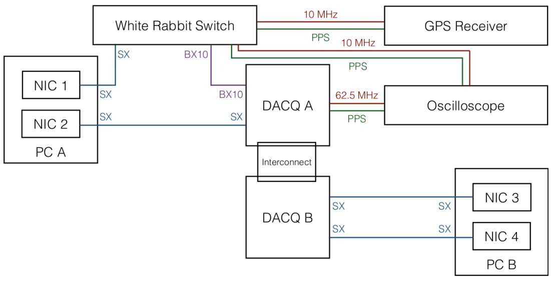
The reliability of the White Rabbit components (SPEC cards and White Rabbit switches) has been demonstrated by their operation for almost two full data seasons at Tunka without any major problems. No broken or malfunctioning units have been found out of two switches and 11 SPEC cards exposed to operating temperature variations between $+10\,^{\circ}\mathrm{C}$ and $+50\,^{\circ}\mathrm{C}$ and storage temperature variations between $-45\,^{\circ}\mathrm{C}$ and $+40\,^{\circ}\mathrm{C}$.

CHEC-M prototype For the CHEC-M camera prototype for a small-size telescope (SST-2M), a custom version of a White Rabbit switch was selected to act as the data acquisition (DACQ) board. This board routes the bulk data coming from the front-end modules to the outside world and timestamps events for the prototype camera. A picture of one of the boards can be seen in Fig. 3.30.

In the tests discussed here, the traffic that would normally be generated by the front-end modules was mimicked using the *iperf* tool, which can create TCP and UDP traffic at arbitrary rates. A LeCroy Waverunner oscilloscope is used to monitor the clock precision and jitter for the PPS signals as well as the $62.5\,\mathrm{MHz}$ clock. The two small form-factor pluggable (SFP) transceivers are used bidirectionally, i.e. sending and receiving at the same time at a rate of approximately 880 Mb/s each resulting in a total throughput of 3.52 Gb/s. The result of a $47.1\,\mathrm{h}$ test is a clock offset of $-3.93\,\mathrm{ps}$ and a jitter of $14.91\,\mathrm{ps}$ which are in perfect agreement with the known capabilities of a White Rabbit distributed clock. Moreover, the CTA requirements for inter-telescope timing are also satisfied. Loss of synchronization (the critical failure mode for a White Rabbit system) was observed in the lab setup under two conditions. The first is



(a) CHEC DACQ Board



(b) Lab Prototype Test Setup

Figure 3.30 – On the left a photograph of a CHEC-M DACQ board is shown; in essence the core board of a normal White Rabbit switch. On the right, the test setup for the DACQ boards of the CHEC-M prototype camera is depicted. The two DACQ boards are interconnected and traffic is generated using two PCs which are connected via the DACQ boards. The White Rabbit clock is supplied by a White Rabbit switch running in *GrandMaster* mode while a GPS receiver provides the input clock. A $10\,\mathrm{km}$ optical fibre between DACQ board A and the White Rabbit switch is used as well.

due to a known bug that has been fixed in later revisions of the hardware. In this case synchronization was lost approximately once every $72\,\mathrm{h}$ for a period of no more than a few seconds and then immediately re-established. During this period the clock (and therefore the timestamps) could be off by several tens of nanoseconds). Overloading a White Rabbit port with UDP traffic above 880 MB/s will also result in a loss of synchronization. Synchronization is re-established once the rate drops below 880 MB/s. Lab tests also confirm that the DACQ boards support jumbo frames up to $6500\,\mathrm{b}$.

Array Trigger Prototype

SWAT prototype A prototype software-based system for the ordering and association of timestamps has been developed and tested [27]. The SWAT prototype is a multi-threaded process written in C. It uses Linux-specific epoll() system calls to efficiently manage hundreds of simultaneous TCP network connections. To achieve the required performance, all the main operations on data streams (data reception, data merging, data sorting and array trigger detection) were performed in parallel by dedicated threads (see Fig. 3.31). The scheme used here (as shown in figure 3.31) is one in which the camera servers directly provide the trigger input records in addition to receiving and processing array triggers.

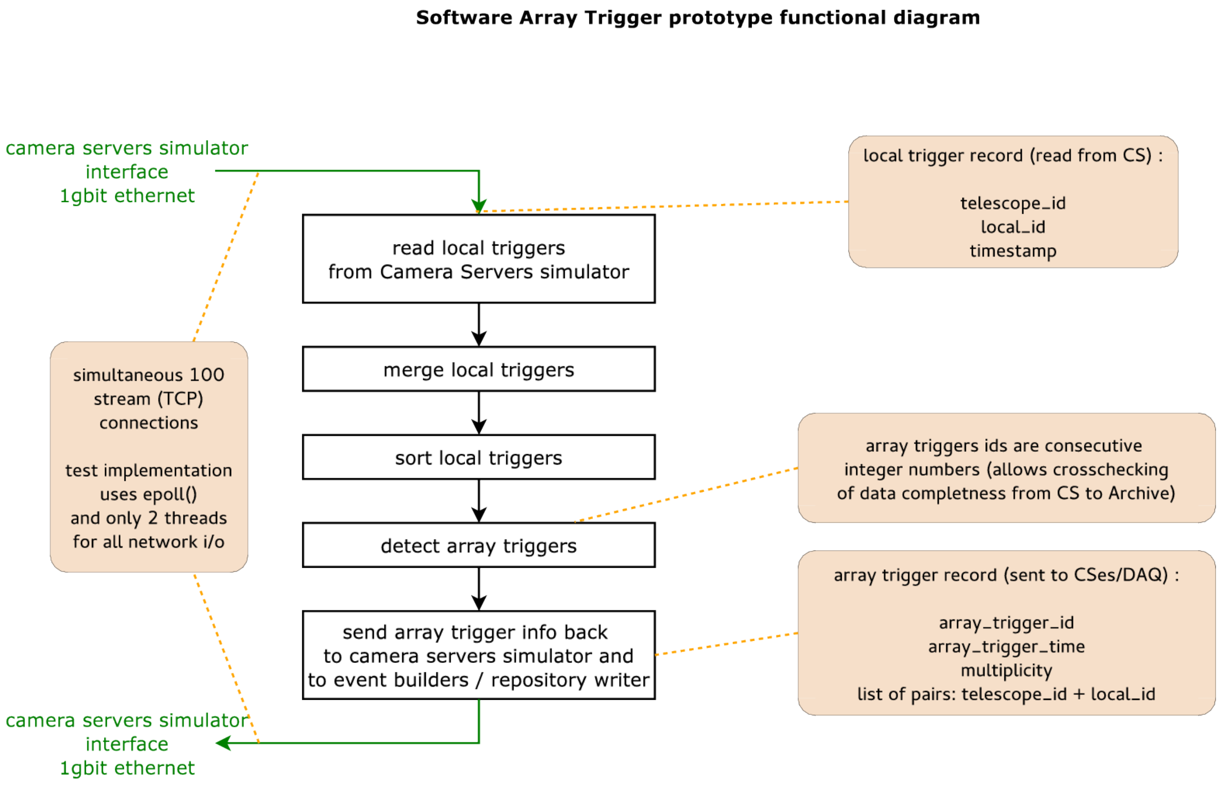

Software Array Trigger prototype functional diagram
camera servers simulator interface 1gbit ethernet
read local triggers from Camera Servers simulator
merge local triggers
sort local triggers
detect array triggers
send array trigger info back to camera servers simulator and to event builders / repository writer
camera servers simulator interface 1gbit ethernet
local trigger record (read from CS) : telescope_id local_id timestamp
simultaneous 100 stream (TCP) connections
test implementation uses epoll() and only 2 threads for all network i/o
array triggers ids are consecutive integer numbers (allows crosschecking of data completness from CS to Archive)
array trigger record (sent to CSes/DAQ) : array_trigger_id array_trigger_time multiplicity list of pairs: telescope_id + local_id

Figure 3.31 – SWAT prototype functional diagram.

The camera server was replaced in these tests by a simple emulator. This emulator generates trigger input records at a specified rate, according to Poisson statistics, and sends them to the SWAT prototype. The emulator waits and packages the data into the largest packets possible (Ethernet: 9KB, Infiniband: 64KB) before shipping them to the SWAT over the network. The standard Transmission Carrier Protocol (TCP) was used to communicate between the camera server and the SWAT. The data format was defined *ad-hoc*.

A worst-case scenario with a single array of 100 telescopes was chosen to stress-test the system. Ten simultaneous instances of the camera server emulator were run on each of ten dual Xeon X5650 CPU servers for a total of 100 instances. These machines were interconnected with both QDR Infiniband and Ethernet and had 48 GB of ECC RAM and PCI Express Gen2 network cards. The operating system used was a standard Scientific Linux 6.x 64-bit distribution. An eleventh Prototype SWAT daemon ran on a dedicated server with dual Xeon E5-2665 CPUs, 64 GB ECC RAM, and a Gen3 FDR Infiniband card running in QDR mode. All eleven servers were rack-mounted, were remotely controlled via IPMI, and had redundant power supplies and hard disk drives. Iterations of the stress-test were run using both standard 1.5K and jumbo 9K ethernet frames and both normal Ethernet and Infiniband network connections.

The results are summarized below :

- The SWAT prototype successfully processed trigger input records delivered at rates of ~ 2 MHz, with the maximum array trigger rate achieved for the minimum array trigger requirement of two coincident telescopes. Coincidence windows used ranged from 50 nanoseconds to 10 microseconds. This processing rate exceeds the basic CTA requirement B-ACTL-0180.

- The system was capable of generating array triggers at rates up to 200-400 kHz, with the precise rate again depending on the multiplicity requirement. This exceeds the basic CTA requirement B-ACTL-0190. Beyond this rate threshold the SWAT's efficiency would drop due to failure to detect coincidences within the 1 second time limit and due to losses of events due to buffer overflows. The precise fraction of events lost under these circumstances has not been established.
- The prototype can tolerate network (aggregated) data rates up to 50–80 MB/s. This number assumes 16 bytes/trigger input record, 2–4M events per second, and events aggregated in the largest possible packet. In this case "largest possible" corresponds to 9 kB for a standard Ethernet connection (most closely analogous to the White Rabbit Ethernet connection that transmits these packets in the actual system) and 64 KB for an Infiniband connection. This amount of steady traffic can be accommodated by a single GB/s port on a White Rabbit Ethernet switch, but leaves scant room for rate spikes. Splitting the traffic over multiple port would alleviate this issue.
- The load is expected to be ~ 3/CPU with CPU core utilization at 100%, with the limiting factor being the I/O subsystem, regardless of whether Ethernet or Infiniband connections are used. The SWAT processing algorithm contributes 50% to this total load. The remainder comes from operating system (OS) overhead due to frequent context switching. The precise load values depend on the quality of the network cards and driver used. The quoted numbers are for server-class Mellanox cards. Binding the CPU and network card to specific memory chips on the motherboard via the *numactl* program could improve the performance further but this was not tested. The primary takeaway from these numbers is that we should aggregate data transmitted over the network so that we transmit the largest possible packet size as infrequently as possible.
- The prototype latency, measured as the time from receipt of trigger input records to the production of a trigger, is of order 1 second and tuneable above 0.2 seconds. An optimal value would be on the order of a second. The maximum value is constrained not by the SWAT prototype but by the depth of the camera server buffer. As the camera server emulators were only used to simulate inbound connections to the SWAT, this aspect of the problem has not been tested in real time.

3.4.5 ACTL-ONSITE: On-site ICT Infrastructure

In this section the propotypes associated ACTL-ONSITE (On-site ICT Infrastructure, see Sec. 3.2.5) numbered *3.5* in the CTA PBS structure, are described.

Computing cluster for ACTL software tests

An ACTL testbed has been deployed in the beginning of the year 2015 by Humboldt University and DESY. The aim of a testbed cluster is to setup a prototype of the on-site computing facility in terms of storage, computing and networking to test the design of the hardware and software, especially the installation, the services and configuration and change management. This is foreseen to further evolve into an environment used for software development, software validation and verification as well as full system integration.

Initially, the testbed includes 10 computing nodes and about 5 data servers for database studies; the addition of 10 more nodes is anticipated for the end of the year 2015. All nodes are connected via a network setup resembling, on a smaller scale, the setup to be deployed in CTA. The setup allows us to perform these tests (the list is not exhaustive):

- Large scale tests of ACTL software emulating the array operation and control.
- Test for integration of the software provided by the device teams into the ACTL control software.
- Bandwidth tests of the bulk data emulating telescope Cherenkov cameras

- Emulation of alarm and error recovery tools.
- Test of the performance of the instrument monitoring data and the querying of large volumes of stored data.
- Verification and validation of ACTL software products.
- Test of the performance of the level-A and level-B analyses.

This testing ground is open for any ACTL activity. It is also be available for full system integration tests with DATA.

3.4.6 ACTL-SCHED: Central Scheduler

In this section the propotypes associated with ACTL-SCHED (Central Scheduler, see Sec. 3.2.6) numbered *3.6* in the CTA PBS structure, are described (see [32] for more details).

Time scheduling of tasks (scientific proposals or SBs) to resources (telescopes/sub-arrays) is a well-known mathematical problem in AI engineering. Applying AI solutions to a ground-based astronomical facility must consider the rapid reaction to changing conditions without losing the long-term perspective. Different scheduling solutions and a reliable model to predict the environment and system conditions could be used. It is, therefore, important to select the correct tools to obtain an optimum schedule and to measure the performance. Therefore prototyping and testing activities are necessary to assess the adequate technology solutions that best fit the requirements for CTA measured in terms of operation efficiency and maximum scientific return.

LTP, STP and simulation scheduler prototypes

The SCTA prototype includes the LTP and STP algorithms, a simulator and an analysis platform to validate the obtained performance. It implements a preliminary interface with the array control software, but it does not include the interface with the off-site software infrastructure. Therefore, the complete data flow cannot be reproduced (see Sec. 3.2.6). However, most of the expected features can be tested.

SCTA prototype architecture A modular and highly configurable architecture is defined for the SCTA prototype in order to be easily modifiable in case the scheduler software requires a significant change in the implementation phase. Figure 3.32 shows the architecture of the developed prototype, including the ACS software infrastructure layer for subsystem communication and a detailed view of the Scheduler subsystem. The most relevant components of the scheduler are:

- The **communication interface** guarantees the exchange of data between the scheduler and the array control software and is based on the ALMA Common Software (ACS).
- The LTP and STP algorithms are inlcuded in the **Scheduler** subsystem. These algorithms were developed based on a preliminary assessment of AI technology [28]. Both algorithms are based on the well-known Guarded Discrete Stochastic Neural (GDSN) Network algorithm, and Constraint Propagation techniques. The STP algorithm is prepared to deal with the events given in Tab. 3.7 during the computation process of new schedules. These events help adapt the schedule to changing conditions. The Scheduler subsystem communicates with a Simulator or, alternatively, with the real operation control system using an ACS component (called CTA Scheduling in Fig. 3.32). The scheduling process starts by retrieving the scientific proposals from an internal data base, computing the schedule and returning it back to the Simulator for its execution. Schedule is returned with the list of tasks, and a starting time and the required configuration for each task execution. Other modules in the Scheduler subsystem are defined for different purposes: GDSN is devoted to implementing the scheduling algorithm (a different algorithm may be used by just adding a similar component and reconfiguring the application accordingly); Events module (see Tab. 3.7) provides

thread-safe event handling capabilities and allows asynchronous communications between the Simulator and the Scheduler.

- The **Simulator** module is in charge of the full system simulation. It generates ad-hoc data that simulates the observatory's location, environment and weather conditions, the status and the configuration of the sub-arrays of telescopes and makes use of the definition of the scientific proposals. The Simulator communicates with the Scheduler each time an event occurs (see Tab. 3.7) and this triggers the scheduling process. Finally, the Simulator stores the outcome of the full simulation process for its further analysis.
- Several **common tools** are designed to call support libraries that provide, for instance, the computation of an astronomical ephemeris.

A database is also defined as an external entity that provides input and output data storage support to the main subsystems. A relational database, mysql, is used in the prototype. However, a different database technology, such as document-based databases, might be used if required at the ACTL or CTA system level.

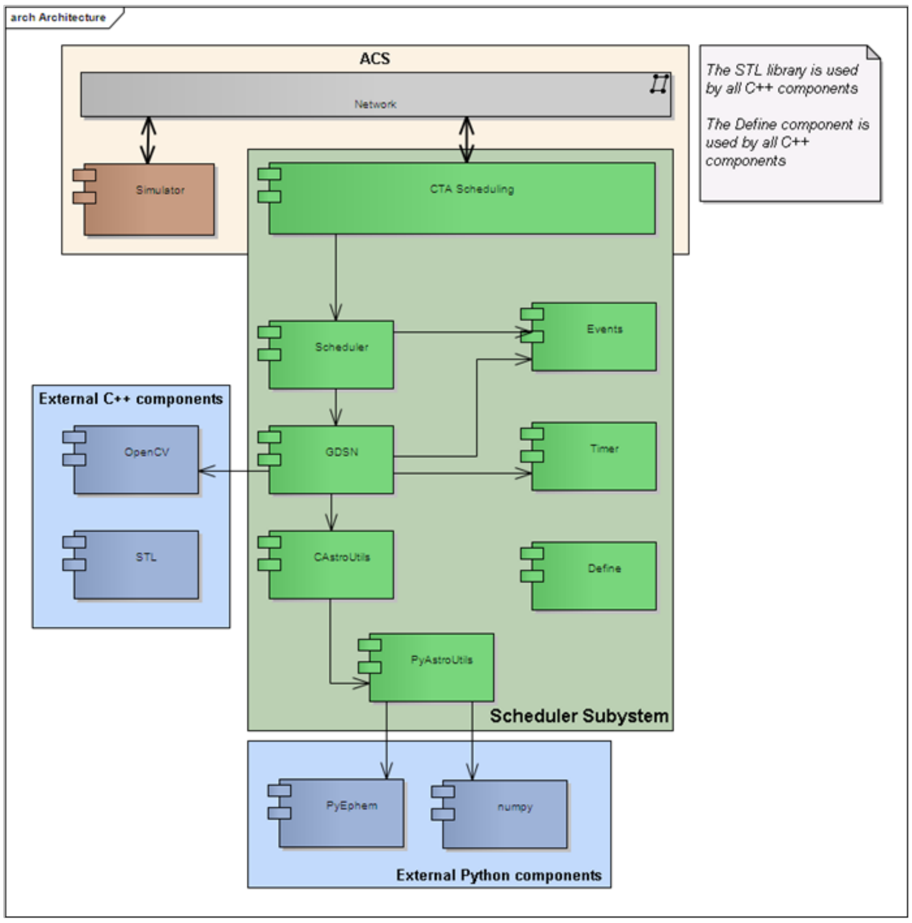


Figure 3.32 – Scheduler architecture based on two main subsystems, Scheduler and Simulator, that communicate using ACS infrastructure and use external support libraries. A detailed view of the Scheduler subsystem is shown to illustrate the modularity of the software.

Start of a Scheduling Block	When the astronomical observatory starts the observation of one target, the scheduler receives the scheduled observation period of this target and the sub-array or the main array from which it is being observed. New schedules will not contain scheduling blocks thaqt overlap with scheduling blocks that are currently being observed and, thus, they will meet hard constraints.

Abnormal termination of a Scheduling Block	When the observation of one target is aborted, data related to that period, target, and sub-array are deactivated by the algorithm to allow the target to be scheduled again in those conditions.
Completion of a Scheduling Block	When the observation of a target is finished, the scheduler receives the observed final period, which can differ from the scheduled one.
Enable/Disable sub-arrays	When a sub-array is deactivated, the algorithm may not schedule observations using the subarray within the given deactivation period.
Change of weather conditions	When the weather conditions change, the scheduler checks whether each proposal can be observed under the new weather conditions.
Science alert	When a new science alert (e.g., Target of Opportunity) arrives, the scheduler takes it into account in subsequent iterations of the algorithm in order to schedule it in the minimum required time.

Table 3.7 – Set of events used by the STP before and/or during the computation process of new schedules. The aim of the events is to let the scheduler know important changes that would modify the GDS-Network. All events are asynchronously sent to the scheduler, which in turn processes them at the start of each iteration

Performance analysis platform A performance analysis platform is developed to configure and run multiple scheduling simulations in parallel and to automatically summarize the results obtained for an easy and agile analysis. It runs some statistics and computes the metrics that help checking the quality of the scheduler performance. Figure 3.33 shows some screenshots of the platform.

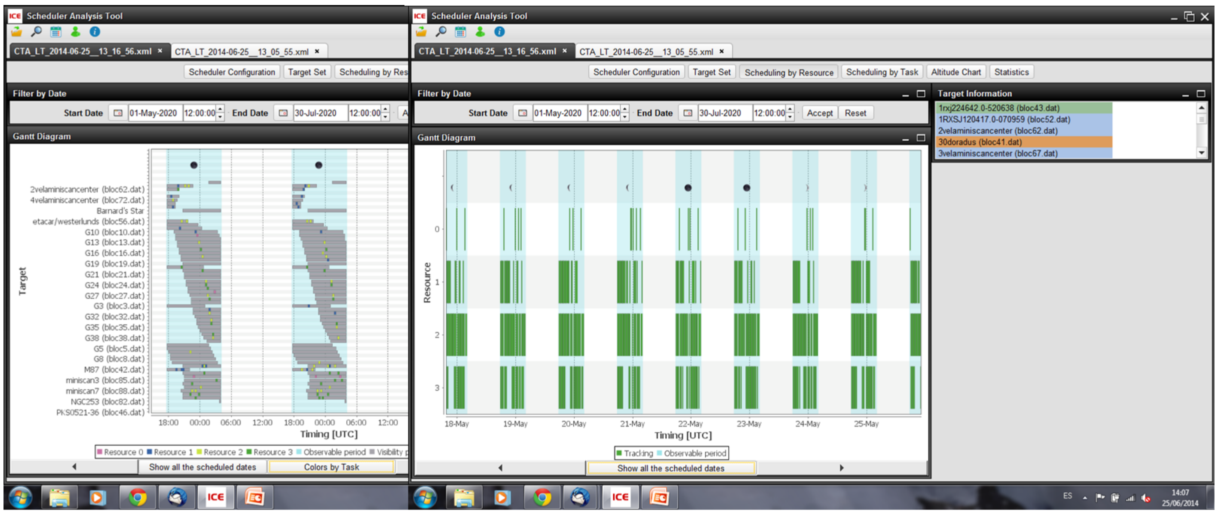


Figure 3.33 – Screenshots of the performance analysis tool used to test the Central Scheduler prototype.

Validation tests

Scheduling algorithms are generally stochastic. The test procedure is based on the implementation of a deterministic approach to ensure correctness in first results and to isolate the configuration parameters and find the optimal values for each of them. A non-deterministic algorithm is used afterwards to re-execute the test several times to detect non expected behaviour/deviations. And, finally, (non)deterministic tests are executed with nominal/expected amounts of data.

A set of tests with realistic test scenarios is defined. Several metrics are taken into account in the analysis to evaluate the scheduling performance: (i) night load metric or ratio of the total available time spent to collect data in a given observing night; (ii) optimal altitude metric that evaluates the altitude distribution for the scheduled targets; (iii) priority metric that measures the achieved global priority, in terms of the level of completion of tasks grouped by scientific priority.

As an alternative goal, a full simulation of the CTA observatory operating period can also provide valuable information to improve the observation strategy to obtain a higher scientific return. In that sense, the scheduler prototype was tested using some of the Key Science Programmes under discussion for CTA and the obtained results contributed to their definition.

Simulation with real MAGIC proposals An incremental test was defined using a reduced dataset of the MAGIC observatory and the analysis was carried out by comparing the simulated results against those obtained by hand by a MAGIC expert user, based on the same dataset and same observatory configuration. The results show that the SCTA prototype covers the majority of the scheduling requirements with very good performance and improves the number of finished proposals, even doubling the statistics retrieved when scheduling by hand. However, this significant difference in performance is caused by an incomplete model of the real system that artificially simplifies the scheduling problem: reconfiguration time between observations is underestimated, and no calibration runs are considered. An improved model is under development to increase the reliability of the test.

Simulation with CTA survey As a second test bench example, we focused on an observational program for the entire southern array. In this example, identified as a key science programme for CTA[33], a large part of the Galactic plane is planned to be observed with homogenous exposure (hereafter case A, see Fig. 3.34). Rather than a continuous scanning mode throughout an extended period of time, the observation strategy is a discrete sequence of 60 fields, separated by 2 deg in Galactic longitude, and with each single pointing assumed to span a duration of 20 minutes.

We scheduled only this observation program, a situation akin to giving it top priority. We found that the scheduler can distribute the observations so to have excellent performance as to the average observed zenith angle (ZA) in each field (see Fig. 3.35, left). All fields are observed and completed. No calibration time is scheduled in this example, but this would not impact this particular result, as the top priority associated with the observation program would trump the scheduling of calibration runs.

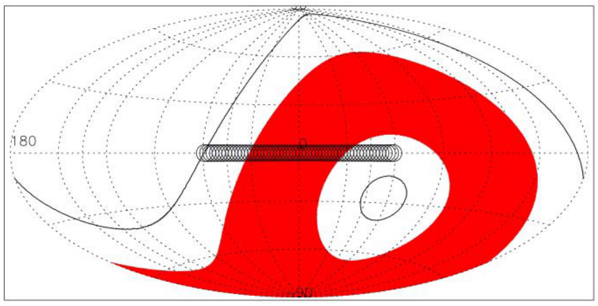
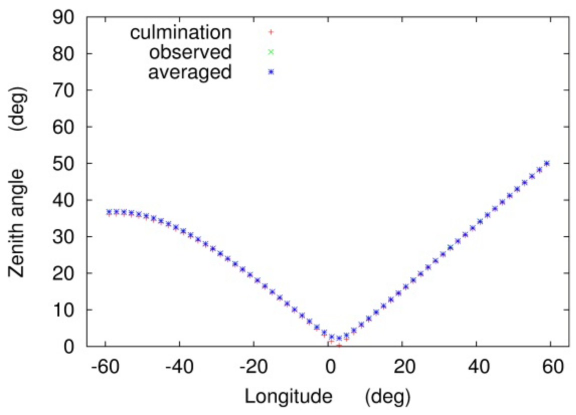


Figure 3.34 – AITOFF projection of the sky in galactic coordinates. Projection centered at (l,b)=(0,0), with longitude axis increasing towards left. The red contours limit the area in the sky which locations would culminate at ZA$<25^\circ$ when observed from the southern CTA Observatory. Black contours are for ZA$<50^\circ$. Black circles represent the fields scheduled here with $|\,l\,| < 60$ deg and b=0 deg, with a radius of 5 deg, case A.

A second test was carried out by reducing the time available in which the observation program can be allocated (hereafter referred as case B). The CTA Observatory will be an open time observatory, therefore there will be observational programs put forward by different investigators. We here mimic a situation in which half of the observation time is not accessible by the party proposing the 60 galactic fields described above. In order to do this we allow the scheduler to allocate time for our observation proposal only every second day. While the program is still completed, the ZA distribution of the observation necessarily suffers. However, such a reduction on observed ZA is acceptable (see Fig. 3.35, right).



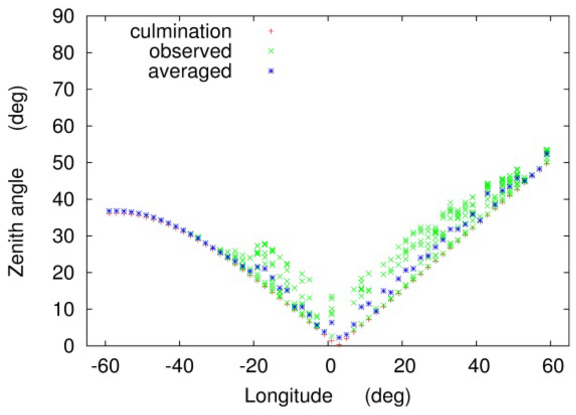
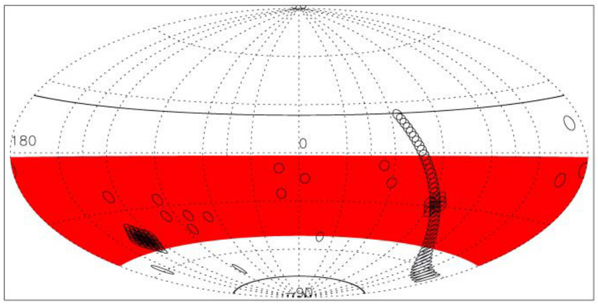


Figure 3.35 – Left: Distribution of zenith angles of the scheduled fields in case A. We show here the ZA at culmination for each field, the ZA for each pointing and the average per field. The scheduler assured optimal zenith angle condition for all fields. Right: Distribution of zenith angles of the scheduled fields in case B. We here show the ZA at culmination for each field, the ZA for each pointing and the average ZA per field. The ZA of single observations is not optimal. However the average value per field points towards a small deterioration of performance.

More complex and realistic tests were defined using a combination of proposals for Galactic and extra-galactic pointings and with an oversubscription of the available time (hereafter case C, see Fig.3.36). The scheduler prototype is not able to allocate all the requested observation time. It is to be noted that many of the requested observations are in the same RA band and, thus, the failed completion can be partially due to the RA crowding.

The results obtained with these preliminary tests show that the schedules obtained comply with the defined constraints, optimize the scientific return and can react to unexpected situations in less than one second. These promising results will have to be validated with a higher number of proposals and including calibrations tasks.



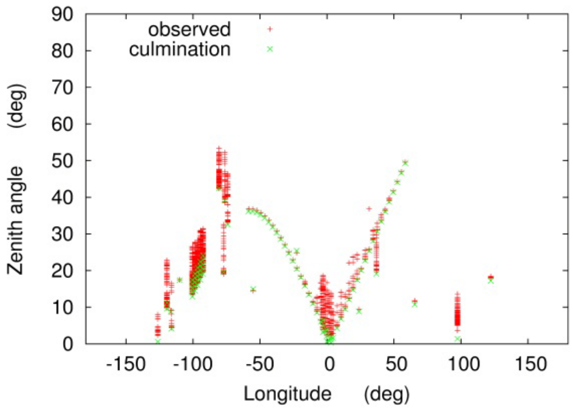


Figure 3.36 – Left: AITOFF projection of the sky in equatorial coordinates. Projection centered at (RA,DEC)=(0,0), with Right Ascension axis increasing towards left. The red contours limit the area in the sky whose locations would culminate at ZA<25 deg when observed from the southern CTA Observatory. Black contours are for ZA<50 deg. Black circles represent the fields scheduled in case C. Right: Distribution of zenith angles of the scheduled fields for case C (see text). We show here the ZA at culmination for each field and the ZA for each pointing. The ZA of single observations is not optimal but acceptable for all cases.

3.4.7 Array control prototypes associated with telescope prototype projects

The telescope prototype projects have constituted an important driving force first in the evaluation of software technologies and later in the development of solutions to various requirements for the whole CTA array control work package. For example, the on-going activities of developing tools for monitoring device status, configuring detectors, operator GUIs and standard libraries to communicate between ACS and OPC UA have been driven by these projects. Also the evaluation of ALMA software for its re-use (see Sec. 3.2.7) is being performed on prototypes, in particular in ASTRI. These prototypes are also the main testing ground for ACTL-SLOW and ACTL-OPS software products. One of the most important tests that has been carried out around these prototypes has been the successful excercise of the device integration within the OPC UA server and ACS bridge paradigm.

Table 3.8 gives an overview of the telescope prototype projects that are currently underway within CTA and are allowing in-depth exercise of the ACS and OPC UA technologies. From the ACTL point of view, the MST and ASTRI prototype projects have constituted an in-depth test of the ACS and OPC UA technologies and the device integration in such frameworks. The SST-2M-GCT project will not explore ACS but use OPC UA for the device access. An LST prototype will be built at La Palma. If La Palma is chosen as the CTA-N site, then that prototype will eventually become a CTA LST. If La Palma is not chosen, then that prototype will move to the CTA site (either north or south). The pSCT project will partially use ACS and OPC UA for certain devices, and partially use custom VERITAS solutions to steer the prototype. The SST-1M prototype will use an approach analogous to the one of MST and ASTRI. The LST telescope project is involved in the development and evaluation of ACTL technologies (with e.g. drive unit test stands, CCD cameras and the AMC units).

	MST	**SST-2M-ASTRI**	**SST-2M-GCT**
Telescope Type	Davies-Cotton	Schwarzschild-Couder	Schwarzschild-Couder
Prototype Location	Berlin-Adlershof, Berlin	Serra La Nave (Mt. Etna), Sicily	Meudon, near Paris
Schedule	2013	2014	2014
Drive System	Bosch Rexroth	Beckhoff/Siemens	TBD
Devices	Drive system CCD cameras Weather station AMC Sensors Dummy Camera	Drive system CCD cameras Weather station AMC Cherenkov Camera	Drive system CCD cameras Weather station AMC Sensors Cherenkov Camera
Software Framework Common Layer	ACS, OPC UA	ACS, OPC UA	OPC UA
	pSCT	**SST-1M**	**LST**
Telescope Type	Schwarzschild-Couder	Davies-Cotton	Parabolic
Prototype Location	Mt. Hopkins, Arizona (VERITAS Site)	Krakow, Poland	La Palma, Spain
Schedule	Summer 2014	2015	2016/2017
Drive System	Bosch Rexroth	Bosch Rexroth	Siemens
Devices	Drive system CCD cameras AMC Sensors Cherenkov Camera	Drive system CCD cameras AMC Cherenkov Camera	Drive system CCD cameras AMC Sensors Cherenkov Camera
Software Framework Common Layer	ACS, OPC UA	ACS, OPC UA	OPC UA

Table 3.8 – Overview of the CTA Telescope Prototype Projects. *Note:* the LST prototype will eventually move to the CTA site to become a CTA LST

3.4.8 Missing tests and prototypes

While prototypes for the main building blocks of ACTL already show promising results, some prototype tests have been postponed as they depend on (i) a more detailed interface definition and (ii) a detailed architectural design of the central array control software. Both of these points will be addressed early in 2015, leading to a complete set of prototype tests. This section gives a short overview on the most important tests to be done.

A very important set of tests missing and to be scheduled early after the ICD definition is the test of the integration into the ACTL environment of the various camera projects (FlashCam, NectarCam, etc.) in both the data acquisition system and slow control, monitoring and configuration of the cameras.

The Cherenkov Data Repository requires extensive tests in order to evaluate the integration of the ACTL DAQ and the DATA management needs and the performance of the data storage in such schema.

In addition the best technologies and setup for the engineering and calibration repository will be tested after the requirements from the device teams have been determined via the ICD definition process.

The integration of the various ACTL software components with the central control will first be tested by mocking the basic building blocks of the system and later on via continuous integration tests following any further development in any of the components.

Finally, the technologies for the operator GUI need to be tested after a basic set of specifications have beed defined.

4 Design Validation and Product Acceptance

The following sections provide a snapshot summary of the full Design and Validation Document [14].

4.1 Safety

Safety procedures are integrated into the design concepts for the ACTL system. The main safety issues regarding ACTL systems have been identified as: safety of the devices (telescopes, etc.), safety of humans (engineers, operators, etc.) interacting on-site with ACTL systems as well as safety of the on-site ACTL hardware such as the computing and storage cluster.

4.1.1 Safety of devices (telescopes, etc.)

ACTL follows the paradigm that instrument safety has to be assured at instrument level. From this it follows that ACTL should not be able to send a specific control command (e.g. with configuration parameters out of range) that is executed by the instrument low-level software without further sanity checks. It should be assured at instrument level that it is not possible for ACTL to e.g. drive the telescopes to dangerous positions during daytime (if not overridden by instrument experts for e.g. tests done during commissioning) or switch on the high voltage of the cameras at improper times. Nevertheless, ACTL will implement sanity checks in the higher-level software to prevent from sending improper control commands. Special care will be taken e.g. by the scheduler and central array control software to not schedule observations so that failing telescopes are endangered by the rising sun at the end of an observation night. In addition, ACTL will provide an alarm and logging system, which will allow the ACTL system as well as operators to immediately react to endangering situations. A well-described set of interface control documents will allow to document all possible alarm and failure states as well as clearly define the validity range for control commands. The proper behaviour of the interface between ACTL and instruments will be tested prior to the operations phase of CTA.

4.1.2 Safety of humans interacting with on-site ACTL systems

Humans can be endangered while interacting with the on-site hardware of the ACTL system, in particular inside of the data centre and control room. Both facilities will be designed according to safety standards applicable for large-scale computing facilities, e.g. fault protection against electrocution etc.

4.1.3 Safety of on-site ACTL hardware

The devices of the ACTL system (computing and storage nodes, network infrastructure such as switches and cabling, etc.) will be protected against environmental impact similar to what is used in standard data centres (cooling units, uninterruptable power supplies, etc.).

4.2 Performance

Several ACTL products have high performance requirements. Among them are e.g. the data acquisition system and the reaction of the scheduler and central array control system to ToO alerts. These cases have been identified and will be accommodated during the architectural design of the system. In addition, regular regression tests (after each upgrade, patch, etc.) will be performed both well in advance on the ACTL test bed cluster before deployment and on the on-site data centres after deployment to monitor closely any change in performance.

4.3 Reliability

Reliability is the ability of a system or component to perform its required functions under stated conditions for a specified period of time. In ACTL, the reliability is studied both at the system level, in order to understand failure processes and the subsystem level, in order to evaluate failure probability and define potential solutions. Some of the ACTL systems are heavily software-based (e.g. DAQ) while others (SWAT-CDTS) are hybrid systems that involve specialized hardware as well as software. In this section, we outline mechanisms to assess and ensure reliability of all ACTL systems.

Software Development Standards

Software development standards to ensure reliability, quality and maintainability of the ACTL software products are foreseen to be used for the design, implementation and maintenance and operation phase of CTA. These standards are detailed in the preliminary Design Design Verification Document of ACTL [14] and in preparation for the CTA Software policy document (in cooperation with DATA) [34]. The latter software policies is planned to be followed (as close as possible) by all teams (DATA, ACTL, instrument teams) producing software products for CTA. In this document, a special section will be created that is applicable to all ACTL software products and will take into account the diverse development environments, such as the usage of already existing software frameworks (OPC UA, see e.g. [5], ACS) or the usage of different programming languages (Java, C++, python). The guidelines and rules should allow for choosing the right framework, programming language, etc. while maintaining best practices and development standards to ensure the reliability of the ACTL software products. The best practices comprise (not complete list):

- central code repository for a well-defined ACTL software product
- versioning of the ACTL software product
- tools for building and testing software products
- wiki pages and mailing lists for software developers
- bug tracking and ticketing systems
- regular code reviews by experts
- test suite (unit tests, etc.) to be delivered with each software product
- emulation/simulation tools for testing the behaviour of each software product
- naming conventions and coding style
- regression, performance, integration etc. tests where applicable
- documentation for all software products using a common documentation system
- manuals where applicable
- usage of external libraries will be minimized and for the same functionality only one well-defined external library shall be used

- software shall be produced in a modular way allowing for easy exchange of software modules

The different tests and test tools will be used for the validation and verification of each software product as defined in [14]. A test bed cluster will be available for the different tests and in addition for full system integration tests. This test bed cluster will represent a down-scaled version of the final computing cluster, network and software environment as deployed on-site. Any software product developed within ACTL (and where applicable also within DATA, e.g. the pseudo real-time level-A analysis, or within instrument teams, e.g. OPC UA servers) has to go through a full test cycle (the cycle will be defined for each software product) at the test bed cluster before it is deployed on-site. The same holds true for any update, upgrade or patch-release for the already deployed software. The appropriate actions shall be well-documented and controlled. Such high-reliability software standards can be enforced by a well-defined group of (long-term) experts, such as a software change control board with software coordinators from each product. The role and setup of such a board still needs to be defined.

Dedicated test procedures for software reliability

During the implementation of the software, several dedicated test scenarios will be developed to ensure software reliability. Not only maximising the coverage of unit tests contribute to the ensuring the software reliability. In addition, tests such as feature tests, load tests and regression tests will be executed and documented. As explained in more detail in Sec. 5.1.4, resilience tests such as netflix' simian army or 'chaos monkeys' can test for software reliability.

Hardware reliability

The hardware (computing and storage cluster, network, clock distribution, etc.) to be deployed on-site will be redundant where applicable and special automatic failover and error recovery systems, such as expert systems, are under evaluation to guarantee a high reliability of the ACTL system.

Global reliability of clock distribution system and software array trigger

The clock and timing distribution system (CDTS) constitutes a critical point of failure for CTA. The array trigger also constitutes a single point of failure mechanism at this point in time. Such a failure can be hardware, software, or network-related or arise from a combination of these factors. Failure modes may also arise at the interface between the CDTS and the SWAT or at the interfaces between both of these subsystems and the cameras.

Studies will be performed for both the CDTS and SWAT subsystems to evaluate failure probabilities and define potential solutions to increase reliability (some of the failure modes and mechanisms for increasing reliability of the CDTS are considered in subsequent sections). A Failure Mode Effects and Criticality Analysis (FMECA) will also be performed at the system level, for both the CDTS-SWAT combination and for the entire CDTS-SWAT-Camera-DAQ loop, and a maintenance and recovery plan will be written. We consider in this section what is needed to perform these assessments.

Different types of components require different strategies to assess failure modes and rates:

- Custom electronics boards (UCTS) and custom firmware. For custom boards, it is necessary to components derating and perform a failure in time (FIT) analysis using FIDES methodology. The result of these analyses is useful for design improvement and for defining the spares strategy.
- Commercial components, such as computers (servers), network I/O cards, Ethernet switches, White Rabbit switches, and power systems should be selected taking into consideration the reliability data given by the manufacturer and long-term availability for component replacement. These assessments may be revised based on the input from laboratory calibration work. Spares should be available for each component (mandatory when critical for the observatory) in order to keep as

low as possible the Mean Time To Repair (MTTR). Neither commercial White Rabbit installations nor the prototype setups discussed in 3.4.4 have reported major component reliability concerns. The first industrial-class White Rabbit switches are currently being developed by companies which will further improve the reliability of this key CDTS system component.

- An FMECA should be performed on software architecture. Unitary tests performed in order to detect bugs and check the software corruption immunity. The SWAT and the CDTS control software must also conform with the global ACTL software standards. Key performance criteria for the SWAT must also checked, in particular the packet loss rate as a function of input rate, packet size, choice of network I/O card and the number of ports used by the SWAT to receive timestamp information from the UCTS.

At the system level, a full FMECA requires a functional analysis of the system with all components defined, including software architecture, operating systems, and commercial software, as well as the PBS. As previously noted, a full FMECA must include the operation of the SWAT and CDTS systems in concert and in context, which means that the design and function of other subsystems inside and outside the ACTL work package must be taken in to account. Critical examples include ONSITE (signal and/or information loss due to broken fibres) and the cameras and DAQ (e.g. high rate bursts of camera triggers overloading the White Rabbit network or SWAT). The results of a full FMECA will depend sensitively on the paths taken by timing information and bulk data and the interfaces specified between the timing system, software array trigger, and the telescope cameras. The results of this analysis will be fed back into the final design and cost estimates.

4.4 Availability

4.4.1 Availability of data acquisition software

The data acquisition software requirement states that the availability of the DAQ must be larger than 99%. This is quite challenging as it translates to a downtime not longer than 200 seconds every night. Various events can lead to an interruption of data taking. Some of them can be mitigated by automatic recovery procedures, while others may require specific action by the operator or simply cannot be resolved immediately. The requirement only applies to the DAQ-related errors, thus errors coming from the environment are not included.

The errors that can be mitigated will be dealt with by introducing a *DAQ-Master*: a component that will monitor the state of all other components and take appropriate actions in due time.

- *Server failure*: Assuming 12h long nights, one server will probably fail as often as every 40 nights. If many system errors are detected (e.g. by a log-inspector process) then the server will be pre-emptively taken out of the working pool.
- *Silent data corruption*: despite extensive error-correction mechanisms built into modern hardware, data can still get silently corrupt [35]. A checksum [36] will be calculated with each piece of compressed data before it is written to the on-site repository. Corrupted data can be discarded before being stored on disk.
- *Software errors*: the DAQ software will be extensively unit tested and debugged before being deployed. Software errors can never be fully excluded, however remaining bugs will be fixed as they are discovered which should reduce the rate of this source of error along time.

The errors will be detected and healed as follows:

- *Missing heartbeat*: if no heartbeat is received by the master after a given duration, then the node is considered as faulty and a new one will be started.

- *Connection loss*: if one or more components report losing the connection to another one. This might be caused by a failure in the remote connection itself rather than a faulty peer, thus at least two components reporting this error are required before it is registered by the master. New components will be restarted.
- *Idle Component*: if a component is sending or receiving data at a much lower rate than expected. This component most likely has an internal problem and will be replaced by a new one.

Errors not related to a hardware failure indicate a software issue. Hence a report of the error will be automatically logged by the DAQ-Master and delivered to the persons in charge of the faulty component. More information regarding the error handling strategy can be found in section 3.3 of [16].

Availability of the software array trigger

In the case of SWAT hardware failures, a cold-swap replacement unit shall be available at all times (A-RAMS-0050). SWAT hardware failures will be monitored for by the central control system (ACS) with a time-to-failure detection of seconds. An FPGA register can be used to monitor and record the current White Rabbit synchronization status along with the timestamp. It is less clear at this point in time how to successfully monitor for and detect the other types of errors on a short timescale. In the case of a software failure the SWAT program can be automatically restarted. Network-related errors (e.g. duplicated, out of order, late, early, buggy data) shall be logged and in most cases SWAT can continue its operation despite the errors.

4.5 Product acceptance

4.5.1 Information, computing and storage infrastructure

The test bed cluster allows for acceptence testing of the to-be-deployed ICT infrastructure. Basic tests for the performance and reliability of the ICT hardware are on-going and will allow, among others, for the following tests, which need to be continued with the full on-site computing cluster:

- *switches*: data rates, data flow and load balancing
- *computing nodes*: load balancing, auto negotiation of IP addresses, CPU performance
- *storage servers*: read/write throughput, layout of storage system

4.5.2 Clock distrubition and trigger-timestamping system

Accurate calibration of the White Rabbit system on which the clock distribution and time-stamping system (CDTS) relies is essential to achieving the CDTS performance goals and CTA requirements. The calibration of the complete CDTS system will be based on the official White Rabbit calibration document [1].

The main goal of the White Rabbit calibration, and therefore of the calibration of the CDTS system, is to determine the TX and RX latency of all White Rabbit devices as well as the fibre asymmetry (the difference in TX and RX propagation time) of each fibre used. In principle, this must be done for each device on its own, unless lab calibration tests indicate that we can fulfill the timing requirements of CTA by using one set of calibration values for each type of device (i.e. a universal set of calibration values for all UCTS boards, another universal set for White Rabbit switches).

[1] `http:www.ohwr.orgattachments3338WR_Calibration- v1.0- 20140813.pdf`

Laboratory (off-site) calibration: In order to calibrate the CDTS in the lab, a White Rabbit calibrator (plus spares), is required. The calibrator can be any White Rabbit capable device. For ease of use in both the lab and the field we recommend a “portable” version of the UCTS board as the calibrator instead of a White Rabbit switch. The calibration also requires several White Rabbit devices (i.e. White Rabbit Switches and UCTS boards), various optical fibres to connect these devices, a suitable high-precision oscilloscope, and two long (several km) fibres from different production batches.

All White Rabbit devices can be calibrated in the lab using the components listed above and these components are naturally a subset of those required for the on-site calibration (since they are a subset of the CDTS components that will be deployed on site). Lab calibrations will be used to troubleshoot and perfect the calibration scheme and test the monitoring of the system in addition to measuring the setup's overall performance. The lab calibration setup will connect a few devices using the multi-kilometer fibres and connect the rest using shorter (i.e. 2 m) fibres. All calibration values obtained during the lab tests will be stored (i.e. in a database) and made available for later use on-site.

Field calibration: The calibration values of the White Rabbit devices determined in the lab can be safely re-used in the field without being re-measured. However, all fibres that might be used by the CDTS system (this includes spares as well) must be measured to determine their fibre asymmetry. As the fibres to be measured will already be deployed, the default calibration scheme is to use a loop-back fibre (one of the other fibres already present in the system) as described in the White Rabbit calibration document. Lab tests will be used to determine if a simplified loop-back calibration scheme with no electrical to optical signal converter can be used. (This also depends on the fibres to one telescope being of exactly the same type). Using this scheme, two CTA-fibres will be calibrated for each telescope (aka four plain fibres). If possible, a third CTA-fibre will be calibrated as well.

Absolute time calibration: If the absolute timing needs to be verified beyond the guaranteed precision of the used GPS receiver or atomic clock, a few days use of a more precise reference clock is required. The accuracy and the precision of our clock can then be compared to the more precise atomic clock by comparing the distribution of the differences between the two PPS signals.

CDTS verification

Depending on the precise nature of a CDTS failure it may impact data and/or cause data loss from a single telescope, a set of telescopes, or the entire array. Detailed laboratory tests, including camera-integration tests, will be used to a) verify the performance of individual CDTS components and b) confirm that a standalone CDTS system delivers the required precision. The only remaining challenge is the verification of the system in the field, i.e. at the CTA sites, under real conditions. This stage of the process is particularly critical, since we expect that only at the sites will all telescope components (drive system, all auxiliary components, final cabling and grounding, etc.) be fully assembled and integrated. This means that only at the site can we exercise the system in all possible operational modes and measure how certain effects, such as noise from a telescope drive system, impact the function and stability of the CDTS.

During the commissioning phase of the first telescopes of each type, verification tests are foreseen that would last for a few days. These include tests performed during assembly and integration tests (AIT); a good example is exercise of the drive system in all possible modes, with the camera placed in a test/debug mode. The successful outcome of these initial tests for each telescope type will allow to shorten the routine verification of additional telescopes. The same setup used for verification can be applied in case of debugging of a particular telescope, i.e. in the (unlikely) event of timing-problems for a given telescope, which cannot be solved by standard error handling (e.g. software resets, power cycling, or simple component swaps).

We detail below what is required for *in-situ* verification and monitoring of the full CDTS system.

Master clock verification Passive verification of the camera master clocks provided by the UCTS boards is done via Monitoring UCTS-boards (MUCTS boards) that can be temporarily located close to the UCTS boards at the telescope. The intention is that this verification setup is temporary and will have no influence on any concurrent data-taking or tests.

Figure 4.1 illustrates, for the left-hand UCTS board, how the MUCTS board can be located nearby. This MUCTS board can, in the simplest case, be identical to an UCTS board (but with adapted firmware). The MUCTS is, like every UCTS board, connected by a standard WR-fibre to the WR-Switch network, and thus synchronized to the central master clock. The camera UCTS board sends a PPS signal to the MUCTS, which time-stamps the PPS on its trigger input with nanosecond precision based on the MUCTS clock just as the UCTS time-stamps the camera trigger signals. The MUCTS board sends the monitoring time stamps at a low rate (1 Hz) to the CDTS-server, where they are filtered out of the normal camera trigger-messages stream before being analyzed for verification. The algorithm is straightforward. Any deviation of the time difference between two subsequent MUCTS trigger from $(10^9 \pm 1)$ ns would indicate a problem on the UCTS clock (or on the MUCTS), and would trigger a deeper investigation.

If necessary, the UCTS board can supply clock signals different from simple PPS pulses, i.e. higher frequency PPS-aligned pulses, or signals generated in a more complicated way (i.e. not phase-aligned to the UCTS PPS). Current experience with existing field installations strongly suggests that unstable UCTS operation modes are extremely rare and that the PPS-based tests described here are sufficient to detect them. The duration of the verification period is not yet strictly defined and will be adjusted based on the results of the first tests. From our point of view, it is possible to operate a few telescopes in “verification mode” for a longer period without incurring additional effort or impacting science operations.

Timestamp verification The time-stamps generated by the CDTS will be routinely monitored and verified during normal operation. This error-checking process relies on the the redundant timestamp information provided by the double-clock/double-timestamp architecture described in section3.3.2. The redundant information is generated in parallel on the UCTS and on the camera side during normal operation. Two cross-checks are made on an event-by-event basis: verification of the correct synchronization of the camera clock (which is slaved to the UCTS master clock) and verification of the WR time-stamp itself.

Synchronization between the UCTS master clock and the camera clock is done using the PPS and 10 MHz clock signals provided by the UCTS. Under ideal conditions, where the PPS and 10 MHz lines suffer from neither pulse jitter nor electromagnetic interference, this synchronization is trivial to accomplish. Given more realistic conditions, we must be able to detect significant slippage of the camera clock with respect to the UCTS clock under a variety of circumstances. This slippage can occur periodically and accumulate over a longer timescale (for example, if induced by noise pickup on the PPS and/or 10 MHz lines) or happen due to a one-time insult (e.g. if the UCTS clocks were readjusted without also resetting the camera electronics). The simplest, cheapest (since it avoids additional hardware), and most effective method for monitoring for clock slippage is to monitor the redundant event counters and time-stamps from the UCTS and camera in software. The best location for this monitoring software is the camera server (CS) where both the UCTS and camera timestamps and event counters are available (see Fig. 3.20) to be checked for every event.

As a simple and effective method (avoiding additional hardware) the redundant event time-stamps from UCTS and camera are used. Such an approach is well-known from merging of data originating from different subsystems, which are read-out into independent data streams and are tagged by an incremented counter. As previously noted, the following data are available on the camera server (CS):

- UCTS timestamp and UCTS counter for event i, abbreviated $[t, count]^i_{UCTS}$.
- Camera timestamp and camera counter for event i, abbreviated $[t, count]^i_{Cam}$.

An application on the CS will check for each event i the correct correspondence of the UCTS and camera times using the time difference $\Delta T_{TRIG} = (t_{UCTS} - t_{Cam})$. This ΔT_{TRIG} should be within expected limits set by the respective resolution of the WR and camera systems; in practice this is always

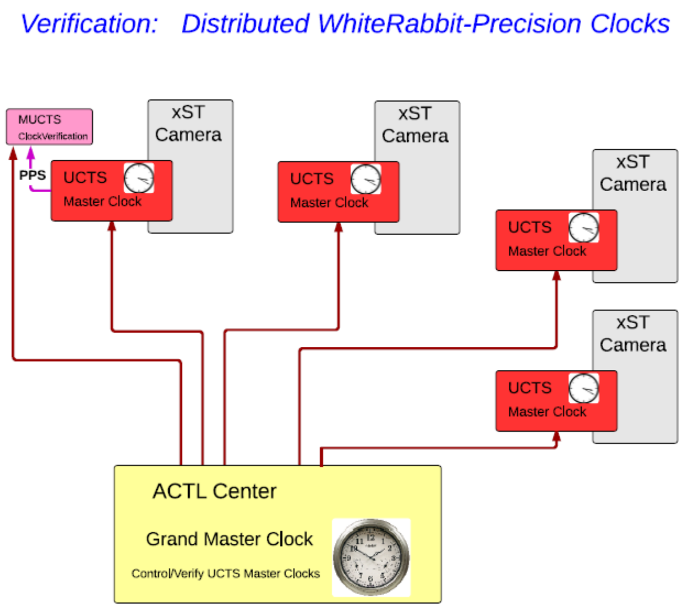

Verification: Distributed WhiteRabbit-Precision Clocks
MUCTS
ClockVerification
PPS
UCTS
Master Clock
xST
Camera
ACTL Center
Grand Master Clock
Control/Verify UCTS Master Clocks

Figure 4.1 – The UCTS clock-verfication principle: a Monitoring UCTS board (MUCTS, purple board in upper left) is located close to the UCTS. A PPS-pulse from UCTS is time-stamped by the MUCTS trigger input with the same nanosecond resolution as for normal events, thus recording with nanosecond precision any relative phase change between the UCTS and MUCTS clocks.

dominated by the system with coarser timestamp resolution, i.e. the camera. The expected limits will vary camera by camera due to differences in the clock discretization (e.g. $4/\sqrt{12}$ ns for FlashCam and $100/\sqrt{12}$ ns for NectarCam and LSTCam). The monitoring/verification process can safely set the time difference for the first event of a run to zero to cancel any absolute time offsets between the UCTS and camera timestamps.

To detail all possible error patterns that might be detected during the comparison of the UCTS and camera timestamp data is beyond the scope of this document (and indeed, some patterns may only become evident during future laboratory and/or camera integration tests). Some of the most common error patterns and their manifestation in the $[t, count]^i_{Cam}$ and $[t, count]^i_{UCTS}$ stream on the CS are given below.

- UCTS: spurious or missing trigger camera signal causing a jump in the $[count]^i_{UCTS}$ [count] or spurious or missing UCTS trigger messages.
- Camera: a strong jitter in the PPS and/or 10MHZ signals (or even spurious pulses) can cause a shift in the camera clock. The main symptom will be camera trigger times that are offset from the UCTS reference. The duration of the problem may be short or long and the size of the offset is unpredictable, since it depends on details of the pulse distortion and on the camera clock synchronization algorithm.

We expect that monitoring ΔT_{TRIG} using the $[t, count]^i_{Cam}$ and $[t, count]^i_{UCTS}$ counters will permit us to: detect the majority of errors, such as a missing event number in the data; correct the problem (e.g. by re-aligning counters); and raise alarms if the failure frequency exceeds some threshold. Even rare errors can be detected since the monitoring process is continuous and parallel to data-taking.

Difficulties are anticipated only in the unlikely cases where the error rate on either side becomes extreme or the clock and trigger signals malfunction simultaneously. These scenarios obviously demand a systematic back-tracing of the cause and rapid repair of the problem. If an additional safety margin for the timestamping of the camera bulk data is required, it can be obtained by running on each camera an independent clock (at 10 or 100 MHz), that is not synchronized to any external reference. Such clocks would be only subjected to slow drifts due to environmental changes, oscillator degradation, etc. This independent clock counter could be added to the bulk camera data.

5 Plans

5.1 Construction Plans

This section describes the steps needed for the construction of the ACTL products, i.e. all steps of manufacture, procurement and production, assembly and integration as well as commissioning and verification of the software and hardware products. This section gives an overview on the tentative plans, based on a preliminary software architecture. The software architecture is currently being refined and will be followed by a revision of the project plans based on the refined software architecture.

5.1.1 Manufacture

Manufacture of software

In this section, the steps necessary to manufacture the ACTL software products are described. These steps comprise all steps from project organization to software architecture, from software development to the deployment of software releases.

Software Architecture

The main input for the project planning is the software architecture. A well-defined software architecture together with a process and implementation model are the basis for the project planning (incl. WBS/PBS), the organization of work and cost estimation as well as the risk assessment and prioritization of work. The software architecture efforts thereby allow for documenting the design reasoning and decision-making processes within the project. When followed throughout the project phase, these efforts ensure the coherency and consistency of the ACTL software products.

Documenting the software architecture facilitates communication between all stakeholders, captures early decisions about the high-level design, and allows reuse of design components between (sub-)projects. These efforts make the design decisions accessible and allow to manage the knowledge about the architecture and its communication. In addition, software architecture helps to identify risks and is thereby a means to reduce costs in complex projects. The ACTL team recognizes the need to have such a formal software architecture process throughout the whole project phase.

The preliminary ACTL software architecture is largely based on experiences from current IACTs, from projects of similar scale such as ALMA and from ACTL prototyping efforts. In order to revise and refine the ACTL architecture the ACTL team has acquired the support from external software companies specialized in software architecture and project organization. The collaboration with and consultation of such external companies started in early 2015 with initial workshops and has been continued since.

As a first step in this process, the architecture drivers, i.e. the functional and quality requirements as well as the list of stakeholders and their relation with the ACTL system, are derived. The ACTL requirements [13] and specifications [14] as well as design assumptions (see Sec. 2) and ACTL use cases (see Sec. 3.2.2 and [15]) provide main ingredients for identifying the architecture drivers. The architectural design choices are narrowed down by the following decisions:

- **Software frameworks:** ACTL will use ACS as the software framework to control the array. OPC UA is the standard interface to the instrument software. See Sec. 3.2.2 for details.
- **Device integration:** ACTL will follow the prescription on how to integrate the low-level software and firmware of the devices into the architecture (see Sec. 3.2.1).

From the architecture drivers, different architectural models (or architectural views) are derived:

- **context model:** treats ACTL as a black box and illustrates all relations to the outside world as well as all responsibilities and dependencies on outside systems
- **functional model:** high-level and solution-independent functional model of the ACTL system, including all high-level interfaces
- **logical model:** software components, data types and all internal and external interfaces, addressing the question on how the functionality is implemented and mapped to the components
- **technology/deployment model:** details on where the software is running

The full architectural model will detail all of these views, their relations and the mapping of these views with each other (functional to logical, functional to deployment, logical to deployment) and with the architecture drivers. In the course of the project organization, the ACTL PBS can be refined from the functional model and its functions, while the refinement of the WBS can be derived from the logical model (e.g. the software development effort) and its relations (e.g. the integration effort). In addition, the definition of the process model will enable a detailed scoping of all activities.

In addition, the architectural work will address cross-cutting concerns (e.g. error handling) and give implementation guideline (e.g. UI integration). The architectural documentation will serve as a refinement for the project planning and organization and will grow as the project evolves. The process of formally defining the subsystems and relations will allow to uncover holes and sharpen the separation of responsibilities between subsystems, as well as to easy the evaluation of the reliability and availability of the system.

After this process finishes in Q2 2015, a full refinement will be done. It will not only allow the refinement of the project planning, but also allow identifying the software products to be outsourced to industry. Further refinements, going into more depths for high-risk and high-impact architectural design choices, are planned together with iterations until the level of detail is sufficient for catpuring all architectural impact and the level of detail is sufficient for software development. Existing prototype software can be evaluated against this architectural design and the cost for further refinement of the prototype software can be estimated. Regular architectural reviews of the on-going efforts will help to keep the coherency and consistency within the software development efforts.

Software development

Each team creating an ACTL software product within each ACTL sub-work package will follow a set of defined rules and procedures established by the ACTL management and central activities teams. These procedures are backed up by the so called ACTL *tooling* system, which is able to enforce the rules and procedures with automatic means. Software products converge into releases which are fully tested, including compatible sub-products, and ready to be deployed on-site. In other words, all the software PBS items will be produced and deployed using the same procedures and tooling.

The ACTL software in the form of advanced prototypes and different projects from the sub-work packages will realize the system architecture which is built based on the existing requirements and external interfaces. See section 5.1.1 for further details.

The implementation of the software (writing the code) will follow the general CTA software policy as detailed in [34] and will take advantage of the specific tools for each framework. Most ACTL code will be developed in the Java, C++ and Python languages. The process to incorporate new ACTL software starts by committing the piece of software into the ACTL repository (currently the *Subversion* repository provided by DATA), which triggers the automatic quality check and peer-to-peer code review procedures by ACTL team members and organized by the software release team. The new piece of software will not be incorporated into the ACTL software *trunk* and thus be ready to be included into a release until it fulfils the required quality parameters.

The software development process will be supported for dedicated tooling (see also Sec. 5.1.4) to allow the collaborative writing of software, the verification of the software in a well-defined manner (including automatization such as continuous builds). All software will come with a test suite (e.g. unit tests) and go through further tests (system tests, acceptance tests, performance tests) applicable to the respective software product. In addition, all software will come with a well-defined set of documentation. The tooling includes virtual machines for software development as well as a dedicated test cluster for integration and performance tests.

Workshops and instrument support

In order to allow for the convergence of various points of view within and outside the ACTL team and to transfer the knowledge between members of the ACTL work package and to other work packages, the ACTL teams established dynamics of dedicated workshops and *bootcamps*. Regular workshops are key for the progress of the project and are an active part of the project planning. The scope of the workshops can be:

- **Workshops between ACTL and other work packages:** used to formalize the interface definitions, to organize the creation of system demonstrators, to align the integration work, to refine the features of the ACTL systems and when they should be available etc. Scientist should be part of these workshops.
- **Workshops within ACTL sub-work packages:** used to synchronise the activities within a sub-work package and to plan and distribute the work ahead. Each sub-work package will organize its dynamics depending of the specific needs and the distribution of the manpower.
- **Workshops on architecture:** used to define and follow-up the evolution of the ACTL system architecture.
- **Workshops for knowledge transfer and instrument support:** Workshops to transfer the knowledge from experts to beginners. Instrument support, e.g. by transferring the expertise of OPC UA experts from ACTL to device teams as in the *OPC UA bootcamp* that was organized in October 2014.
- **Workshops with experts team consultation:** Workshops used to gather help and collect expertise on specific topics. Such workshops include e.g. the operations of large infrastructures or GUIs for astronomical installations.

Software releases and deployment

Fully operational ACTL product versions are provided in the form of releases produced in 3 month cycles, which would be organized in such a way that the ACTL products required for the CTA operation schedule are provided on time. The release management team takes care of the organisation of the releases and the enforcement of the quality of the obtained software. Appropriate tooling for releasing of the software and its deployment (release management systems, ACTL virtual machine with ready-to-install kits, RPM-based installation, etc.) are currently being tested and will be used.

The ACTL work-package will use an iterative and incremental delivery of software seems to be a practical approach for the ACTL software releases. In order to be agile enough but also flexible in order to integrate the effort of the geographically distributed software teams, the delivery cycle of 3 months is proposed for the ACTL team products. The length of the cycles may eventually need to be adjusted later on during the software creation process. In addition, the possibility to include some special *service releases* to fulfil eventual specific needs is considered. In any case, the procedures defined in Sec. 4.3 will be applied to any ACTL software product prior to its release. The release process will be coordinated by the ACTL software release and management team who will identify and assign the corresponding roles to coordinate the task. The releases will be fully tested in the ACTL test-bed before any deployment on site.

The software development efforts of ACTL will follow a set of milestones established for every 6 months, and based in specific use cases. These use cases will cover specific functionalities of the system (e.g. operator is able to point telescopes to any position of the sky, or the operator is able to start and configure the system to acquire data from the cameras) and will require the coordinated development in various sub-work packages. For each of the use-cases, a scientist will be assigned in order to make sure that the development follows the scientific needs of the CTA arrays. Given the 3 month release cycles, this means that 2 ACTL software releases are expected to exist within the 6-month milestones period. The ACTL team is currently working in the definition of the specific list of such milestones.

The tentative plan for the ACTL software releases, their scope and purpose comprises the following steps:

Development Releases (ACTL internal, in the pre-production phase on the ACTL test cluster)

1. Test system for central OPS software and software architecture in a single subarray with one type of simplified telescope
2. Like 1, but with data-acquisition for emulated cameras and mass storage
3. Like 2, but with an additional telescope type
4. Like 3, but with operation of subarrays and parallel observations
5. Readout and control system for the first telescope on the southern CTA site
6. A test system for RTA, event filtering and compression
7. A test system for interoperability of the SWAT and the telescope data acquisition

Test Releases (Official releases, for hardware tests in the laboratory)

1. Readout and control of a complete Cherenkov camera
2. Readout and control of CCF devices (e.g. a weather station)

Commissioning Releases (Official releases, for commissioning on CTA sites)

1. Readout and control of the first Cherenkov telescope deployed on a CTA site
2. Like 1, but with readout and control of central calibration facilities

Construction Releases (Official releases, for commissioning and science verification)

1. Joint readout and control of several telescopes of different types
2. Like 1, but with reaction to GRB alerts, ToOs and changes of weather conditions (for the scheduler)
3. Like 2, but with automatic scheduling

It is exptected that the software for the releases will be developed along the lines of important use-cases. The main strategy is to provide a basic funtionality early (e.g. script-based observations) and to implement less essential features (automatic observations and scheduling, advanced operator GUIs) only when they are really needed.

Individual work packages

WBS 3.1: ACTL-SLOW

A prerequisite for the development of the ACTL-SLOW software products are the interface definitions, a high-priority task for ACTL at the time of writing. The interface definitions will focus first on defining a uniform set of minimal functionality for the slow control software components interfacing the various hardware devices. These efforts will be concluded by the full and detailed specification of the interfaces, allowing for the filling of the instrument configuration databases and alarm registers. Prototype components exist already at the various prototype telescopes and it is expected that they will evolve over time into the final slow control software components, which shall implement as uniform interfaces to the different telescope types as possible.

WBS 3.2: ACTL-OPS

The ACTL-OPS products will be developed incrementally following the 3-month cycles of the ACTL release plan and being ready to fulfil the 6-month use cases based milestones. The varios PBS items from ACTL have different natures and thus the development has some specific caveats, in particular regarding the priority of the products.

Software frameworks, WBS 3.2.1 The ACTL software frameworks (ACS, OPC UA) are already established as well as the inter-framework communication library, only requiring some maintenance work whenever such a need is identified. The main work to be done is for the ACTL virtual machine (WBS 3.2.1.2) which will grow to incorporate the tooling to support the ACTL software development and contain a mocked version of the main ACTL components.

Central array control and operation system, WBS 3.2.2 Being such a central entity, the first release of the ACTL software will incorporate a mocked version of the whole system included in this item, thus allowing any other component of ACTL to grow around it. The various components of the system will be further developed following the needs of the use-case based milestone program. An early integration of the already existing ACS tool for the alarm system (WBS 3.2.2.4.1) will be done, and the work on the recovery system (WBS 3.2.2.4.2) will be postponed for later stages of the project.

Detector quality assessment tool, WBS 3.2.4 The development of these tools would be considered a low priority work until the Central Array Control and Operation System and the On-site Data Repository are functional.

On-site data repository, WBS 3.2.5 The on-site data repository tools will be developed in collaboration with DATA-Archives sub-work package and at the first stages most of the work will comprise on evaluating the performance of the database and archive technologies for the CTA needs. After the right technology is selected, specific software libraries will be created in order to interface with the repositories the various ACTL software products. In addition, user interfaces and procedures (e.g. standard queries) will be created.

WBS 3.3: ACTL-DAQ

The development of the DAQ system will be divided in three separate phases:

- **phase 1: single telescope data acquisition system** which will allow for early integration tests with the Cherenkov cameras,

- **phase 2: array data acquisition system** which will allow integration tests with multiple cameras and an array trigger,
- **phase 3: array data acquisition system with event reduction capability** which will allow tests of the full integrated chain and develop in complexity over the full construction phase of CTA.

The first phase aims at operating a single telescope and has already started. A first, simple version of the modules is being produced to allow for single telescope operations. No load balancing will be available, the raw data will be written uncompressed (or with little compression) to disk and the data access layer will only be a file reader. The event buffer will only be capable of adding the precise timestamps to the events. The tentative delivery date is set for Q3 2015.

The second development phase will aim at enabling array-operations, with time coincidence detection and stereo trimming of events. The DAQ master will be introduced to configure the various components automatically; the event buffer will be updated to allow for time-coincidence trimming of events. It will be possible to load-balance the processing using the second version of the camera server API and data transfer protocol. The compression of the raw data will be investigated to achieve the best possible ratio and the data access layer will allow to automatically retrieve events based on their originating telescope and timestamp. The first and simplified version is planned for Q4 2015 and will increase in complexity and functionality (i.e. supporting multiple sub-arrays) over time.

The third phase will aim at enabling the full real-time data reduction pipeline, taking into account the results of the real-time analysis. The third version of the events buffer will accept a feedback from the level-A analysis to trim background events further. The DAQ master will be gradually improved, adding nodes failure detection, automatic error recovery and re-processing of lost events. A first version is planned for Q4 2016.

Eventually, the end of the commissioning period will be devoted at making the system grow in scale, the commissioning of new telescopes and system optimization and problem solving.

WBS 3.4: ACTL-TRIG – Software array trigger

The work for the development of the software array trigger (SWAT) are distributed over 4 years, with a final deadline set to Q2 2018. Initially, the development will focus on detailing the interfaces with UTCS, the camera servers and the DAQ subsystem. This will be done in close collaboration with other groups. Then, the core algorithm will be coded. Finally, the SWAT performance and its interoperability with other modules will be verified.

Three releases of the prototype are planned :

- **single array prototype system** (Q4 2015) - initial release, with basic functionality supporting only single sub-arrays and single telescope types. This release will enable first integration tests of SWAT. It will be also possible to test the SWAT prototype even before first telescopes are operational.
- **multi array prototype system** (Q2 2016) - second release of the SWAT prototype. Added functionality includes support for multiple sub-arrays and multiple telescope types.
- **final prototype system** (Q4 2016) - final release of the SWAT prototype. Full functionality with management, configuration and performance monitoring through the ACS interface.

The release of the final SWAT prototype marks the beginning of SWAT commissioning and iterative refinement of its performance and reliability.

WBS 3.6: ACTL-SCHED

The work on the scheduler is expected to evolve in parallel with other work-packages. The scheduler thereby contributes to a better understanding of the operation and scientific return of CTA: it is used to improve observation strategies (e.g., definition of the key science programs) or the expected performance of CTA based on mechnical design, site conditions, etc. For this, the scheduler needs to be involved in all construction phases of CTA and needs to fit in different places in the CTA control infrastructure (on- and off-site), thereby providing different outcomes to different work packages (ACTL, OBS/DATA,...). Recurring adjustments and feedback to the design and implementation of the scheduler are therefore expected throughout the construction phase of CTA.

The work for the full development of the central scheduler (long-term and short-term planner) is distributed over 4 years, with including more and more complexity and features from the first release. An initial phase is planned to obtain the final specifications for the software development. This will be finished once the CTA operational plan is closed, as this document has a strong impact on key points that constrain the design of the scheduling software and the optimization algorithms. Some of these points are: the observation strategy, the workflow design, the data model, the control architecture, etc.

Several software releases are scheduled in the software development plan, and they will provide: new features, the implementation of interfaces with external subsystems, and the GUIs for off- and on-site users. It is important to mention that the scheduler will have to be integrated in the off- and the on-site software infrastructures in order to provide the identified services to the scientific community and to the operators of the Northern and Southern observatories. The first release with basic features is planned for 2015, while 2018 and 2019 are foreseen to finish the GUIs and complementary features, such as the science alert service.

5.1.2 Procurement and production

Procurement of ACTL products will be done both for software and hardware. Procurement procedures will be compliant with CTA regulations and follow the national and institutional processes and procedures of the member institutes. While a detailed procurement plan still needs to be defined for each of the products, a short overview is given in this section.

Procurement of software

Software for ACTL will be both produced within CTA, i.e. designed, implemented and tested by the software developers within the ACTL work package, and contracted out to external companies for well-defined software products. In addition, it is planned to bring external expertise into the project for certain tasks such as software architecture work, software architecture reviews and consultation to assure the high quality of the software products and minimize any software development risks. In all of these cases, the final product (software, review reports, etc.) shall be owned by ACTL, including source code, documentation etc. The procurement for these software products will follow standard regulations of the institutes acquiring such services. The identification of the software products to be sub-contracted and corresponding cost estimates will follow the continuous architectural work and progress of the project.

The investment costs for software mainly comprise costs for software licenses (e.g. for OPC UA licenses) and software development tools (see also Sec. 5.4.1). The procurement for the licenses as well as software development tools will be executed according to standard procurement regulations of the institutes purchasing the licenses and/or offering the central services for the software development tools.

Timing distribution system: hardware, firmware, and software

The UCTS boards of the CDTS (WBS 3.4.2) are a custom system and the prototypes will be developed in-house. Mass-scale production of these boards will be out-sourced to industry, with the acceptance testing of the delivered boards done in-house. The firmware and software for all UCTS boards will also be done completely in-house. The White Rabbit switches will be bought from industry, with only slight in-house modifications to the firmware, if necessary, for remote monitoring and remote maintenance. The SFPs and GPS receiver will likewise be bought from industry and no modifications are anticipated. The optical fibres necessary for the CDTS will be provided by ACTL-ONSITE and a calibration and performance tests will be done in the lab with a km-scale sample-fibre provided also by ACTL-ONSITE. Note that White Rabbit and all its components are open-hardware and open-software projects and that all components will be available from multiple vendors. This precludes vendor lock-in. The assembly, software development and testing of the CDTS server will be done in-house.

Procurement of information, computing and storage hardware

All hardware for computing nodes, storage servers and network components will be bought as off-the-shelf hardware. This hardware includes all computing needed for data acquisition, array trigger, CDTS server, scheduler, central array control as well as data analysis. This homogeneous setup not only simplifies the procurement of the hardware, but in addition allows for easy maintenance and high redundancy among the components. The procurement will be done in accordance with standards applied in typical data centres. Different parties can contribute to the purchase of the hardware (regulations still to be defined) and will follow the specifications of the hardware as defined by the leading institute due to the need for a homogeneous setup.

The network cabling will be sub-contracted, including manpower for the buildup, and will be deployed in accordance with the site infrastructure development.

5.1.3 Logistics

Software

For the special case of the initial setup of the on-site data centre, i.e. when no external network is available, the software needed will be pre-installed and configured on the hardware devices and, in addition, send on backup storage (portable disks) alongside the hardware.

The ACTL software will in general be deployed with remote procedures, but in certain cases the commissioning of specific ACTL software will require that corresponding sub-work package experts will travel to the observatory sites for the testing, verification and full commissioning of the software on-site, before allowing usual operator usage and handover to local staff. Specifically, local observatory staff will be trained on site by the ACTL experts on the usage of the ACTL software products.

Any problems arising while using the software on-site, and which cannot be handled by the local crew, will be dealt with first by remote help by ACTL experts. In exceptional situations, including feature upgrades of the ACTL software e.g. to implement new scientific observation modes, travelling of experts to the observatory sites will be necessary.

Hardware

A detailed planning for the logistics of shipment of the computing and storage hardware as well as the clock distribution and trigger-timestamping system still needs to be done and depends on the deployment strategy (see Sec. 5.4.3) and selection of CTA sites. In total, 5 containers of standard 40' size are sufficient to transport the hardware for computing, storage, network and racks to the site. However, 10 or more shipments of smaller-sized containers, stretched over the construction phase, are needed due to the incremental buildup. Any components for which a preconfiguration and/or product acceptance testing is needed (e.g. computing nodes, storage servers and network switches for the initial buildup phase) will be first send to the authorized institutes, before being send to the CTA sites. The detailed shipment strategies will be defined as part of the purchase of the hardware, in compliance with the central CTA logistics regulations.

5.1.4 Integration, testing and commissioning

General approach for software

All main components of the ACTL system will be available before the actual deployment on-site to allow for full system integration tests, including all affected work packages of CTA. A deployment on site is planned only when the respective tests have been passed.

The integration, testing and commissioning of the ACTL software is integral part of the overall software development activity. A well-defined set of tooling and guidelines, supported by a dedicated group of developers, will help in these activities. The pre-requisite for a seamless software integration is achieved by the preparatory work of producing the interface control documents (both internal and external). In particular, the integration of the devices in the ACTL software will be performed in several steps:

- integration tests using simulation software and an accompanying test suite (incl. unit tests, regression tests, etc)
- integration tests in the ACTL test cluster (see Sec. 3.4.5) with an emulation version of the OPC UA servers of the device
- integration tests in the laboratory with real hardware
- integration tests on-site with operation hardware
- science verification of major releases

During the on-site commissioning phase, ACTL experts will travel to the telescopes site and instruct the local crew of the proper usage of the software. On-call ACTL experts will be ready to answer questions and solve problems that may arise during the observations after the commissioning phase.

The described tests will be performed on all individual software components, and for all releases, but especially for the external relaeses as defined in Sec. 5.1.1.

Test and integration tooling for the ACTL software

The ACTL tooling allows to both easy the creation of software that follows the general design principles of teh ACTL team and check the quality of a piece of software, detecting as well when it diverges from the architectural vision.

- **Source control:** All source code of the whole ACTL system and all device integration will be kept in single central source code repository platform not necessarily inside a single repository this simplifies the setup of a central continuous delivery pipeline but also encourages cross-codebase collaboration and communication.
- **Continuous integration and delivery:** All teams will be required to use a continuous integration system to permanently monitor the state and quality of the software they create. Continuous delivery takes this a step further by not only building the software but also creating all the deployment artifacts of it and making them available so that the whole system could be commissioned immediately. The system is supported by a delivery pipeline where different levels of testing are applied to each piece of software. In this way, only thoroughly tested software will ever reach the final stage of the pipeline to become distributed and commissioned. ACTL will provide a central continuous integration and delivery service that is available to all ACTL and device integration groups. If the development model for a software project involves branch-based development, an extended pipeline can be used to perform what-if builds of branches before they get merged giving an early warning for breaking changes.

- **Issue tracking:** A central issue tracking solution used by all groups fosters cross-codebase collaboration by cross-referencing issues, or even by moving tickets from one subsystem to another, when the responsibilities or the root cause had not been cleared up front.

- **Code review tools:** Code review web-based tools allow to easy comment, down to individual lines of code, and discuss changes in various ways. ACTL will enforce procedures that guarantee that a piece of code only ever enters the main code base after it has been reviewed by a responsible person (four-eyes principle).

- **Automated quality scans:** As part of the continuous delivery pipeline, tools performing static code analysis should be used to look for problems like common bug patterns, software anti-patterns, or violations of architectural constraints.

- **Virtual machine with a development environment** The existing vagrant ACTL box (See Section 5.1.4) will be expanded to contain as many of the tools required for developing software for ACTL or device integration as possible.

- **Project archetypes** Provide an automated way to create a skeleton project with the recommended file system layout, a working build system, the expected dependencies, and some initial skeleton code that shows how to implement certain core features. It may deliberately contain code that initially causes certain tests like a test that verifies the new project provides information about its health to fail.

- **Subsystem mocks** Each of the subsystem will provide others with a mock subsystem. This mock should provide and/or use the same external interface as the real subsystem, but require much less external dependency (in particular, no hardware). Ideally, such a mock should not just return trivial results, but provide reasonable behaviour for its domain (via simulation). An example would be a telescope drive mock, which updates the position it communicates when its has been told to move. Such a mock can be used by the other teams to develop dependent subsystems with more confidence into their understanding of the implemented contract than would be possible when they had created a mock by themselves.

- **Virtual machine with a mocked ACTL system:** The ACTL will provide a virtual machine that contains a big enough self-contained subset of ACTL software, with some of the subsystems mocked, so that other teams can test their own subsystems against this mock during development.

- **Deployment automation:** ACTL will provide the tooling for automated commissioning and updates of software to both testbed and later production. ACTL will decide whether an already existing RPM and script based solution used at DESY, or some other some of the existing provisioning tools like Chef, Puppet, or Ansible, are best suited for the task.

- **Documentation repository:** Provide a central repository through which all components can provide their documentation. It should be easy to use, searchable, and ideally provide a system for comments and questions, and also informal blog-like documents which teams can use to quickly outline decisions shortly after they have been made.

- **Documentation templates:** ACTL will provide a system to make easy to document aspects of the software being built in the same way across all groups by providing templates for some standard documents like a ”component description” or an ”interface definition”.

The ACTL system is using currently the RedMine-based system from the DATA work package for some of the tooling tasks and ticketing and code repository. However, the ACTL management team is considering to employ an alternative based on the ATLASSIAN system[1] which should allow an easier integration of the remaining tooling needs. The decision of the final tooling setup of the ACTL work package will be determined early 2015.

[1]www.atlassian.com

The “mini-ACTL” virtual machine

The ACTL team has currently created the first version of the standard virtual machine (VM) that can be used to perform the software development and testing under the same environment. The preliminary version of such machine consists of a *VirtualBox* tool managed by *Vagrant*. The VM contains at least the following items:

- ACS (includes generic GUIs and suprort for scripting with Python)
- Minor ACS dependencies (like ksh)
- Required Java version
- MongoDB
- MySQL
- Required extra Python packages
- A simplified version of the instrument monitoring tools

The VM will be available in different flavours. Specific branches of the VM will include the ACS bridges and OPC UA servers and associated configuration files, as well as examples of scripts for Python-based operation for the specific telescope prototype projects. For the instrument teams, tools like the Multipurpose OPC UA server will also be delivered. For all the projects a simplified version of the instrument monitoring tools will be provided. In addition, a VM including all projects code together could be created. Whenever a technological choice is made, for example the usage of Cassandra instead of MongoDB, such changes will be reflected in the content of the virtual machine.

Extensive software testing

Every piece of ACTL software will be accompanied with unit test which will allow for early testing for flaws in the created software. Standard tools for each programming language and/or incorporated in the framework are to be used (JUnit, ACE, boost, etc.). The software will be documented following the standard language conventions and tools (doxygen, javadoc, pydocs). In peer-to-peer code reviews it will verify that this is correctly performed.

The software will be stored in the ACTL code repository. Mocks and/or prototypes of the main software products will be made available to allow for full system integration tests of the ACTL software products, including integration tests of devices in the laboratory.

The ACTL test cluster can be used for more than just system integration tests. The ACTL team will deploy automation tools (see Sec. 5.1.4) which will be used to roll out the latest artifacts created by the continuous delivery process. This way the deployment tools, and their level of integration into the software being developed, get tested and verified as well. At the same time, the result of the integration tests should feed back into the continuous delivery pipeline, rejecting artifacts that caused test failures. Whenever feasible, the ”real” implementation of each part of ACTL should be run inside the test cluster during integration tests. In some cases, this will not be warranted, in which case the whole system should run against the mock implementation which has been provided for the system in question. Outside of regular integration tests, the cluster can also be used to perform other kinds of tests. In particular, ACTL will run resilience tests. A model for this can be Netflix’ simian army [2], where certain tools try to destabilize the whole system (by randomly introducing latency or killing services) and see how the system reacts to such problems. Similar tests should be used to increase the robustness of ACTL as a whole. In deviation from Netflix’ approach, ACTL would use these tools inside the test cluster, rather than the production systems. These tests will help to perform the FMECA analysis of the ACTL products.

[2]see http://techblog.netflix.com/2011/07/netflix-simian-army.html

Commissioning of ACTL products

As mentioned in Sec. 5.1.1, the CTA code will be deployed in the form of *releases* with 3 month cycles, and will be deployed in a commissioning phase in concordance with the deployment of the CTA telescopes and their commissioning. This will mean an extended period of final, real integration testing and training period for the users (local telescope and operator crew). An extended activity of maintenance is expected, where users will notify any issue arising with the software. ACTL will us a ticketing system for an efficient transmission of issues and requests from the users for the software developers (see Sec. 5.1.4). Furthermore, software patches, with associated verification and commissioning of the patches, will be performed after any updates of the operating system or changes in the hardware and software environment.

In order to prioritize work and reduce the needed manpower, the ACTL releases will evolve with increasing complexity. These steps will be in concordance with the available array of telescopes on site and the plans for validation/verification as well as early science verification. Such incremental releases will be incorporating, step by step, the required ACTL software products until all the ACTL products are deployed on site.

Minimum software delivered to telescope teams to perform prototype tests

The minimum software to be delivered to the telescope teams is basically what has been described before as a “mini-ACTL” VM. With that setup and the ACS bridges created by the ACTL-SLOW team, the telescope teams are provided with the basic functionality of an ACTL system needed to perform the integration and testing of the different devices. The device control and readout can be performed via Python scripts and generic GUIs, while the monitoring of the system can be performed with the simplified version of the monitoring system. In order to enable performance tests, mechanisms to make similar deployments of the ACTL software in physical machines will be provided.

Individual work packages

WBS 3.1: ACTL-SLOW

All slow control software components will be tested in the laboratory and with prototype equipment prior to the deployment on site to avoid any debugging efforts on site. Identifying the minimal needed functionality for certain tasks to be performed with the telescopes allows for iterative development of the software.

WBS 3.2: ACTL-OPS

Central control and operation system The core structure of the central array and operation system (central control system) will be deployed in an early release in order to allow the integration of the different ACTL software components around it. Some of the functionalities of the central control system will be mocked in early releases until they are required by other subsystems.

The interfacing to the central array control system will be achieved in early stages based on script-based operation (since the first ACTL release) to allow for early commissioning of the telescopes. Preliminary versions of the operator interface will be provided in following releases while the integration of the final human machine interface to the array comes at a later stage.

The preliminary version of the operator GUI will be evaluated by professional telescope operators in order to assess how adequate the product is.

On-site data repository ACTL repositories are tested in the ACTL test-bed cluster and then deployed in the on-site data centres of each array. The activities for each of the repositories (monitoring, engineering, configuration, logs, on-site Cherenkov data) are relatively independent and will be deployed and commissioned as such following the associated milestones, but based in the 3 month software release development cycles as the rest of the ACTL software products. The functionality delivered in each of the releases will be based in the specific use-case based milestone.

WBS 3.3: ACTL-DAQ

The data acquisition system will be available in three major releases. Throughout the development duration, debugging and optimization activities will be conducted in parallel with the development itself. Commissioning time will be devoted to this so that each new telescope being installed on-site can be connected to the DAQ pipeline in due time.

WBS 3.4: ACTL-TRIG – Software Array Trigger

Commissioning of the SWAT, in the form of the first on-site integration tests, will begin after three interim prototype releases and once two telescopes are operational. It is expected that the commissioning of the SWAT will be an iterative process, with integration tests being re-performed each time a telescope is added. Once the test is passed the SWAT will be marked as commissioned with the new (increased) number of telescopes. Furthermore, several iterations will be needed to fine-tune the configuration setup via ACS control. Additional time will be needed to gather enough statistical data in order to verify the SWAT performance.

WBS 3.6: ACTL-SCHED

Commissioning of the LTP and STP scheduler algorithms is based on operation indicators collected during the first months of operation. The feedback of the observations will be used to refine the scheduling algorithms. A similar approach as for the ACTL-OPS operator GUI will be followed for the GUI used both on- and off-site, where feedback during the actual usage will be used to refine its layout and functionality.

Hardware-related commissioning plans

Clock distribution and trigger-timestamping system

A more detailed discussion of the calibration and verification procedures required for the CDTS is discussed in Sec. 4.3. We summarize the key elements here. First, extensive lab testing and pre-calibration of all CDTS components (e.g. the UCTS boards) will be done to verify component performance and the fulfillment of CTA timing requirements. An integration test of the UCTS board, CDTS server, camera server and SWAT will be performed. Laboratory integration tests, such as an integration test of the UCTS board, camera server, and SWAT, should be done with each of the cameras so as to detect problems early. The integration of the UCTS boards, White Rabbit switches, GPS receiver and CDTS server will also be done in the laboratory before the components will be shipped to the site.

Components should arrive on-site fully calibrated and working (“plug and play”) with the exception of an initial fibre calibration. Following this calibration a final overall test and verification of the CDTS will be performed. This test requires that the fibre installation be complete but does not require actual telescopes to be present. Once telescopes become available, each of them will be integrated into the CDTS and its performance verified. Moreover, for each new unique combination of telescope and camera that arrive on-site, an in-depth integration and verification will take place.

Information, computing and storage infrastructure

The integration, testing and commissioning of the on-site hardware includes the on-site data centre (computing, storage and network), the array network (components and cabling) as well as the buildings network (components and cabling). The assembly and commissioning of the on-site hardware components can begin as soon as the building infrastructure (incl. power connection) is available. A tentative plan for the assembly and commissioning of the on-site ICT infrastructure is given in Tab. 5.1 and the overall buildup (excl. verification phase) lasts about 1 month. The individual steps are described in the following.

One of the first steps is the installation of the rack mounts of all racks in the data centre. The racks for the infrastructure, i.e. racks for the patch panel or the switches, are needed first in the data centre. The rack installation phase includes the electrical connection and the installation of cable trails. This step is necessary to start the array cabling and requires a structurally engineered finished room for the data centre.

The array cabling includes the installation of the cables from the patch panels in the data centre to the power transformer buildings or patch points. The cables will be installed in completed trenches and patch points, making the cabling highly dependent on the building measures of the trenches and the transformers. If the first telescopes are ready, the link from the patch points to the telescopes can be installed in this step too. As the construction of the trenches will be done in several steps, the cabling will follow the same approach. The cabling comprises the installation of the patch panels (data centre and telescope / patch points), to lay tracks and to splice the optical fibres. If the first optical fibres are connected and patched completely, it will be necessary to install the trigger system in the data center to go for first on-site trigger tests.

As soon as the interior fittings of the buildings will start, the cabling of the offices and the control room should start as well. The rooms in the technical building will be linked directly with the data centre. The residual building and others will need to have a small technical room, where a switch, which is connected to the data centre, can be located. From this switch all rooms will be linked with Ethernet cables. During the array and office cabling the power and cooling solution at least for the data centre has to be in operational mode. This is the time when the core network components will be installed. During the installation of the network components the internet connection should be available.

If the network functionality is available, the server rack mount and server nodes installation will start as well as the storage node installation. That includes the physical installation of servers as well as the patching. The configuration of the server will be done off-site. The first installed servers will be servers for core services like dhcp, dns and the server for the configuration management. The software installation of computing and the storage nodes is based on a working basic infrastructure. After the installation of the basic infrastructure and the final configuration of the servers a validation and verification phase will follow. Some basic tests, like testing the operating system and the connections to the internal network will be done here. After the rack mount of the camera server and its final operating system configuration, other groups, like the camera groups or data management, can install their software on the prepared server. The number of server and storage nodes will grow with the number of telescopes. Every telescope and of course the buildings, will be covered with 2.4 GHz Wi-Fi. The installation of the wireless LAN will start directly after the core network and services are available and will be launched in small steps. There will be no full coverage on the whole array.

The first aim of the commissioning phase is it to provide a basic operational mode of the network and server structure to be able to make functional tests of the first telescopes. Finally, the commissioning phase will be closed with the full operational functionality of the data centre which can be reached by thorough testing of all services, including the network, and a test phase with first users.

Commissioning plan	Task	Detailed work	Dependencies	Expected number of workers	CTA intern or extern workers	detailed timescale	Total time	
							hours	work days
1. Step	Installation of data centre racks (infrastructure racks for installing patch panels first)	Installation of: 33 racks, electrical connection (incl. Earth cabling), cable trails on the racks	a cleaned data centre room is needed	2	extern	install racks: 1hour/rack justify racks: 1hours/row (2 rows) install cable trails: 3hours/row electrical connection: 1hour/rack trails between rows: 2hours	76	9,5
2. Step	Cabling from the data center to the patch points. That is an iterative process in dependence of the finishing of trenches and power transformer buildings	Installation of patch panel, lay cables, splicing, build cable trails in the data center building, test measuring, lable, generate measurement documents	trenches and patch points are needed	6	extern	lay cables trough a street: 1hour lay cables into the data center building: 2hours lay cables into the patch points: 1hour lay cables: 1hour/km splicing: 1,5hour/12 port patch panel test measurement: 1hour/12 port patch panel labling: 1hour/patch point build cable trails in building: 8hour	~48hours per patch point (8 telescope s)	~6
3. Step	Cabling from the patch points to the telescopes. That is an iterative process in dependence of the finishing of trenches and power transformer buildings	Installation of patch panel, lay cables, splicing, test measuring, lable, generate measurement documents	trenches and patch points are needed	6	extern	lay cables trough a street: 1hour lay cables into the patch points: 2hour lay cables into the telescope: 2hours lay cables: 1hour/km splicing: 1,5hour/12 port patch panel test measurement: 1hour/12 port patch panel labling: 0,5hour/patch point	~65hours per patch point (8 telescope s)	~8
4. Step	Install trigger system (rack mount of trigger)	Not defined yet. In agreement with the trigger group.						
5. Step	Cabling of the buildings, eg network for offices or access points	build cable trails, lay cables (copper inside the buldings and single mode between accommodation and data center building), install RJ45 plugs,		2	extern	**Accommodation:** install 34 x 3 RJ45 plugs and lay cables: 32hours build cable trails: 8hours install patch panels: 4hours **Data center building:** install 13 x 3 RJ45 plugs and lay cables: 16hours build cable trails: 8hours install patch panels: 2hours **Single mode connection between buildings:** lay cables and build cable trails: 8hours (if trenches are ready) install patch panels, test measuring, splicing: 3hours	81	~10
6. Step	Installing of active network components.	physical installation of the network components, define cable ways (data center), without configuration (will be done offsite)	data centre should be clean & temperatured, power should be available	2	intern		48	6
7. Step	Server rack mount & patching	physical installation of servers, patching, without configuration (will be done offsite)		2	intern	installation of the server (incl. unpacking): 1hour/server patching network and management ports: 0,5hour/server	145,5 (whole data center)	~18
8. Step	Completion of the basic infrastructure. (Network, core services,)	installation & configuration of dhcp, dns, configuration management, etc		1	intern		16	2
9. Step	Installation of the operating system for computing nodes, storage	final configuration of the servers (preparation offsite)		1	intern	final configuration: 0,5hours/server	48	6
10. Step	Telescope control validation and verification	testing of OS, connections, etc		2	intern	testing of OS, connections: 8hours/telescope	136	17
11. Step	Installation of camera server	physical installation of the network components, define cable ways (data center), without configuration (will be done offsite)		2	intern	installation of the server (incl. unpacking): 1hour/server patching network and management ports: 0,5hour/server	150	18,75
12. Step	Installation of software for ACTL, camera groups, DATA, etc	Not defined yet. In agreement with the groups in charge.						
13. Step	Installation of the wireless LAN in buildings and the first telescopes	installation of the access points and the controller, without configuration (will be done offsite)		1	intern	installation of the access point: 1hour/access point installation of controller: 1hour/controller	152	19
14. Step	Verification and validation	testing services, test phase with first users, operational		many	intern			

Table 5.1 – Plan for commissioning the on-site data centre and network.

Decommissioning Outline

It is envisaged that the ACTL software shall be available beyond CTA lifetime and become legacy software. This is not only important for the reconstruction of specific time segments of data taking and should be archived together with the historical snapshots of data analysis software. In addition, we aim for contributing to the efforts in astrophysical instrumentation and software engineering and for collaboration with other projects. The open source part of the software (including all documentation) should be available to the broader community for reuse.

The disassembly of all computing, storage and network as well as infrastructure equipment will be done in reverse order of their buildup. During the disassembly, no health-relevant security risks are expected. Only in the case no other experiment can make use of the equipment and recycling renders uneconomical, an environmentally friendly and secure disposal of the electronic equipment and peripherals will be provided on site. The actual rules and laws for data centres redeployment at the appropriate CTA site shall be followed.

5.2 Management Structures

The CTA Observatory is broken down into separate PBS branches and the management plan establishes one sub-project for each PBS branch. The “Array Control” product (PBS id 3) is a first-level item of the CTA Product Breakdown Structure and is dealt with by a corresponding “Array Control and Data Acquisition” (ACTL) project. The ACTL project is further developed in a dedicated ACTL PBS according to a hierarchical definition of corresponding sub-products. The main classes of sub-products are:

- ACTL-DOC: Documentation
- ACTL-SLOW: Instrument Slow Control Software
- ACTL-OPS: Instrument Operation Software
- ACTL-DAQ: Data Acquisition Software
- ACTL-TRIG: Array Trigger and Clock Distribution
- ACTL-ONSITE: Onsite IT infrastructure
- ACTL-SCHED: Central Scheduler

ACTL has responsibility for delivering these products and established for each sub-product a corresponding work package.

In the following sections, the Product Breakdown Structure (PBS) and the Work Breakdown Structure (WBS) are shown. The presented structures are the current snapshots, created to define the full list of products and work to be done for the different work packages.

5.2.1 Product Breakdown Structure

The first four levels of the ACTL project PBS are shown in Fig. 5.1 for the production phase. The PBS elements of the first level are:

- *PBS 3.0:* The *ACTL-DOC* sub-product includes all ACTL-internal documentation as well as all ACTL documents, which are CTA deliverables.
- *PBS 3.1:* The *ACTL-SLOW* sub-product comprises all tools and guidelines for implementing the control software of all the low-level components and sub-systems in the CTA framework. The list of devices to be integrated comprises all sub-systems (assemblies) of a telescope, including Cherenkov cameras, and any auxiliary devices used for e.g. weather monitoring and/or calibration.

- *PBS 3.2:* The *ACTL-OPS* sub-product contains the high-level ACTL functionalities, in particular the delivery of the Central Array Control and Operation software.
- *PBS 3.3:* The *ACTL-DAQ* sub-product is the central module of the Data Acquisition (DAQ) pipeline. It is responsible for interconnecting the camera servers, central trigger and real time pipeline. To help reduce the amount of raw data stored in the on-site repository, it is desirable that the DAQ is modular to allow for real time pipeline processes to occur early in the acquisition pipeline.
- *PBS 3.4:* The *ACTL-TRIG* sub-product covers the array trigger and clock distribution systems.
- *PBS 3.5:* The *ACTL-ONSITE* (also known as ICT; Information and Communications Technology) sub-product comprises the hardware for on-site computing, storage, communications infrastructure as well as the basic environment to operate the on-site systems (e.g. system administration).
- *PBS 3.6:* The *ACTL-SCHED* sub-product contains all software necessary for the scheduling of the observations and other telescope-related tasks. The main purpose of the Central Scheduler for CTA is the allocation of multiple tasks to one single array or to multiple sub-arrays of telescopes, while optimizing the use of the available resources and maximizing the scientific return. This application will consider a two step process: science planning will perform the optimization process for time allocation of tasks, while schedule planning will translate the selected tasks into a detailed task pattern ready to be transferred to the Array Control system (ACTL-OPS) for their execution.

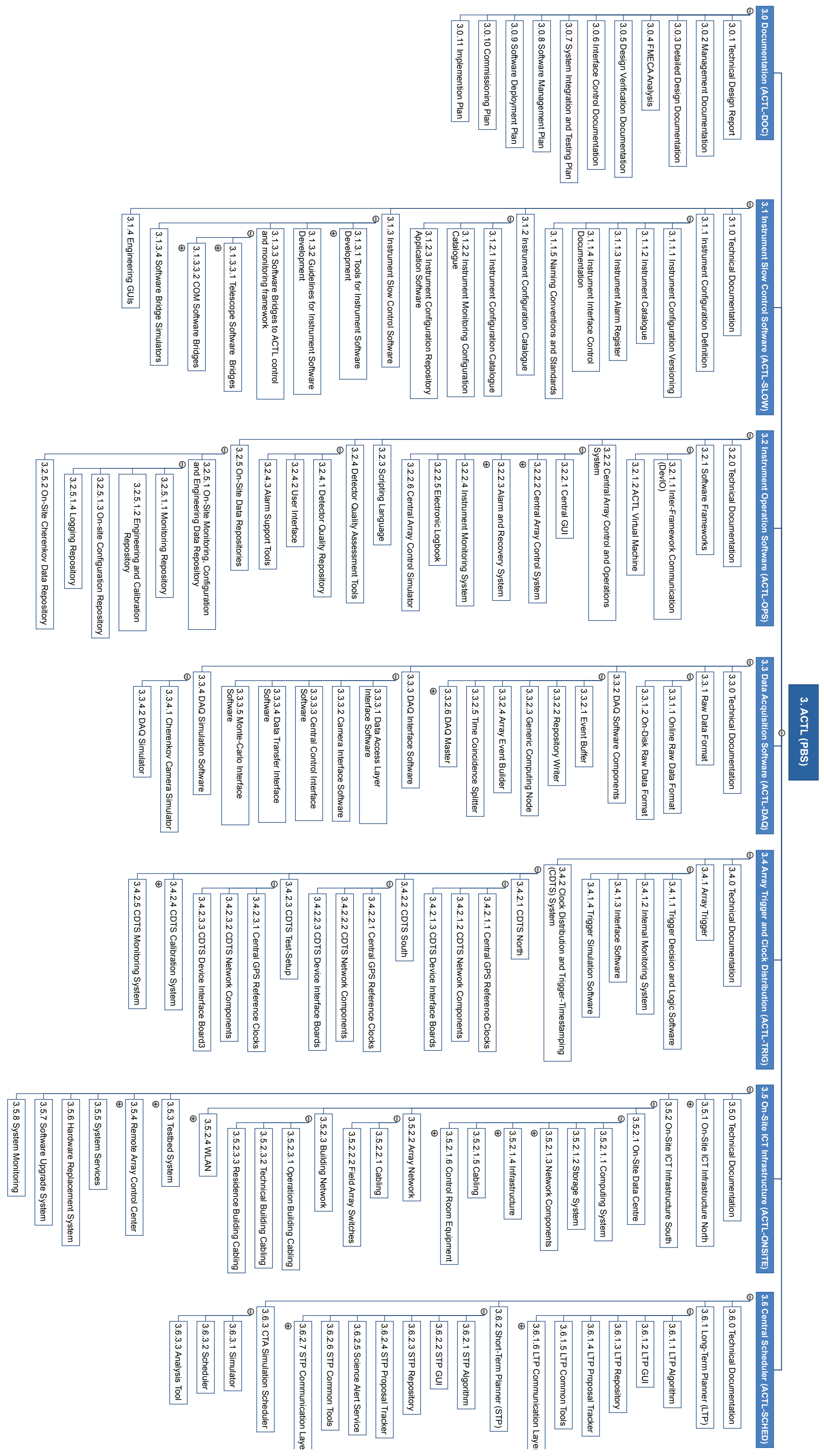

3. ACTL (PBS)
3.0 Documentation (ACTL-DOC)
3.0.1 Technical Design Report
3.0.2 Management Documentation
3.0.3 Detailed Design Documentation
3.0.4 FMECA Analysis
3.0.5 Design Verification Documentation
3.0.6 Interface Control Documentation
3.0.7 System Integration and Testing Plan
3.0.8 Software Management Plan
3.0.9 Software Deployment Plan
3.0.10 Commissioning Plan
3.0.11 Implemention Plan
3.1 Instrument Slow Control Software (ACTL-SLOW)
3.1.0 Technical Documentation
3.1.1 Instrument Configuration Definition
3.1.1.1 Instrument Configuration Versioning
3.1.1.2 Instrument Catalogue
3.1.1.3 Instrument Alarm Register
3.1.1.4 Instrument Interface Control Documentation
3.1.1.5 Naming Conventions and Standards
3.1.2 Instrument Configuration Catalogue
3.1.2.1 Instrument Configuration Catalogue
3.1.2.2 Instrument Monitoring Configuration Catalogue
3.1.2.3 Instrument Configuration Repository Application Software
3.1.3 Instrument Slow Control Software
3.1.3.1 Tools for Instrument Software Development
3.1.3.2 Guidelines for Instrument Software Development
3.1.3.3 Software Bridges to ACTL control and monitoring framework
3.1.3.3.1 Telescope Software Bridges
3.1.3.3.2 COM Software Bridges
3.1.3.4 Software Bridge Simulators
3.1.4 Engineering GUIs
3.2 Instrument Operation Software (ACTL-OPS)
3.2.0 Technical Documentation
3.2.1 Software Frameworks
3.2.1.1 Inter-Framework Communication (DevIO)
3.2.1.2 ACTL Virtual Machine
3.2.2 Central Array Control and Operations System
3.2.2.1 Central GUI
3.2.2.2 Central Array Control System
3.2.2.3 Alarm and Recovery System
3.2.2.4 Instrument Monitoring System
3.2.2.5 Electronic Logbook
3.2.2.6 Central Array Control Simulator
3.2.3 Scripting Language
3.2.4 Detector Quality Assessment Tools
3.2.4.1 Detector Quality Repository
3.2.4.2 User Interface
3.2.4.3 Alarm Support Tools
3.2.5 On-Site Data Repositories
3.2.5.1 On-Site Monitoring, Configuration and Engineering Data Repository
3.2.5.1.1 Monitoring Repository
3.2.5.1.2 Engineering and Calibration Repository
3.2.5.1.3 On-site Configuration Repository
3.2.5.1.4 Logging Repository
3.2.5.2 On-Site Cherenkov Data Repository
3.3 Data Acquisition Software (ACTL-DAQ)
3.3.0 Technical Documentation
3.3.1 Raw Data Format
3.3.1.1 Online Raw Data Format
3.3.1.2 On-Disk Raw Data Format
3.3.2 DAQ Software Components
3.3.2.1 Event Buffer
3.3.2.2 Repository Writer
3.3.2.3 Generic Computing Node
3.3.2.4 Array Event Builder
3.3.2.5 Time Coincidence Splitter
3.3.2.6 DAQ Master
3.3.3 DAQ Interface Software
3.3.3.1 Data Access Layer Interface Software
3.3.3.2 Camera Interface Software
3.3.3.3 Central Control Interface Software
3.3.3.4 Data Transfer Interface Software
3.3.3.5 Monte-Carlo Interface Software
3.3.4 DAQ Simulation Software
3.3.4.1 Cherenkov Camera Simulator
3.3.4.2 DAQ Simulator
3.4 Array Trigger and Clock Distribution (ACTL-TRIG)
3.4.0 Technical Documentation
3.4.1 Array Trigger
3.4.1.1 Trigger Decision and Logic Software
3.4.1.2 Internal Monitoring System
3.4.1.3 Interface Software
3.4.1.4 Trigger Simulation Software
3.4.2 Clock Distribution and Trigger-Timestamping (CDTS) System
3.4.2.1 CDTS North
3.4.2.1.1 Central GPS Reference Clocks
3.4.2.1.2 CDTS Network Components
3.4.2.1.3 CDTS Device Interface Boards
3.4.2.2 CDTS South
3.4.2.2.1 Central GPS Reference Clocks
3.4.2.2.2 CDTS Network Components
3.4.2.2.3 CDTS Device Interface Boards
3.4.2.3 CDTS Test-Setup
3.4.2.3.1 Central GPS Reference Clocks
3.4.2.3.2 CDTS Network Components
3.4.2.3.3 CDTS Device Interface Board3
3.4.2.4 CDTS Calibration System
3.4.2.5 CDTS Monitoring System
3.5 On-Site ICT Infrastructure (ACTL-ONSITE)
3.5.0 Technical Documentation
3.5.1 On-Site ICT Infrastructure North
3.5.2 On-Site ICT Infrastructure South
3.5.2.1 On-Site Data Centre
3.5.2.1.1 Computing System
3.5.2.1.2 Storage System
3.5.2.1.3 Network Components
3.5.2.1.4 Infrastructure
3.5.2.1.5 Cabling
3.5.2.1.6 Control Room Equipment
3.5.2.2 Array Network
3.5.2.2.1 Cabling
3.5.2.2.2 Field Array Switches
3.5.2.3 Building Network
3.5.2.3.1 Operation Building Cabling
3.5.2.3.2 Technical Building Cabling
3.5.2.3.3 Residence Building Cabling
3.5.2.4 WLAN
3.5.3 Testbed System
3.5.4 Remote Array Control Center
3.5.5 System Services
3.5.6 Hardware Replacement System
3.5.7 Software Upgrade System
3.5.8 System Monitoring
3.6 Central Scheduler (ACTL-SCHED)
3.6.0 Technical Documentation
3.6.1 Long-Term Planner (LTP)
3.6.1.1 LTP Algorithm
3.6.1.2 LTP GUI
3.6.1.3 LTP Repository
3.6.1.4 LTP Proposal Tracker
3.6.1.5 LTP Common Tools
3.6.1.6 LTP Communication Laye
3.6.2 Short-Term Planner (STP)
3.6.2.1 STP Algorithm
3.6.2.2 STP GUI
3.6.2.3 STP Repository
3.6.2.4 STP Proposal Tracker
3.6.2.5 Science Alert Service
3.6.2.6 STP Common Tools
3.6.2.7 STP Communication Laye
3.6.3 CTA Simulation Scheduler
3.6.3.1 Simulator
3.6.3.2 Scheduler
3.6.3.3 Analysis Tool

Figure 5.1 – First four levels of product breakdown structure of the ACTL project for the production phase.

5.2.2 Organization Chart

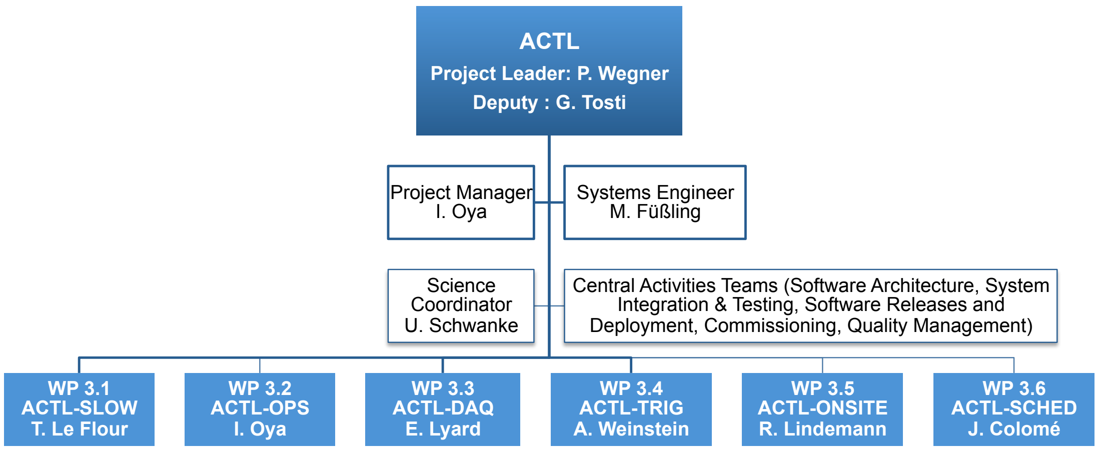

ACTL
Project Leader: P. Wegner
Deputy : G. Tosti
Project Manager I. Oya
Systems Engineer M. Füßling
Science Coordinator U. Schwanke
Central Activities Teams (Software Architecture, System Integration & Testing, Software Releases and Deployment, Commissioning, Quality Management)
WP 3.1 ACTL-SLOW T. Le Flour
WP 3.2 ACTL-OPS I. Oya
WP 3.3 ACTL-DAQ E. Lyard
WP 3.4 ACTL-TRIG A. Weinstein
WP 3.5 ACTL-ONSITE R. Lindemann
WP 3.6 ACTL-SCHED J. Colomé

Figure 5.2 – Organization chart of ACTL

The ACTL work package is subdivided into dedidacted sub-structures, which are reflecting the product and work breakdown structures. The work package is led by a management team consisting of the ACTL convener, an ACTL deputy convener, a project manager, a systems engineer, a science coordinator and dedicated teams for the central activities (software architecture, system integration and testing, etc.). The ACTL convener and its deputy coordinate all activities of the sub-work package leaders, which include planning of the sub-work package activities and design, implementation, testing, commissioning and documentation of all sub-work package products. The convener and its deputy are assisted by the project manager, whose activities further include the management of the overall ACTL schedule and costs. The sub-work package leaders are supported by a systems engineer, responsible for the technical coordination of all sub-project and software engineering activities in addition to the management of documentation, interface control descriptions, quality and RAMS. The science coordinator is the link to other work packages on science-related issues (e.g. on data rates and volumes, science verification) and supports the sub-work package leaders in addressing these issues in the ACTL project. For all central activities, which cross the work-package boundaries and are of cross-cutting concern (e.g. software architecture, use cases, tooling), dedicated groups are being formed, that support all activities in the sub-work packages related to these central activities. An overview of the different roles can be found in Sec. 1.4.

The current organizational structure is foreseen to be further developed, taking into account the revised PBS/WBS, to include further roles for common tasks and to support the ACTL project throughout the construction phase.

5.2.3 Work Breakdown Structure

The work breakdown structure (WBS) aims to provide a decomposition of activities required to achieve the PBS items. The WBS is useful for planning and for organizational purposes of the Array Control project. The WBS is organized around the primary products of the array control project. The sub-work package activities are shown in Fig. 5.3 and are displayed to lower level for some of the main activities. Additional high-level and sub-work package cross-cutting central activities have been identified, including project coordination and system engineering activities, which are displayed in Fig. 5.4.

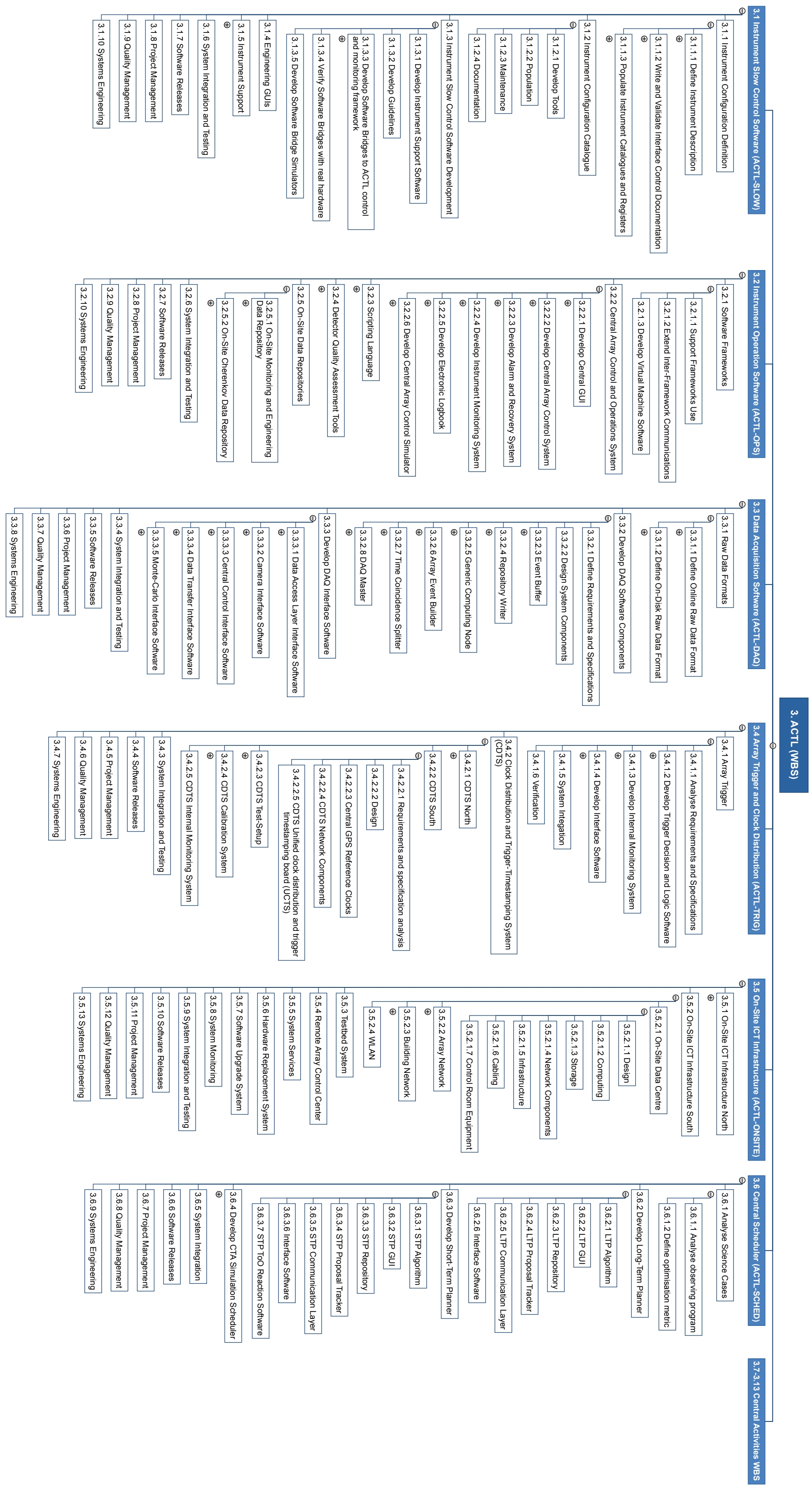

3. ACTL (WBS)
3.1 Instrument Slow Control Software (ACTL-SLOW)
3.1.1 Instrument Configuration Definition
3.1.1.1 Define Instrument Description
3.1.1.2 Write and Validate Interface Control Documentation
3.1.1.3 Populate Instrument Catalogues and Registers
3.1.2 Instrument Configuration Catalogue
3.1.2.1 Develop Tools
3.1.2.2 Population
3.1.2.3 Maintenance
3.1.2.4 Documentation
3.1.3 Instrument Slow Control Software Development
3.1.3.1 Develop Instrument Support Software
3.1.3.2 Develop Guidelines
3.1.3.3 Develop Software Bridges to ACTL control and monitoring framework
3.1.3.4 Verify Software Bridges with real hardware
3.1.3.5 Develop Software Bridge Simulators
3.1.4 Engineering GUIs
3.1.5 Instrument Support
3.1.6 System Integration and Testing
3.1.7 Software Releases
3.1.8 Project Management
3.1.9 Quality Management
3.1.10 Systems Engineering
3.2 Instrument Operation Software (ACTL-OPS)
3.2.1 Software Frameworks
3.2.1.1 Support Frameworks Use
3.2.1.2 Extend Inter-Framework Communications
3.2.1.3 Develop Virtual Machine Software
3.2.2 Central Array Control and Operations System
3.2.2.1 Develop Central GUI
3.2.2.2 Develop Central Array Control System
3.2.2.3 Develop Alarm and Recovery System
3.2.2.4 Develop Instrument Monitoring System
3.2.2.5 Develop Electronic Logbook
3.2.2.6 Develop Central Array Control Simulator
3.2.3 Scripting Language
3.2.4 Detector Quality Assessment Tools
3.2.5 On-Site Data Repositories
3.2.5.1 On-Site Monitoring and Engineering Data Repository
3.2.5.2 On-Site Cherenkov Data Repository
3.2.6 System Integration and Testing
3.2.7 Software Releases
3.2.8 Project Management
3.2.9 Quality Management
3.2.10 Systems Engineering
3.3 Data Acquisition Software (ACTL-DAQ)
3.3.1 Raw Data Formats
3.3.1.1 Define Online Raw Data Format
3.3.1.2 Define On-Disk Raw Data Format
3.3.2 Develop DAQ Software Components
3.3.2.1 Define Requirements and Specifications
3.3.2.2 Design System Components
3.3.2.3 Event Buffer
3.3.2.4 Repository Writer
3.3.2.5 Generic Computing Node
3.3.2.6 Array Event Builder
3.3.2.7 Time Coincidence Splitter
3.3.2.8 DAQ Master
3.3.3 Develop DAQ Interface Software
3.3.3.1 Data Access Layer Interface Software
3.3.3.2 Camera Interface Software
3.3.3.3 Central Control Interface Software
3.3.3.4 Data Transfer Interface Software
3.3.3.5 Monte-Carlo Interface Software
3.3.4 System Integration and Testing
3.3.5 Software Releases
3.3.6 Project Management
3.3.7 Quality Management
3.3.8 Systems Engineering
3.4 Array Trigger and Clock Distribution (ACTL-TRIG)
3.4.1 Array Trigger
3.4.1.1 Analyse Requirements and Specifications
3.4.1.2 Develop Trigger Decision and Logic Software
3.4.1.3 Develop Internal Monitoring System
3.4.1.4 Develop Interface Software
3.4.1.5 System Integration
3.4.1.6 Verification
3.4.2 Clock Distribution and Trigger Timestamping System (CDTS)
3.4.2.1 CDTS North
3.4.2.2 CDTS South
3.4.2.2.1 Requirements and specification analysis
3.4.2.2.2 Design
3.4.2.2.3 Central GPS Reference Clocks
3.4.2.2.4 CDTS Network Components
3.4.2.2.5 CDTS Unified clock distribution and trigger timestamping board (UCTS)
3.4.2.3 CDTS Test Setup
3.4.2.4 CDTS Calibration System
3.4.2.5 CDTS Internal Monitoring System
3.4.3 System Integration and Testing
3.4.4 Software Releases
3.4.5 Project Management
3.4.6 Quality Management
3.4.7 Systems Engineering
3.5 On-Site ICT Infrastructure (ACTL-ONSITE)
3.5.1 On-Site ICT Infrastructure North
3.5.2 On-Site ICT Infrastructure South
3.5.2.1 On-Site Data Centre
3.5.2.1.1 Design
3.5.2.1.2 Computing
3.5.2.1.3 Storage
3.5.2.1.4 Network Components
3.5.2.1.5 Infrastructure
3.5.2.1.6 Cabling
3.5.2.1.7 Control Room Equipment
3.5.2.2 Array Network
3.5.2.3 Building Network
3.5.2.4 WLAN
3.5.3 Testbed System
3.5.4 Remote Array Control Center
3.5.5 System Services
3.5.6 Hardware Replacement System
3.5.7 Software Upgrade System
3.5.8 System Monitoring
3.5.9 System Integration and Testing
3.5.10 Software Releases
3.5.11 Project Management
3.5.12 Quality Management
3.5.13 Systems Engineering
3.6 Central Scheduler (ACTL-SCHED)
3.6.1 Analyse Science Cases
3.6.1.1 Analyse observing program
3.6.1.2 Define optimisation metric
3.6.2 Develop Long-Term Planner
3.6.2.1 LTP Algorithm
3.6.2.2 LTP GUI
3.6.2.3 LTP Repository
3.6.2.4 LTP Proposal Tracker
3.6.2.5 LTP Communication Layer
3.6.2.6 Interface Software
3.6.3 Develop Short Term Planner
3.6.3.1 STP Algorithm
3.6.3.2 STP GUI
3.6.3.3 STP Repository
3.6.3.4 STP Proposal Tracker
3.6.3.5 STP Communication Layer
3.6.3.6 Interface Software
3.6.3.7 STP ToO Reaction Software
3.6.4 Develop CTA Simulation Scheduler
3.6.5 System Integration
3.6.6 Software Releases
3.6.7 Project Management
3.6.8 Quality Management
3.6.9 Systems Engineering
3.7-3.13 Central Activities WBS

Figure 5.3 – Work package activities of the work breakdown structure of the ACTL project for the production phase.

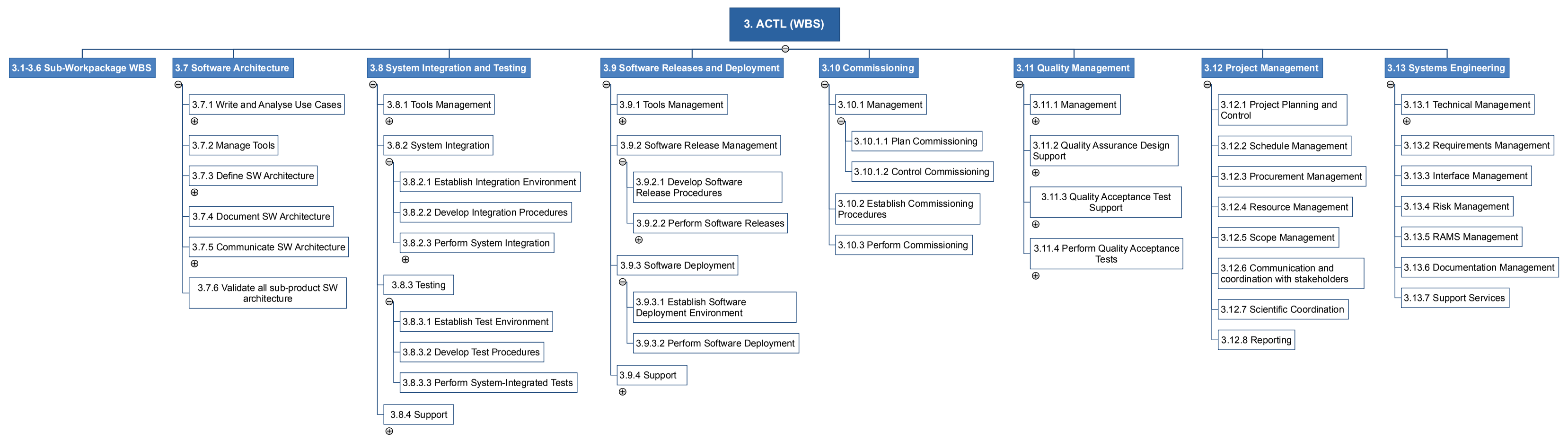

3. ACTL (WBS)
3.1-3.6 Sub-Workpackage WBS
3.7 Software Architecture
3.7.1 Write and Analyse Use Cases
3.7.2 Manage Tools
3.7.3 Define SW Architecture
3.7.4 Document SW Architecture
3.7.5 Communicate SW Architecture
3.7.6 Validate all sub-product SW architecture
3.8 System Integration and Testing
3.8.1 Tools Management
3.8.2 System Integration
3.8.2.1 Establish Integration Environment
3.8.2.2 Develop Integration Procedures
3.8.2.3 Perform System Integration
3.8.3 Testing
3.8.3.1 Establish Test Environment
3.8.3.2 Develop Test Procedures
3.8.3.3 Perform System-Integrated Tests
3.8.4 Support
3.9 Software Releases and Deployment
3.9.1 Tools Management
3.9.2 Software Release Management
3.9.2.1 Develop Software Release Procedures
3.9.2.2 Perform Software Releases
3.9.3 Software Deployment
3.9.3.1 Establish Software Deployment Environment
3.9.3.2 Perform Software Deployment
3.9.4 Support
3.10 Commissioning
3.10.1 Management
3.10.1.1 Plan Commissioning
3.10.1.2 Control Commissioning
3.10.2 Establish Commissioning Procedures
3.10.3 Perform Commissioning
3.11 Quality Management
3.11.1 Management
3.11.2 Quality Assurance Design Support
3.11.3 Quality Acceptance Test Support
3.11.4 Perform Quality Acceptance Tests
3.12 Project Management
3.12.1 Project Planning and Control
3.12.2 Schedule Management
3.12.3 Procurement Management
3.12.4 Resource Management
3.12.5 Scope Management
3.12.6 Communication and coordination with stakeholders
3.12.7 Scientific Coordination
3.12.8 Reporting
3.13 Systems Engineering
3.13.1 Technical Management
3.13.2 Requirements Management
3.13.3 Interface Management
3.13.4 Risk Management
3.13.5 RAMS Management
3.13.6 Documentation Management
3.13.7 Support Services

Figure 5.4 – Central activities of the work breakdown structure of the ACTL project for the production phase.

5.2.4 Schedule

A simplified version of the overall ACTL schedule is presented. The schedule is still under revision. The schedule was created under the assumption that (i) all basic infrastructure to begin deployment of the on-site computing, storage and network insfrastructure is available end of 2016 and (ii) the full funding is secured for the construction phase. The schedule is split into three parts. In Fig. 5.5 and Fig. 5.6, the software-related activities are shown together with the first milestones. It should be noted that for all of these activites, a refinement of the architecture and iterative design is planned, with growing functionality and complexity of the software products over time. In this simplified version, these steps have been omitted. In addition, from the start of first observations with the telescopes on-going operation and maintenance effort are expected for the software (and hardware) products. These activities are also not shown in the schedule. In Fig. 5.7, the schedule for the hardware-related activities are shown for the initial buildup phase. The stageing of the ACTL hardware products (on-site ICT infrastructure and clock distribution system) as a function of telescope deployment is also not shown in the schedule.

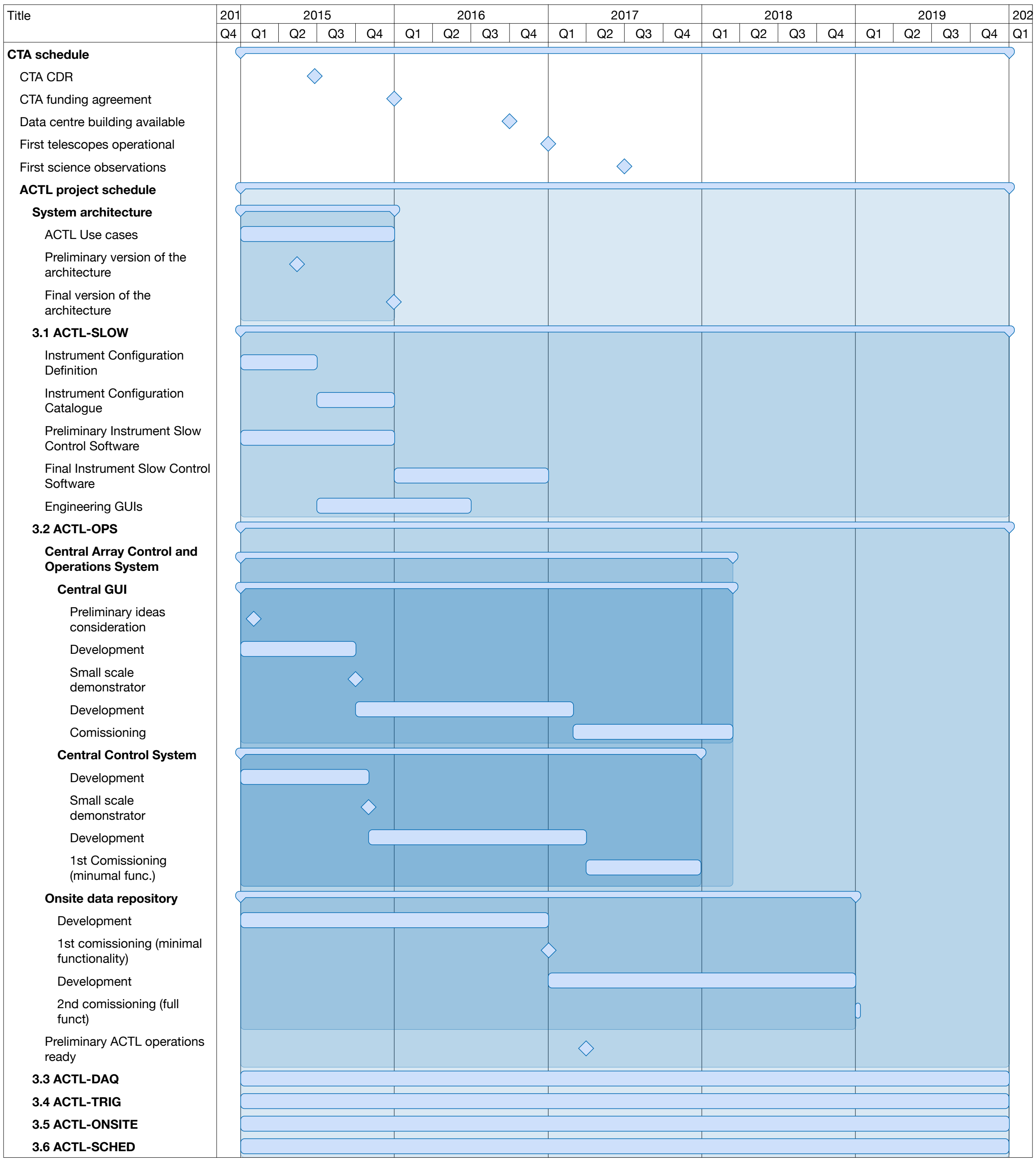


Figure 5.5 – Reduced version of the schedule of ACTL (part 1)

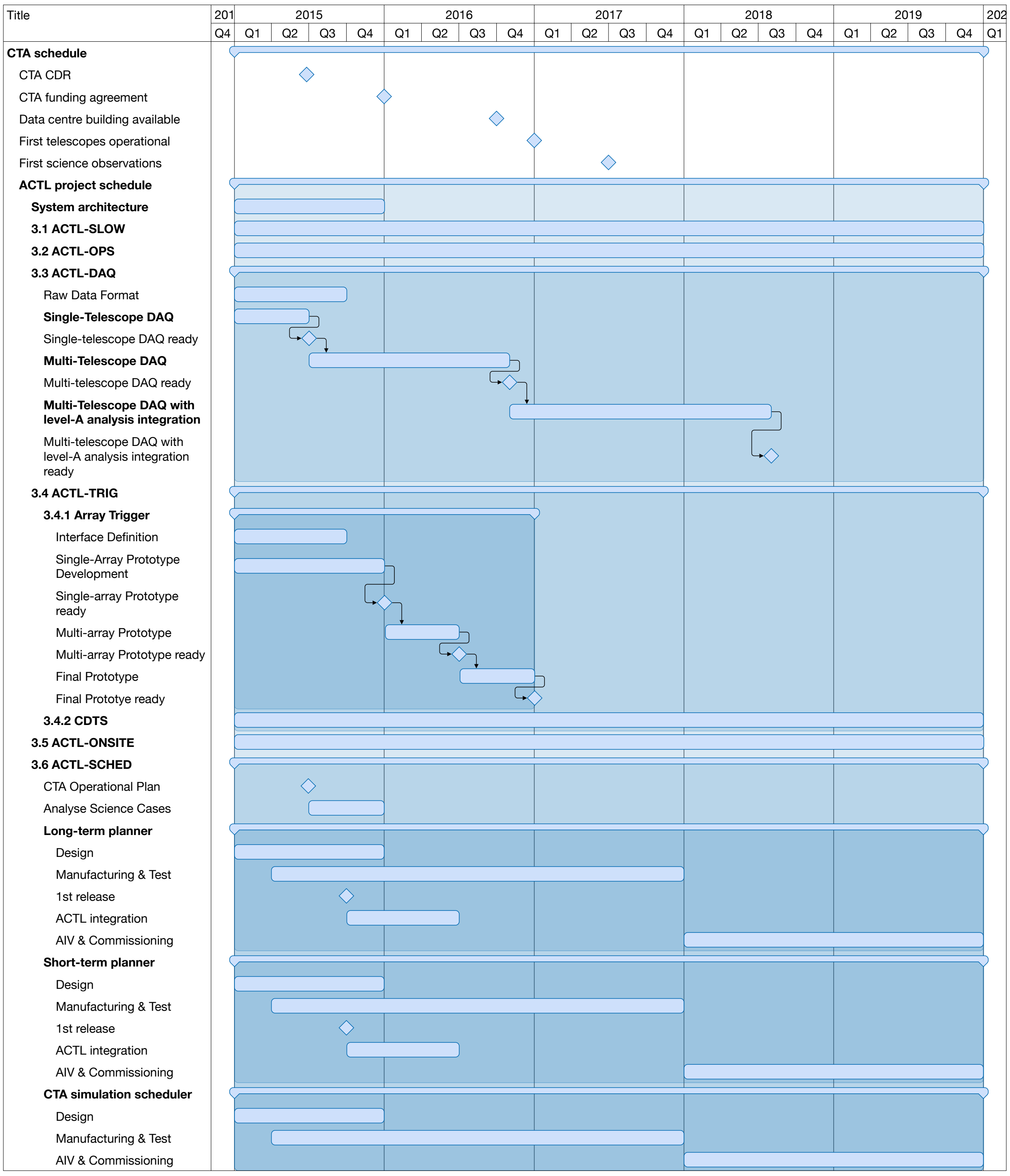


Figure 5.6 – Reduced version of the schedule of ACTL (part 2)

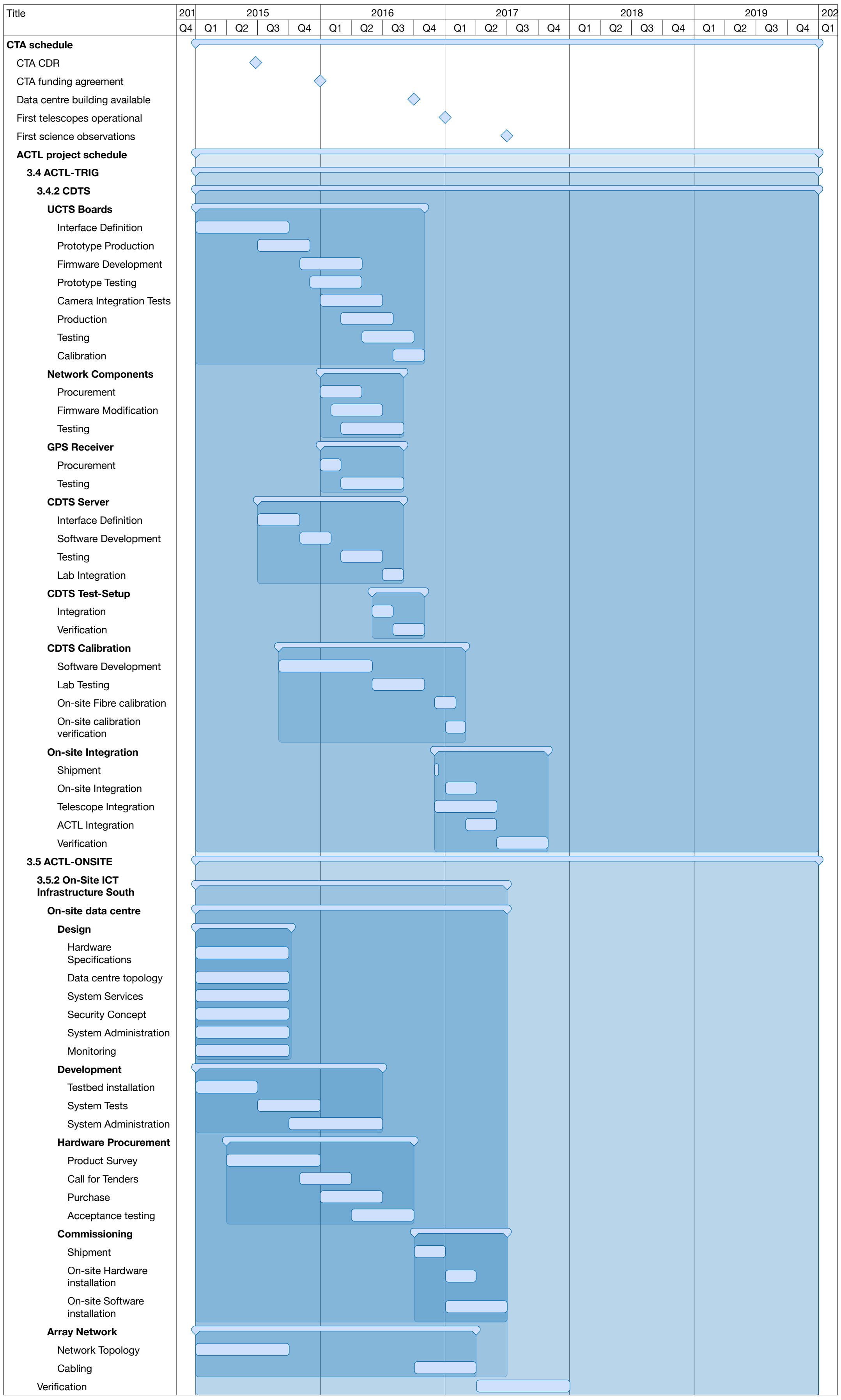


Figure 5.7 – Reduced version of the schedule of ACTL (part 3)

5.3 Milestones

ACTL milestones

Milestone ID	Description	Is a deliverable?	Date			Status	Notes
			Original	Target	Actual		
			3.0 ACTL-DOC				
M0.1	Internal ICDs	No	Q2 2015			Started	
M0.2	External ICDs	No	Q2 2015			In progress	
M0.3	Software deployment plan	No	Q2 2015				
			3.1 ACTL-SLOW				
M1.1	Interfaces with minimum functionality defined	No	Q2 2015				
M1.2	Interfaces with full functionality defined	Yes	Q4 2015				
M1.3	Prototypes for slow control software components	No	Q2 2016				
M1.4	Test of slow control software components in lab finished	No	Q3 2016				
M1.5	Preliminary configuration database	Yes	Q4 2016				
M1.6	Start of commissioning of slow control software	No	Q4 2016				
			3.2 ACTL-OPS				
M2.1	Final list of use cases	No	Q1 2015	Q2 2015		In progress	
M2.2	Operator GUI requirements	No	Q2 2014	Q2 2015		Started	First workshop with external GUI experts date: 25-26 May 2015
M2.3	Intermediate definition of architecture	No	Q2-3 2015			In progress	Collaboration with external software architects
M2.4	Decision on database backend technologies	No	Q3 2015			In progress	Test in test cluster for MongoDB recently started, to be followed by Cassandra tests
M2.5	Small scale demonstrator	No	Q4 2015				Will be built based in the preliminary architecture
M2.6	Final ACTL (OPS) system architecture	No	Q1 2015	Q4 2015			
M2.7	First version of the integral system deployed	Yes	Q2 2017				
M2.8	1st commission of the TMCDB	Yes	Q1 2017				
M2.9	Final system working onsite	Yes	Q4 2019				
M2.10	Deployment of web tools for quality assessment	Yes	Q2 2019				
M2.11	Final commission of the TMCDB	Yes	Q1 2019				
			3.3 ACTL-DAQ				
M3.1	1st single telescope DAQ system	No	Q3 2015			alpha-released	
M3.2	Simple full array level DAQ software	No	Q4 2015			Started	
M3.3	Final camera to DAQ interface definition	No	Q4 2015			Started	
M3.4	Raw data format	No	Q4 2015				
M3.5	Raw data content	No	Q4 2016				
M3.6	1st version of array DAQ with realtime pipeline included	Yes	Q4 2016			Started	
			3.4 ACTL-ONSITE				
M4.1	Definition array network topology	No	Q3 2014	Q2 2015		Started	
M4.2	Basic specifications of computing, network and storage	No	Q3 2014			Done	
M4.3	Full specifications of computing, network and storage	No	Q3 2014	Q2 2015		In progress	
M4.4	Definition Datacenter topology	No	Q4 2014	Q2 2015		In progress	some dependencies concerning the site and the buildings are open
M4.5	Security conceptual design	No	Q4 2014	Q2 2015		In progress	
M4.6	Install a testbed	Yes	Q1 2015			Done	Missing some optical fibres and Nics only.
M4.7	Definition of monitoring and services architecture	No	Q2 2015				
M4.8	Start writing requirement lists for purchasing	No	Q3 2015				
M4.9	Start procurement	No	Q4 2015				
M4.10	Configuration of network hardware	Yes	Q2 2016				
M4.11	Configuration of computing hardware	Yes	Q2 2016				
M4.12	Start building cabling	Yes	Q3 2016				
M4.13	Start array cabling	Yes	Q3 2016				
M4.14	Installation of onsite network	Yes	Q1 2017				
M4.15	Installation of computing hardware (initial stage)	Yes	Q1 2017				
M4.16	Installation of the remote array control center	Yes	Q2 2017				
M4.17	Installation of final hardware	Yes	Q3 2018				
			3.5 ACTL-TRIG				
M5.1	Lab tests of SWAT started	No	Q4 2015			Started	
M5.2	Full specifications of SWAT system finalized	No	Q4 2015			Started	
M5.3	CDTS-Camera Interface definition finalized	No	Q4 2015			Started	
M5.4	Interface definition between CDTS system and SWAT finalizec	No	Q4 2015			Started	
M5.5	Defined calibration procedure for CDTS system	No	Q4 2015			Started	
M5.6	Full specifications of CDTS system finalized	No	Q4 2015			Started	
M5.7	White Rabbit switch procedurement done	Yes	Q2 2016				
M5.8	UCTS / camera integration tests with prototype	No	Q2 2016				
M5.9	UCTS prototype complete and tested	No	Q2 2016				
M5.10	SWAT multi-array prototype complete	No	Q2 2016				
M5.11	Off-the-shelf hardware testing (GPS, WR switches) complete	No	Q3 2016				
M5.12	Integration-test of CDTS with SWAT complete	Yes	Q3 2016				
M5.13	UCTS board production complete	No	Q3 2016				
M5.14	Lab tests of UCTS board done	No	Q4 2016				
M5.15	SWAT development complete	Yes	Q4 2016				
M5.16	All CDTS lab tests and calibration procedures complete	No	Q4 2016				
M5.17	Installation of CDTS system on site started	Yes	Q4 2016				
M5.18	Integration of SWAT board with camera servers done	No	Q4 2016				
M5.19	Integration of SWAT board with DAQ done	No	Q4 2016				
M5.20	Installation of SWAT system on site started	Yes	Q1 2017				
M5.21	Onsite fiber calibration and verification complete	?	Q1 2017				
M5.22	CDTS - Telescope integration complete	Yes	Q2 2017				
M5.23	CDTS system completed	Yes	Q3 2017				
			3.6 ACTL-SCHED				
M6.1	Requirements extraction from CTA operational plan	No	Q1 2015	Q2 2015		Started	
M6.2	Final specifications	Yes	Q2 2015			Started	
M6.3	Final interface definition with ACTL-OPS	Yes	Q3 2015				
M6.4	Release of version 0.1	No	Q3 2015				Including SCHED-OPS interface and basic scheduling functionalities (STP algorithm without dynamic optimization, SB delivery to OPS)
M6.5	Partial integration in ACTL-OPS	No	Q4 2015				
M6.6	Final interface defintion with DATA	Yes	Q1 2016				Interface Control Document
M6.7	Release of version 0.2	No	Q2 2016				Including SCHED-DATA/OBS interface and basic scheduling functionalities (LTP), and Simulation scheduler
M6.8	Release of version 0.3	No	Q3 2016				Including preliminary version of the Simulation scheduler with completed functionalities (LTP and STP simulator, analysis tool, etc.).
M6.9	Partial integration with DATA-OBS	No	Q4 2016				
M6.10	Release of version 0.4	No	Q1 2017				Including LTP & STP: algorithms, archive (DBs), proposal tracker, common tools and communication layer
M6.11	Release of version 0.5	No	Q1 2017				Including STP GUI: preliminary version
M6.12	Continued integration in ACTL-OPS	No	Q3 2017				Goal: testing integration of STP GUI, start commissioning of the STP software with first telescopes on-site
M6.13	Release of version 0.6	No	Q1 2018				Including LTP GUI: preliminary version
M6.14	Release of version 0.7	No	Q2 2018				Including STP Science Alert Service: preliminary version
M6.15	Continued integration in ACTL-OPS	No	Q3 2018				Goal: testing integration of SciAlert Service and report commissioning results.
M6.16	Release of version 0.8	No	Q1 2019				Including LTP & STP & Simulation Scheduler - complete version
M6.17	Final commissioning with ACTL-OPS	No	Q2 2019				Final system working on-site
M6.18	Final commissioning with DATA	No	Q2 2019				Final system working off-site (integration of the LTP with the Observatory Planning and/or Proposal Handling tools)
M6.19	Release of version 1.0	Yes	Q3 2019				Including LTP & STP & Simulation Scheduler - complete and final version
M6.20	STP and LTP Commissioning Reports	Yes	Q4 2019				Document

Table 5.2 – Preliminary milestones for ACTL.

5.4 Construction costs

This section describes the cost estimates for the construction phase of the ACTL project and is split in two parts. The first part summarizes the labour and software construction costs. These estimates include all activities related to project management and central activities, activities related to the design, production and procurement of hardware systems as well as software development efforts. Software construction costs are given as equivalents of manpower costs, but procurement of software products from industry is anticipated. The second part summarizes the investment costs for all ACTL hardware products.

5.4.1 Labour and software construction costs

Cost estimation procedure

The requirements on the array control and data acquisition software of CTA are demanding compared to existing Cherenkov telescope experiments regarding reliability, availability and complexity, and are comparable to large-scale astronomical facilities such as ALMA or SKA. For such complex systems, an estimation of the needed manpower for software development is not an easy task. Various algorithms exist for estimating the manpower needed for software projects, such as the Constructive Cost Model (COCOMO) algorithm [37], but cannot be easily adapted for the estimation of manpower for ACTL. The usage of existing software frameworks such as OPC UA and ACS reduces the overall needed manpower for software development. At the same time, the estimation of the needed manpower for tailoring and adaptation of the frameworks for CTA needs is more complex. In addition, a significant fraction of manpower needed within ACTL will be used for integration of telescopes and devices, including extended testing and debugging purposes, to reach the required availability of the CTA observatory.

First estimates on the total manpower have been made based on estimates for ALMA and SKA [38], scaled to the complexity of CTA and subtracting the costs for framework development. These estimates have been verified with a WBS-based approach using the preliminary software architecture, where experiences with current software prototypes have been taken into account.

Labour and software development costs

The total estimated manpower is about 165 FTE for the construction phase, corresponding to a total net value of about 13M €, see also Tab. 5.3. An average salary of 78k € was assumed to derive the total costs. These cost estimates do not include any travel expenses needed for e.g. workshops, instrument and user support, testing with hardware devices or commissioning. A revision of these cost estimates on the basis of a refined software architecture (see Sect. 5.1.1) is ongoing.

		Total Costs		
		FTE	FTEy	Costs
WBS Item				
3	**Array Control and Data Acquisition**	**33**	**165**	**12.870.000 €**
3.1	Instrument Slow Control Software (ACTL-SLOW)	6,0	30,0	2.340.000 €
3.2	Instrument Operations Software (ACTL-OPS)	8,0	40,0	3.120.000 €
3.3	Data Acquisition Software (ACTL-DAQ)	4,0	20,0	1.560.000 €
3.4	Array Trigger and Clock Distribution (ACTL-TRIG)	4,0	20,0	1.560.000 €
3.5	On-site ICT Infrastructure (ACTL-ONSITE)	3,0	15,0	1.170.000 €
3.6	Central Scheduler (ACTL-SCHED)	4,0	20,0	1.560.000 €
3.7-3.13	Central Activities	4,0	20,0	1.560.000 €

Table 5.3 – Summary of manpower distribution per sub-workpackage. An average yearly salary of 78k € was used to estimate the total costs.

Costs for software licenses

No software license costs are currently planned. The approach taken by ACTL is to use open source software, whenever possible. In the case of OPC UA, software licenses are needed for Software Development Kits and compilation rights. Each development team will be in charge of purchasing the respective licenses in the construction phase of CTA. A document describing the license purchase process is in preparation. In addition, ACTL will offer support for compilation at different platforms.

Costs for tooling supporting software development efforts

Adequate tools for software development as well as for automated integration, building and testing, deployment and maintenance of software are key for assuring the high quality and reliability of the software and enabling a well-defined collaborative software development process. Yearly costs of about 5k € are estimated for the construction phase.

5.4.2 Clock Distribution and trigger-timestamping system construction costs

The hardware components of the clock distribution and trigger-timestamping system consist of white rabbit network switches, GPS reference clocks, unified clock distribution and time-stamping boards and calibration equipment and amount to 190k€. The investment costs are shown in Tab. 5.4 and are based on a survey of existing products and on experience with similar experiments.

			Costs		
WBS Item	**Comment**	**Unit**	**Prize per unit**	**Total**	**% of total**
3.4	**Array Trigger and Clock Distribution (ACTL-TRIG)**				
3.4.2	**Clock Distribution and Trigger-Timestamping System (CDTS)**				
3.4.2.2	CDTS South			189.000 €	100%
3.4.2.2.3	Central GPS reference clocks	2	2.500 €	5.000 €	3%
3.4.2.2.4	CDTS Network Components/Switches	6	4.000 €	24.000 €	13%
3.4.2.2.5	CDTS Device Interface Boards (UCTS)	100	1.500 €	150.000 €	79%
3.4.2.4.3	CDTS Calibration Equipment	1	10.000 €	10.000 €	5%
Total Costs (CTA South Array only)				**189.000 €**	

Table 5.4 – Summary of ACTL-TRIG investment costs.

5.4.3 Information, computing and storage (ICT) infrastructure investment costs

The ICT infrastructure for the on-site installations of the South and North array is provided by ACTL-ONSITE and consists of the on-site data centre (computing, storage and infrastructure as well as network components and cabling), the array network cabling and components and the buildings network cabling and components. An overview on the investment costs for the South array is given in Tab. 5.5 as total costs for the construction phase.

Cost estimation procedure

The estimated construction costs of the ICT infrastructure are based on estimates for typical data centres. The construction costs include only investment costs, excluding manpower (except for the network cabling), travel expenses and spares. The largest uncertainty in the cost estimates stem from the uncertainty of the data rates which impacts the storage and network infrastructure cost estimates.

The cost estimates themselves are based on recent information coming from computer vendors (for CPU cores and storage: offerings from DELL, for network components: offerings from ARISTA and CISCO, for racks and UPS: various vendors). Due to the high availability and reliability requirements high-end

		Total Costs	
		€	% of total
WBS Item			
3.5	On-site ICT Infrastructure (ACTL-ONSITE)		
3.5.2	On-site ICT Infrastructure South		
3.5.2.1	On-site Data Centre	2.245.150 €	66%
3.5.2.1.2	Computing	364.720 €	11%
3.5.2.1.3	Storage	360.000 €	11%
3.5.2.1.4	Network		
	Cabling	36.500 €	1%
	Central Switches	846.650 €	25%
	Management Switches	84.780 €	3%
	Firewall	250.000 €	7%
3.5.2.1.5	Infrastructure		
	UPS	200.000 €	6%
	Rack	40.500 €	1%
	Printer	12.000 €	0%
3.5.2.1.7	Control Room Equipment	50.000 €	1%
3.5.2.2	Array Network	808.400 €	24%
3.5.2.2.2	Array Cabling	688.400 €	20%
3.5.2.2.3	Network Components	120.000 €	4%
3.5.2.3	Building Network	66.400 €	2%
3.5.2.3.1	Operation Building Cabling	21.500 €	1%
3.5.2.3.2	Technical Building Cabling	19.100 €	1%
3.5.2.3.3	Residence Building Cabling	25.800 €	1%
3.5.2.4	WLAN	257.200 €	8%
Total Costs (CTA South Array only)		3.377.150 €	100%

Table 5.5 – Summary of investment costs for the on-site ICT infrastructure. The costs are shown for the South array only.

hardware products are considered, e.g. containing redundant power supplies. Cost estimates for the cabling of the array and buildings will be delivered and installed by a sub-contractor. Therefore, the cost estimates for the cabling include all investment and manpower costs. The cost estimates are given for the southern array only, where a detailed analysis has been done. It is assumed that the costs for the ICT infrastructure at the northern site are a factor of two lower than for the southern array.

For more details about the ICT infrastructure architecture, see Sec. 3.2.5.

Computing and storage investment costs

The estimation of the computing and storage resources has been worked out according to the needs of the different ACTL products (see Sec. 3.2.5). Those estimates are based on experiences coming from other Cherenkov telescope experiments and first prototypes including test installations as well as extrapolations for CTA. The detailed investment costs for the computing and storage resources are shown in Tab. 5.6.

Infrastructure investment costs

The UPS for the data centre will be planned and designed from CTA-INFRA. A detailed cost plan will be part of the final design provided by CTA-INFRA. The 200k € are an estimation from ACTL-ONSITE based on experiences with other existing UPS systems in standard computer centres. ACTL-ONSITE will be responsible for procurement and installation of the UPS. The number of racks are derived from

				Costs		
WBS Item		**Comment**	**Unit**	**Prize per unit**	**Total**	**% of total**
3.5	**On-site ICT Infrastructure (ACTL-ONSITE)**					
3.5.2	**On-site ICT Infrastructure South**					
3.5.2.1	**On-Site Data Centre**					
3.5.2.1.2	**Computing**		**Cores**			
		16 cores per machine	1.552	235 €	364.720 €	100%
		SLOW/OPS Compute node	208	235 €	48.880 €	13%
		DAQ Compute node	400	235 €	94.000 €	26%
		TRIG Compute node (SWAT, CDTS)	32	235 €	7.520 €	2%
		ONSITE Service node	32	235 €	7.520 €	2%
		ONSITE Login node/ workgroup server	64	235 €	15.040 €	4%
		SCHED Compute node	16	235 €	3.760 €	1%
		Level-A/B Analysis Compute node	800	235 €	188.000 €	52%
3.5.2.1.3	**Storage**		**TB**			
		10 x 6 TB disks per server	3.000	120 €	360.000 €	100%
3.5.2.1.4	**Network Components**					
	Central Switches				846.650 €	100%
		16 slot chassis incl. fabrics moduls, power and one supervisor engine	2	73.700 €	147.400 €	17%
		Supervisor engine	2	4.300 €	8.600 €	1%
		Slot card 48 port each 10 Gbit/s copper	15	9.750 €	146.250 €	17%
		Slot card 48 port each 10 Gbit/s optical	8	19.500 €	156.000 €	18%
		SFP SM 10 Gbit/s	120	1.730 €	207.600 €	25%
		SFP SM 1 Gbit/s	226	800 €	180.800 €	21%
	Management Switches				84.780 €	100%
		Switch control room 48 x 10 Gbit/s RJ45, 4 x 10 Gbit/s uplink	18	1.000 €	18.000 €	21%
		Building cabling switches 48 x 1Gbit/s RJ45, 2 x 10 Gbit/s uplink	10	3.000 €	30.000 €	35%
		Management switches data center	3	8.000 €	24.000 €	28%
		SFP for uplinks SM 10 Gbit/s	6	1.730 €	10.380 €	12%
		SFP for uplinks MM 10 Gbit/s	4	600 €	2.400 €	3%
	Firewall				250.000 €	100%
		Firewall	1	250.000 €	250.000 €	100%
3.5.2.1.5	**Infrastructure**					
	UPS			200.000 €	200.000 €	100%
	Racks				40.500 €	100%
			27	1.500 €	40.500 €	100%
	Printer				12.000 €	100%
			4	3.000 €	12.000 €	100%
3.5.2.1.7	**Control Room Equipment**					
					50.000 €	100%
3.5.2.2	**Array Network**					
3.5.2.2.3	**Network Components**					
	Field Array Switches				120.000 €	100%
		24 x 1Gbit/s RJ45, 2 x 10 Gbit/s uplink SFP+	100	1.200 €	120.000 €	100%
Total Costs (CTA South Array only)					**2.328.650 €**	

Table 5.6 – Detailed view of investment costs for the on-site ICT infrastructure components (exc. cabling and WLAN). The costs are shown for the South array only.

the number of servers, network devices, trigger and auxiliary devices as well as from all passive equipment like patch panel. The costs of the racks include all costs for power plugs and mounting material. Furthermore, it is planned to support printer in every building on site. The 12k € for printing are pure investment costs (see Tab. 5.5).

Network components investment costs

The network costs are divided into the following main parts (see also Tab. 5.5): network components, network cabling and WLAN. A detailed overview on the investment costs for the network components is given in Tab. 5.6 for the South array.

The central switches in the on-site data centre will consist of two chassis switches, which will be redundant to each other and belong to the core switch structure. The costs include all 10 Gbit/s copper and optical fibre cards as well as all optical modules for 10 Gbit/s and 1 Gbit/s. The full configuration concerning supervisor engines, fabric modules and power supplies is implied too. This setup supports 126 telescopes with the different requirements regarding all connections between the data centre and the telescopes as well as all connections inside the data centre (computing nodes, storage nodes etc.).

The ACTL field array switches will be located in the telescopes themselves and will collect the data from the WLAN network and all auxiliary devices for the transport to the data centre. It is planned to install the switches inside the SSTs, MSTs and the LSTs.

The part office/control room and part management switches comprise all active components, which are required to connect other buildings (residual building), offices and important rooms like the control room to the data centre.

The costs of the firewall highly depend on the data traffic to the outside world. With the supposed 1 Gbit/s external network the firewall would be cheaper, but until now it is not clear how to transport the data with such a small bandwidth. Hence, a bigger firewall with up to 10 Gbit/s throughput is planned here.

Network cabling and WLAN construction costs

A full overview on the investment costs for the network cabling and WLAN is given in Tab. 5.7.

The Wireless LAN will consist of hotspots at each telescope and coverage of the technical buildings. The whole array will not be covered. For this about 135 access points and a wireless control system will be necessary.

One of the main parts of the overall ACTL-ONSITE investment costs is the network cabling. The cost estimates for the network cabling for the array and buildings are based on design layouts provided by CTA-INFRA. The array network topology as shown in Fig. 3.15 was used for the detailed array cabling costs. In the costs, all the hardware is included which is needed to deliver and lay cables in existing trenches and to connect and install patch panels in the telescopes, data centre or other technical rooms, i.e. the cables (optical and copper), hood joints, splice trays, seal connectors, patch panels and more. Furthermore, all the manpower needed to install, to measure and to document the hardware is included as well. Cabling has to be seen as a product which will be delivered and installed by a sub-contractor. Therefore, the costs are not separated into investment and manpower costs.

Cabling and WLAN Costs		Unit	Costs in €
WBS Item			
3.5	On-site ICT Infrastructure (ACTL-ONSITE)		
3.5.2	On-site ICT Infrastructure South		
3.5.2.2.2 Array cabling (Datacenter --> Telescopes)			
	SM 2x12E 9/125 optical fibre cables: delivery and install into existing trenches	26000 m	124.800 €
	SM 9x24E 9/125 optical fibre cables: delivery and install into existing trenches	19000 m	163.400 €
	Hood Joint: Delivery, install into existing trenches and label	13 pc	9.400 €
	LWL 12 ports patch panel, deliver, install and label	200 pc	60.400 €
	FRECAP MAX wall fixture, deliver and install	13 pc	13.800 €
	LWL mobile patch cable A/I-VQ(ZN)11Y PUR 2G50/125μm deliver and lay / 46m	100 pc	49.800 €
	splicing work in a tent with overpressure for dusty outdoor splicing	11520 pc	172.800 €
	Check and measure the LWL connections	2400	40.000 €
	Freight & cost of carriage sea transport: material, tools & machines etc.	1 pc	12.500 €
	Freight & cost of carriage road transport: material, tools & machines etc.	1 pc	12.500 €
	miscellaneous costs: documentation, Seal Connector, SE Splice tray,etc	-	29.000 €
	Total		**688.400 €**
3.5.2.3.1 Operation building cabling			
	datacable 800 MHz, FR/LSOH, delivery and install in cable pathes	5000 m	10.000 €
	data sockets 3 ports, RJ45, delivery, install and label	29 pc	1.300 €
	24-port 19" patch panels kat6 RJ45, delivery, install and label	4 pc	1.200 €
	cable runs and trenches, cable protection pipes, constructive elements, etc	-	3.500 €
	Freight & cost of carriage sea and road transport: material, tools & machines etc.	1 pc	1.500 €
	miscellaneous costs: documentation, measure,etc	-	4.000 €
	Total		**21.500 €**
3.5.2.3.2 Technical building cabling + connection to data center			
	datacable 800 MHz, FR/LSOH, delivery and install in cable pathes	2200 m	4.400 €
	data sockets 3 ports, RJ45, delivery, install and label	13 pc	500 €
	24-port 19" patch panels kat6 RJ45, delivery, install and label	2 pc	600 €
	cable runs and trenches, cable protection pipes, constructive elements, etc	-	3.000 €
	SM 2x12E 9/125 optical fibre cable: delivery and install into existing trenches	800 m	4.000 €
	12-ports 19" patch panel LWL SC, deliver, install and label	2 pc	600 €
	Freight & cost of carriage sea and road transport: material, tools & machines etc.	1 pc	1.500 €
	miscellaneous costs: documentation, measure, pig tails, splicing, etc	-	4.500 €
	Total		**19.100 €**
3.5.2.3.3 Residence building cabling + connection to data center			
	datacable 800 MHz, FR/LSOH, delivery and install in cable pathes	3100 m	6.200 €
	data sockets 3 ports, RJ45, delivery, install and label	34 pc	1.400 €
	24-port 19" patch panels kat6 RJ45, delivery, install and label	2 pc	600 €
	cable runs and trenches, cable protection pipes, constructive elements, etc	-	3.500 €
	SM 2x12E 9/125 optical fibre cable: delivery and install into existing trenches	1500 m	7.500 €
	12-ports 19" patch panel LWL SC, deliver, install and label	2 pc	600 €
	Freight & cost of carriage sea and road transport: material, tools & machines etc.	1 pc	1.500 €
	miscellaneous costs: documentation, measure, pig tails, splicing, etc	-	4.500 €
	Total		**25.800 €**
3.5.2.1.4 Data center cabling			
	LWL patch cables several length	700 pc	10.000 €
	Copper patch cables several length incl management network	2500 pc	13.000 €
	24-port 19" patch panels kat6 RJ45	60 pc	12.000 €
	cable guiding panels	60 pc	1.500 €
	Total		**36.500 €**
3.5.2.4 WLAN			
	WLAN cables in buildings, delivery and install in cable pathes	2400 m	4.800 €
	data sockets 2 ports, RJ45, delivery, install and label	26 pc	900 €
	Access Points Outdoor	105 pc	157.500 €
	Access Points Indoor	30 pc	24.000 €
	Controller	2 pc	65.000 €
	Freight & cost of carriage sea and road transport: material, tools & machines etc.	1 pc	5.000 €
	Total		**257.200 €**
Total (South Array only)			**1.048.500 €**

Table 5.7 – Detailed view on the investment costs (including manpower) for the network cabling and WLAN.

Spending profile

A possible spending profile for the on-site ICT infrastructure (excl. network cabling) is given in Tab. 5.8. The given cost estimates include spares (see also Sec. 5.5.2). The spending profile is given for two different scenarios where it is assumed that in scenario 2 a data reduction factor of 10 is reached already at an early stage, while for scenario 1 the data reduction factor will evolve slowly over time. The spending profile is given the 6 stages of telescope grouping to be deployed at the site and as described in [39]. For the spending profile of the network cabling it is assumed that it will follow closely the site development and build-up of the array.

Spending profile (incl. spares)		1. Step Costs in k€	2. Step Costs in k€	3. Step Costs in k€	4. Step Costs in k€	5. Step Costs in k€	6.Step Costs in k€	Total Costs in k€
Scenario 1								
	Storage	205	36	36	36	36	36	385
	Computing	90	60	60	60	60	60	390
	Network Components	400	245	245	245	245	245	1625
	Total	**695**	**341**	**341**	**341**	**341**	**341**	**2400**
Scenario 2								
	Storage	80	61	61	61	61	61	385
	Computing	75	65	63	63	62	62	390
	Network Components	400	245	245	245	245	245	1625
		555	**371**	**369**	**369**	**368**	**368**	**2400**

Table 5.8 – Spending profile for the on-site ICT investment costs (excl. network cabling) for the southern installation for the build-up of the array in 6 stages as described in the CTA-INFRA TDR [39].

5.5 Maintenance and operation

The maintenance and operation of ACTL on the two CTA sites comprises two main tasks: the maintenance and operation of hardware and that of the software. In both areas, a constant monitoring and expert help for keeping the system running and CTA data-taking at high efficiency are expected as part of regular CTA operations. In addition and during the lifetime of CTA, several upgrade phases with new instrumentation will need support for integration into normal data taking.

5.5.1 Expected needs and schedule

Information, computing and storage (ICT) infrastructure

Spares management (for e.g. array trigger, clock distribution, network, storage, computing) is an important part of a strategic service management. The spares management will cover the exchange of elements such as disks, but also the exchange of larger components such as cold swap units for specific sub-systems. In case of a data centre the goal is to ensure a stable operation at all times. To replace faulty hardware in time, a well-defined workflow is essential and the replacement hardware has to be available quickly. For this purpose a stock of all necessary hardware replacements has to be availabe at the CTA sites and to be checked for completeness regularly. Furthermore the local staff need to know fast and precisely, which hardware has to be replaced on which server or rack. For this purpose, a close collaboration between the spares management system and other services like monitoring is necessary. In addition, the local staff on site needs to be trained to do the necessary replacements. This maintenance will be done throughout the lifetime of CTA.

The spares management concerning the network includes copper cards, LWL cards and optical modules for the core switches (see Tab. 5.9). Furthermore at least two ACTL switches should be part of the spare. So, faulty hardware can be changed in a very short time by the local stuff. Independent of the spares there will be a next business day service contract over 5 years for all network components. This contract includes changing faulty hardware and fixing software bugs.

Hardware renewal allows to profit from the improvements in capacity, computing capability, energy consumption of newer generations of ICT hardware. A renewal cycle of 5 years for computing and storage sytems is expected. For any new hardware, a testing period within a testbed cluster is foreseen to assure a seamless integration into the on-site cluster and continuing support for all software tasks of ACTL and DATA. Active network components are foreseen to be renewed 10 years after the start of the operation.

System administration regular security updates of the operating system, maintaining the ICT services (DNS, time service), installation of new operating system releases, and regular checking of the ICT monitoring system are regular maintenance activities to be unterdaken by local staff.

Clock distribution and trigger-timestamping system

After the initial calibration, integration and on-site verification of the CDTS, its behavior is completely autonomous. Under normal circumstances the daily operation of the CDTS should not require manual configuration of or interaction with the system. Automated monitoring protocols are part of the CDTS software framework and will be used to help ensure high availability of the system. The monitoring will include data from the CDTS system alone, as well as data obtained from the camera servers (i.e. success of bulk data and time stamp merging) to verify the CDTS performance in (near) real-time. Note that the software algorithms used in the camera servers for this purpose can be provided by the ACTL-TRIG group.

The CDTS will also attempt simple error state recovery automatically, e.g. trying to reset a UCTS board that is apparently out of sync. When these automatic recovery procedures fail, local personnel will then review the available information and decide how to intervene. Slow control and monitoring displays providing a quick status overview of the CDTS will be provided to ACTL so that the local personnel have the necessary information and the CDTS software framework will provide manual recovery options that permit actions such as resetting the CDTS to a default state.

The routine maintenance of the CDTS system will be limited to monthly tests of the calibration of a small sub-set of the UCTS boards. CDTS system experts need only analyze slow control and monitoring data from the site if problems are flagged by the automatic monitoring software. A set of pre-calibrated spares for UCTS boards and other CDTS components (e.g. White Rabbit switches, optical fibres and SFP transceivers) will be made available on site and will be swapped in when necessary. From experience with current experiments like H.E.S.S., the number of spares for each component should be in the order of $\approx 10\,\%$ (at least one spare for any given device). Failing components can either be checked on site or sent back to the respective institute for repairs.

Software

Maintenance Updates, patches and feature upgrades are expected for all software products during the lifetime of CTA. A dedicated team of software experts needs to be available to implement and oversee these software changes, including all steps of validation and verification as well as integration tests on testbed systems prior to deployment on site. Software maintenance will follow the ISO/IEC14764(2006) specifications for software maintenance, which distributes the efforts in:

- corrective maintenance: fixing of bugs discovered by users of the software
- adaptive maintenance: adapt software to chagnes in the environment (e.g. changes in OS, third-party software)
- perfective maintenance: modification of software to improve quality attributes (performance, reliability)
- preventive maintenance: modification of software to detect and correct latent faults before they become effective

Upgrade support Software management activities comprise not only the operation of the running ACTL system on site (e.g. updates of operating system and security patches, updates and patches of ACTL software), but also the integration of new devices as well as new features of the ACTL system (e.g. improved data volume reduction schemes). Therefore, a well-defined software change process (e.g. installation of a software change control board, usage of up-to-date bug tracking systems, central code repositories, regular code reviews) is envisaged and will need a permanently available group of experts.

Operations support CTA is required to run with high availability and reliability. As in any experiment, failures of systems during operation cannot be fully avoided and compensated by the automatic recovery system during operations or by the local crew during operations or daytime. A group of experts will need to be permantently available to assist in the debugging of the system and problem solving.

Remote array control centre In addition, a remote array control centre with access to the engineering data, the logging and alarm information and a full test bed cluster is foreseen to support operation and development for the two CTA sites. The benefits are two-fold: new developments can be tested and validated before being deployed at the site and any problems during data-taking can be replayed and debugged in a well-defined environment without the need to remotely connect to the sites.

5.5.2 Life-time Costs

Costs for on-site maintenance and operation of information, computing and storage infrastructure

One main part for operations costs is the power consumption. In Tab. 5.9 the estimated power consumption of the data centre (excl. cooling and the UPS heat) is shown for the fully developed south array. Based on experienced values (from the DESY data centre) a power consumption average of 10W/TB and 16W/core can be supposed. With the required 3 PB and 1600 cores the storage and computing structure will consume 125 kW. The network power consumption is generated by all active network components in the data centre. Here the core switches as well as all smaller management switches and controllers have to be considered. The power consumption of the trigger and time-stamping system as well as auxiliary devices are rather uncertain, but contribute a relatively small fraction. Using estimates from single components, about 20 kW are estimated for everything which is not storage, computing or network. The power consumption of the camera server cannot be calculated by ACTL-ONSITE, because of the expected different computing designs. Here the camera groups are in responsibility. It is expected that the entire on-site ICT infrastructure will be used also during non-observing daytime (e.g. for on-site analysis, data reduction, simulations). Therefore a constant power consumption over the whole day is assumed.

Because of less hardware installed during the first years the warranty costs (given in Tab. 5.9) will later increase until the array is fully developed. After 5 years the network warranty costs will be 35k€/year. The spare parts of the computing and storage are about 3% of the investment costs per year (number based on the experience coming from running a standard computer centre like at DESY) and the warranty is included in the investment costs in that case. The network spares costs are fixed as long as the warranty / service contract is valid. It is planned to replace the entire on-site computing and storage system 5 years and the active network components 10 years after the start of the operation.

Concerning the man power it is planned to have 3 FTE / year on site during the construction (see WBS/PBS). After 5 years 1.5 FTE for maintenance, computing and network support and troubleshooting are expected.

			Yearly Costs €
WBS Item			
3.5	On-site ICT Infrastructure	Comments	
3.5.2	On-site ICT Infrastructure South		
	Spares		37.342 €
	Computing	3 % of total investment costs per year	10.942 €
	Storage	3 % of total investment costs per year	10.800 €
	Network Components		
		2 x 10 Gbit/s copper card per 5 years	4.000 €
		2 x optical fibre card per 5 years	8.000 €
		1 x SFP + 10 Gbit/s SM per year	1.600 €
	Array Control Room		2.000 €
	Service and Warranty costs		35.000 €
	Computing	included in investment costs	
	Storage	included in investment costs	
	Network Components		35.000 €
	Hardware Renewal		275.087 €
	Computing	replacement every 5 years	72.944 €
	Storage	replacement every 5 years	72.000 €
	Network Components	replacement every 10 years	130.143 €
	Power Consumption		1.042.000 €
	Storage	10 W / TB ➔3 PB * 10 W = 30 kW	
	ACTL computing	16 W / Core ➔800 Cores * 16 W = 13 kW	
	Data analysis computing	16 W / Core ➔800 Cores * 16 W = 13 kW	
	Network	30 kW	
	Trigger, auxiliary devices	20 kW	
	Camera server	Camera group responsibility: 800 Cores = 13 kW	
	Total	119 kW * 10 Cent	1.042.000 €
Total Costs (CTA South Array only)			1.389.429 €

Table 5.9 – Yearly expenditure for operation and maintenace of the on-site ICT infrastructure.

Costs for on-site maintenance and operation of software

It is currently estimated that the maintenance of the ACTL software products will require a about 10 FTE of on-site and remote staff from ACTL experts and software architects. Given the current diversity of hardware and software to be installed on-site and the associated needs for maintenance, the estimate is currently rather uncertain. The on-site staff will be trained by ACTL experts in dedicated workshops prior to the handover of the ACTL software products to CTAO.

5.6 Assumptions, Dependencies and Caveats

5.6.1 Assumptions

The ACTL planning is based on the following set of assumptions:

- A trained local operator crew is available for commissioning and routine operation.
- A computer centre with the needed infrastructure (cooling, uninterruptible power, etc.) is ready for the installation of ACTL hardware.
- A counting room with the needed infrastructure (desks etc.) and safety measures (e.g. interlock for GRB alerts) will be provided.
- The camera data rates that result from the camera readout after a local camera trigger are such that they can be transferred on (up to four) 10 Gb ethernet lines from a telescope to the data centre.
- All camera servers will be located in the data centre.

- Camera servers are responsibility of the telescope projects.
- ACTL will only assume responsibility for the maintenance of camera servers which run ACTL-approved operating systems and application software.
- DATA and ACTL will agree on one standard software for astronomical coordinate transformations that will be used both on-line and off-line.
- All relevant configuration parameters (e.g. camera high voltage settings, trigger settings, CCD camera exposures etc. etc.) are stored, modified and logged under ACTL control.
- Data from calibration procedures (e.g. CCD images, data from calibration runs) will be stored in databases under ACTL control.
- Calibration data (e.g. telescope bending models) will be stored centrally in databases under ACTL control.
- The development of data compression and sparsification routines is a responsibility of the teams providing the hardware (e.g. a Cherenkov or CCD camera). ACTL will provide the framework to execute the software.
- The development of quality criteria for science and calibration data is a responsibility of the teams providing the hardware (e.g. a Cherenkov or CCD camera). ACTL will provide a framework to store and apply the criteria.
- The development of error recovery procedures (e.g. what to try when a calibration unit cannot be used) is a responsibility of the device teams. ACTL will provide means to detect and correlate error messages and to execute recovery actions.
- Events' individual timestamps will be added before they leave the camera server.
- Events data will be stored per-camera.
- There will be one single in-memory format for events data coming from all camera projects.
- All high-throughput streams will rely on the zeroMQ library.
- There will be one single operating system used by all components.
- Regarding the data volume reduction, it is assumed that the full wavefront is kept only for a selected (important) pixels, whereas other pixels only keep basic information (integral charge, time of maximum etc.) In particular, it is assumed that the full wavefront is only stored for the 3% of the pixels and for the remaining 97% only 15 bytes are used to characterise the charge and signal shape.
- The capacity required of the White Rabbit timing network are set by the nominal data rates (average and peak) given in the camera TDRs.

5.6.2 Dependencies

Table 5.10 – List of dependencies for ACTL-ONSITE.

Input needed from other projects	**Important for**
Data rates and data flow (from telescope to data centre)	storage and backup
Computing model (which data will be stored, how will the data be stored, how does the analysis work, what happens with the output of the analysis, how large are single files for the analysis, ..)	storage, CPU, bandwidth (transfer rate)
Necessary plugs or interfaces at the telescopes	patch panel and switches
Which time zone will be used on site? Recommendation is UTC without daylight-saving time.	settings in software
What procedures exist for the hardware imports? Do vendors deliver to Africa / South America?	Important to make a decision concerning the hardware delivery from Europe to the site.

Table 5.11 – List of dependencies for ACTL-OPS.

Input needed from other projects	**Important to**
ACTL frameworks (ACS and OPC UA) are installed and keep updated	Run ACTL software
Control room hardware (screens, sound device etc) fits the requirements for the operator interface	Be able to run the operator interface
Remote access is provided for ACTL-OPS experts	Enable remote install/fix software, and allow in-call expert actions
Web server is provided to install the ACTL applications based on web-frameworks	provide remote access to non experts
Only instructed crew uses the operator software	Proper usage of the software
A CTA operations plan should be provided	Produce the adequate system architecture and performance tests

Table 5.12 – List of dependencies for ACTL-SLOW.

Input needed from other projects	**Important to**
ACTL frameworks (ACS and OPC UA) are installed and keep updated	Run ACTL software
Telescope software running in telescopes has been previously tested in the test cluster	Integrate *bridges* in a stable behaviour
The ICD with the telescopes and COM teams reflects exactly the behaviour of the hardware devices	Input to produce the *bridges*

Table 5.13 – List of dependencies for ACTL-DAQ.

Input needed from other projects	**Important to**
Individual telescopes produced data	Implement a common, complete data format
Time coincidence detection data format	Readout of the telescope and array triggers
RTA data format and protocol	Deliver events to be analysed
On-site repository architecture	Commit raw data
Raw data format	Write raw events data as expected by DATA

Table 5.14 – List of dependencies for ACTL-SCHED.

Input needed from other projects	Important to
OBS generates a well defined set of proposals	Generate a proper and correct Scheduler
ACTL frameworks (ACS) are installed and keep up-dated	Run ACTL software
The ICD between ACTL-OPS and ACTL-SCHED is running	ACTL-SCHED requires to know the current status of the system
The ICD between OBS and ACTL-SCHED is running	ACTL-SCHED requires to know the proposals to schedule

Table 5.15 – List of dependencies for ACTL-TRIG.

Input needed from other projects	Important for
Data format and protocols used by individual telescopes	Detection of Array Triggers. Implement a common, complete data format
Time stamp data format and protocols	Readout of the local triggers from telescope.
(sub)arrays configuration data format and protocols	Correct time-sorting of local triggers

5.7 Risks

The full registry of the risks of the ACTL project are contained in a spreadsheet table [40]. In this section we provide a sub-sample of that register, which are the 20 highest risks after a classification of the impact of the risk.

ACTL Risks									
General information		Risk identification			Costs	Inherent risk rating	Actions		
Risk ID	Risk event	Objective	Risk cause	Impact / Consequence	Costs	Risk rating	Actions	Deliverables	Required resources
1	Poor communication inside the project. Fragmented software developers and lots of interfaces.	Deliver a stable and maintainable ACTL software	manpower too distributed, lack of coordination, hierarchy and rules	Delays, extra costs for maintenance (Additional 2 FTE/y)	200 kE / y	HIGH	Create a strong team that will be the core of the ACTL software creations. Assign critical tasks for that team. Create clear ICDs and stick to the rules. Deploy a rich set of tooling to enforce software convergence	WBS	Manpower, tooling set, ACTL test cluster
2	Telescope teams produce their own "full" array control software setup	The ACTL effort should be focused in provide and array control software for all the telescopes in both arrays	Not clear rules or enough enforcement by the management	Divided efforts and unclear later integration. 2 FTE for each occurrence to integrate	200 kE / occurrence	HIGH	Define the rules at CTA management level and enforce them. Produce an early infrastructure of the central control system building blocks to allow telescope prototypes easy integration	WBS	Manpower
3	Political games or conflicts	Clear leadership for the ACTL manpower	Software developers are hired by institutes which may not share the objectives of the ACTL management team	Inefficient work, mapower wasted. 1 - 5 FTE	100 kE - 1 ME	HIGH	Define the clear rule that no task that is not within an ACTL WBS box is considered as an ACTL task	WBS	
4	The preferred data reduction architecture cannot be deployed due to limited computing resources within the camera servers	Reduce on- site network throughput	Limited computing power of a single server	More than one physical server must be used for the computations (load-balancing) Doubles the computing needs for camera servers	200 kE	HIGH	More than one physical server will be used for data reduction. The resulting higher network throughput might require to revise the network topology.		
5	Poor definition of telescope deployment plan.	Understand the priorities and deadlines for the ACTL project	Lack of information flow to ACTL	The ACTL deliverables are not well aligned with the real priorities, thus would delay the normal operation of the CTA array.		HIGH	1) Insist on gaining such knowle+L8dge 2) Make an educated guess for the plan meanwhile	ACTL schedule based on the best guess	
6	Poor support of agencies for software development/maintenance	Employ personal to create and maintain the ACTL software	No funding secured. Irregular funding flux to host institutions	A certain amount of software products cannot be delivered, delaying or downgrading the array operation		HIGH	1) Preforming a clear evaluation of manpower, report to CTA management 2) Produce quality documented software to allow the transfer of code to other teams	WBS, programming guidelines	Funding Approval (Date) Funding scenarios agreement
7	Requirement change	Provide the correct hardware and software	lack of strong management	Inefficient software development, unexpected aditional costs. (1 - 10 FTE?)	100 kE - 1 ME ?	HIGH	Produce a well document, ICDs, use cases and specifications, and after accepted only allow changes after strong justifications	ICDs, specifications, use cases	Manpower to produce the documents (~ 1FTE 1 yr)
8	The central control software is not ready when the first telescope units are commissioned	ACTL provides a control environment for the first integrated operations of the telescopes	Bad management of the project priorities/lack of manpower	The commissioning phase cannot be performed within the final ACTL environment, this requiring preliminary environments and producing overheads and delays. 2 FTE dedicated	200 kE	HIGH	1) make clear assessment of required manpower and define clear priorities fore the project. 2) be ready to provide temporary solutions that cause a minimum overhead. 3) Prioritize over other activities	Update of the WBS. Prepare scrip based operation	ACTL programmers core group: 3-5 FTE/yr
9	Underestimation of human resources, lack of expertise, or loss of experts for software development	Produce the ACTL software	Not enough founding or incorrect usage of existing resources	Low quality of software products, downgrade of functionality and delays		HIGH	Preforming a clear evaluation of required vs existing manpower, report to CTA management of the situation and request plans to solvethe possible lack of manpower.	WBS	0.1 FTE
10	A key project developer leaves the project	The projects should be prepared to avoid single point of failures in key projects	The knowledge has been concentrated in a single individual who has not share conveniently it	Delays in the project	100 kE	HIGH	Sub-task leaders should organize the project so this does not happen. Software documentation rules should be enforced		
11	OPC UA software implementation becomes challenging	Device teams software is implemented under OPC UA	Incorrect choice of OPC SDK or adequate manpower for the task	The creation of software is delayed and/or the quality of the software is poor. Coulr require up to 1 FTE	100 kE	HIGH	1) Suggest to use two specific SDKs for OPC UA, from Prosys and Unified Automation 2) The CTA ACTL team will support and spread the knowledge of OPC UA.	1) Example Code 2)LAPP Generic OPC UA server	Manpower to support, travel budget for bootcamps]
12	Software provided by the telescope/COM teams experiences problems in the integration in the central array control system	Device control is well integrated and with the minimum effort in ACTL	Device software creation was not well coordinated and tested	The behavior of the software piece is not what is expected, thus reveal unstable or cannot be integrated in the array control system. 0.2 FTE to support	20 kE	HIGH	1) Clear interface definition prior to software creation. 2) Extensive tests prior to deployment	ACTL-XST; ACTL-COM ICDS.	1) Manpower to create the ICDs. 2) Manpower for the test procedure
13	The available manpower does not allow for a delivery of the DAQ software in due time	The operations of early telescope projects cannot start	Not enough manpower available for the DAQ project	Data acquisition cannot be performed: delays and extra costs, 1 - 2 extra FTE	100 - 200 kE	HIGH	1) teams will use their prototyping format until the DAQ software is available 2) more manpower will be allocated to developing the basic DAQ modules	WBS	
14	Adding new items to an individual telescope data format breaks the entire pipeline	Transport the relevant data from the cameras through the DAQ pipeline	Input to create the data format was not enough / chosen format not flexible enough	Data taking cannot continue		HIGH	1) Make the best effort to create the adequate data format 2) Decide to use a versioned data format such as the protocol buffers.	ICD with camera teams	Manpower, 0.1 FTE 1 year
15	The operator GUI is very complicated for efficient usage	Provide an intuitive GUI which offers a compact view of the status of the array and a flexible procedures for input	Incorrect design	Would require extra effort to train the operators and potentially to employ more staff to run the array. Can reduce the performance of the array. Revise/redesign the interface using 3 FTE	300 kE	HIGH	Brainstorming for general idea of the GUI with the help of an external expert.	Specifications/use cases of the central GUI	Resources for hiring the help of the external expert (few days).
16	Poor financial support for ACTL onsite infrastructure by agencies	Deploy and maintain the hardware infrastructure	No funding secured / wrong estimate of required funding	Delays on operation, downgrade the array capabilities		MEDIUM	1) Clear evaluation of required maintenance costs, including run-time costs. 2) Produce documentation and prescriptions for maintenance for easy a knowledge transfer	WBS, documentation	Funding Approval (Date) Funding scenarios agreement
17	Incorrect system requirements	Provide the correct hardware and software	Bad evaluation of the requirements	Inefficient software development, unexpected additional costs (1 - 10 FTE?)	100 kE - 1 ME ?	MED	Have a critical revision of the specification, ICDs and use cases within the work package	N/A	
18	Poor definition of raw data volume and data to be archived	Provide the correct hardware setup to transfer, process and store the data	Indefinition and poor communication with the camera projects.	Increase the costs or delay the deployment of the network infrastructure and onsite computing. Assume a 20% additional cost on infrastucture ans storage	900 kE	MEDIUM	1) Insist on gaining such knowledge, and develop a modular software. 2) Prepare a hardware setup with capacity well above these estimate. Use a distributed computing model.	Data rates estimates	0.1 FTE in 1 yr.
19	Insufficient quality of deployed software	Deliver high quality software products	Lack of manpower and or tests of the software	Loss of observation time. Delays. Extra costs dedicating 1 to 5 FTE depending on the affected product	100 kE - 500 kE	MEDIUM	Extensive code reviews and unit, integration and large scale performance tests in the test cluster.	ACTL software policy rules	ACTL software board --> 0.5 FTE
20	Development of wrong software functions	Employ the coding effort in the required functionality	Wrong understanding of specifications / incorrect specifications	Waste of manpower: delays and extra costs. 1 -3 FTE	100 - 300 kE	MEDIUM	Critical revision of specifications. Close monitoring of subtasks		0.1 FTE / y.

Table 5.16 – Abbreviated version of the ACTL risk register. Only the 20 highest risks of the ACTL project are displayed.

6 Lessons incorporated

6.1 Lessons learnt from current astronomical installations

The experience with the control software used with the H.E.S.S., MAGIC and VERITAS telescopes proved very valuable in the phase when the software tools for building the ACTL software were selected. In general, the software developed for these observatories was tailored to the control of an installation with few telescopes, thus the direct re-use of such software was not considered. For example, the approach of the operator GUIs in these installations was a scaling with the number of telescopes, thus not practical. However, the experience from ACTL members who worked in such installations proved very valuable for the CTA ACTL consideration. In addition, the lessons from the experience of ALMA and ICECUBE have been taken into account. The following list provides the lessons incorporated into the ACTL approach:

- **The usage of a control framework enhances the control software:** By using a software framework, the software developers have a common starting point and API to produce software, and this is specifically important in a project like CTA where many relatively small institutes, geographically far away are expected to contribute in the software creation. The use of standard elements and protocols increases the stability of the product, in particular when the scale of the project. For that reason, the ACTL software has adopted the usage of the ACS as described in Sec. 3.2.2.

- **Device firmware and low-level software can cause major instabilities in the control system:** When the software to control low-level functionality is developed in “free-style” and in particular non-standard means of communication are used (like custom sockets), unstable behaviour is often observed. Such behaviour was sometimes difficult to evaluate and fix, and required substantial attention of the central control experts. The usage of a standard software bus (OPC UA) and a corresponding DevIO library together with well defined ICDs and advanced integration tests will greatly reduce such problems. To make sure that the software is stable enough, the ACTL team proposes a set of procedures prior of the deployment of any software (see Sec. 4.3)

- **Instrument monitoring data volume can become very large and not manageable:** The experience from ALMA, which uses about 170000 monitoring points was that the storage of these data in a classical SQL database (Oracle) is not viable, despite of the investing in very powerful server machines, clever storage schemes, and usage of specific knowledge from external consultants. The result is that even if at the end the data can be recorded, the performance for querying them is very bad. To solve such problems, ACTL aims for the use of NoSQL databases, in particular MongoDB. In order to identify that these NoSQL solutions are correct, detailed performance tests are on the way (see Sec. 3.2.2).

- **The classical “workstation” operator GUIs are sometimes unpractical:** The experience in actual installations is that when some expert modes or problems fixing happen, the shift crew needs to go near the affected telescope and also operate the telescope from the control room. This requires that one crew member communicates with the other via walkie-talkies or similar, which in general is cumbersome. For ACTL, we think that the possibility of having web browser based GUIs opens a huge potential to ease these operations, allowing the usage of portable devices like *tablets* to be carried near the telescopes (see Sec. 3.2.2 for further details).

6.2 Lessons learnt from telescope prototype projects

Besides gaining experience for producing software with the ACS and OPC UA frameworks, during the creation of software for the prototypes some additional lessons were leart that were then incorporated in the ACTL approach:

- **The integration of OPC UA servers can be optimized with early ICD and simulation versions:** The programming of ACS bridges can start as soon as ICDs are created and early tests can be conducted by using simulation versions of OPC UA servers. This allows also to exercise integration tests from early on.
- **Code reviews enhance the quality of the software:** By using specialized tools like *trac* and organizing code reviews, potential code problems can be spotted, as well as best practices shared. Code reviews are also a good mean to share the knowledge. The final software product is then much more robust. ACTL will only allow the deployment of ACTL software that has been reviewed.
- **Ticket systems enhance the process of reporting and fixing code bugs and missing features:** The experience in the prototypes is very positive towards the usage of such tools.

6.3 Lessons learnt from DAQ prototype

The specific prototyping of the DAQ system led to some more lessons learnt:

- **Large software projects are difficult to debug and maintain:** Software components will be small, performing basic tasks. The artificial intelligence will be located in a dedicated module.
- **Low level data cannot be modified without impacting the data quality:** There will be one single data format to minimize formats conversion.
- **Third party software lifespan is difficult to evaluate:** Dependance to third party software will be minimized. Only open-source will be used so that the maintenance can be taken over by CTA if needed.
- **TCP streams are limited to 10 Gb/s if the latency is small enough:** The communication layer will open more than one stream if needed.
- **The throughput limiting factor is the storage system:** As much calculations as possible will be done in real-time, before the first write to the on-site archive.
- **Compression performances can be greatly improved if tailored to the input data:** Compression schemes will be integrated in the low-level data access layer, thus enabling to introduce more efficient schemes at any time.
- **New features can introduce side effects in complex software:** New commits will be validated by running a set of unit tests.
- **What can fail will fail:** Error handling will be improved during and after commissioning as long as the available manpower permits it.
- **The on-disk data format should slightly differ from the in-memory format:** Thanks to multi-threading, even compression can be done on-the fly. Thus compressed FITS files written from protobuffers structures via reflection is a suitable candidate for the on-disk format.
- **Pre-conditioning data can alter the compression performances. Compression ratios can be smaller if the data is pre-packed rather than given raw to the compression algorithm:** The data will be given raw to the compression scheme. Suitable pre-processing will be integrated to the compression scheme itself.

6.4 Lessons learnt from the SWAT prototype

Being a near real-time application, the SWAT system led to some more lessons learned:

- **Near real-time processing and the bottlenecks** The major challenge in implementing an array trigger in software is the raw number of network packets per second processed in near real-time. Proper design should minimize number of packets exchanged over the network, by i.e. aggregating the data in as large packets as possible. This is important specifically for SWAT, but also for any other module (like DAQ).
- **Near real-time processing requires extra time margins** High volume data traffic over the network inherently induces delays and jitter in packet delivery. Near real-time software should wait sufficiently long before attempting to process a packet or marking it as missing.

6.5 Lessons learnt for the scheduler

6.5.1 Lessons learned from current astronomical installations

The operational design for the CTA infrastructure was not available at the time the specification of requirements for the CTA Central Scheduler started. Therefore, the design and experience on similar facilities were taken as a reference. The definition of scientific proposals was based on MAGIC project and helped building up the criteria for the creation and prioritization of scheduling blocks, and defining the criteria used by the scheduler to check for the proposal completion. MAGIC operational design might be significantly different from the one for CTA, however it was mature enough to be used as a reference in the scheduler prototype design and its later validation. Other astronomical installations where considered, although they could not be used for prototype validation due to different reasons:

- Autoscheduler of HESS: the scheduling application used in HESS for a dynamic selection of targets (called Autoscheduler) is not appropriate for the CTA scheduler prototype validation. Data model (for data ingestion and output evaluation) and observation strategy are difficult to disentangle and the comparison of performance with other scheduling software requires a significant effort. Therefore, HESS data could not be used for the scheduler prototype validation. However, the HESS long-term planner might be used in the near future for checking long-term optimization. Differences between the operational design applied at MAGIC and HESS helped in a more complete compilation of requirements for the CTA scheduler.
- Scheduling at ESO: the gap between the operational approaches implemented in the CTA scheduler prototype and in the ESO facilities makes it very difficult to use the existing solutions (i.e., TATOO) for comparison. However, the scheme applied at system operation level was very useful to draw the current scheduler architecture design: this is based on long-term and short-term scheduling modules working in a coordinated manner and providing service to the scientific team and the operator, respectively.

6.5.2 Lessons learnt from CTA scheduling simulator prototype

The CTA scheduling simulator was very useful to derive conclusions about the operational performance of the scheduler prototype. In particular, a scientific validation was carried out by using a set of physical parameters and key efficiency metrics (i.e., ratio of completed proposals, telescope time used for data collection, etc.). The tool was made available to different groups involved in the KSP definition and in ACTL. An important feedback was received in terms of the available functionalities, and also new functionalities were suggested.

A Full Product Breakdown Structure

B Full Work Breakdown Structure

C Computing needs estimates

In this section it is assumed that when mentioning computing needs it is referred to state of the art, standard computing servers as available at end of 2014.

C.0.3 PBS 3.1: ACTL-SLOW

ACTL-SLOW needs 124 computing cores for the southern array, and 60 for the northern one. The current estimates rely on scaling the needs of small scale demonstrator environments deployed in both the MST (*case 1*) and ASTRI (*case 2*) prototypes. Each environment represents a different approach on the distribution of the responsibility on the workload for the instrument operation software: in *case 1* most of the computing is performing in the central facilities and thus to be taken into account by ACTL, whereas for *case 2* most of the computing is performed on embedded systems onboard the telescopes and thus not to be considered by ACTL. The overall system at the Southern array will consist of a mixture of telescopes currently implementing one or the other approach, whereas in the Northern array of telescopes implementing just *case 1*.

C.0.4 PBS 3.2: ACTL-OPS

For software products of ACTL-OPS, the computing needs comprise of:

- **Central control services** (Coordinator, sequencer, Resource manager, alarm system etc.) 24 Cores for South and North array each.
- **On-site monitoring engineering and logging data repository:** 40 cores (in 5 nodes) and 0.3 PB of data storage for the southern, and 24 cores (in 3 nodes) and 0.1 PB storage for the northern installation. For the monitoring logging (guaranteeing storage of one year of “raw” monitoring data, including CCD images), configuration and alarm storage.[1]
- **On-site Cherenkov data repository** 60 cores which allow a total of 3.5 PB of data storage for the South array, and 30 cores for the North.
- **Configuration database** 2 servers (1 for operation and another for backup) for each array.
- **Operator room** 14 Screens and 6 standard computers for each array. These are distributed as: 2 large screen, for passive information (+ 1 spare), 1 operator workstation with 3 screens (+ 1 spare set), 4 workstations with its screens for additional operator or staff usage. Room sound system for alarms and wireless access.

C.0.5 PBS 3.3: ACTL-DAQ

The baseline chosen for the DAQ system generates a throughput of 500 Gbps while the required number of compute cores is approximately 500 to be able to transfer, zero-suppress and compress the raw events

[1]For the noSQL solutions we are proposing a minimum set of 3 nodes for instrument monitoring, thereby ensuring quorum reads+writes and both strong consistency and availability if a single node goes down.

data in real-time. The throughput to persistent storage would thus be 34 Gbps. More details are available in the architecture document [16]. The high throughput is not an issue for the DAQ system as long as the network is capable enough. Indeed, the foreseen modular architecture allows to use as many physical nodes as required to operate in parallel. Load balancing will be possible right after the camera servers, ensuring that further computation can be performed. Writing events data to the on-site repository is also challenging. It is foreseen that events will be written per-telescope, thus one separate storage system could be used for each telescope. In case the data rate from a single telescope is too large to allow for real-time data compression, more than one data writer will be used. This allows to use also more than one storage system per telescope. Storage systems will be used at any given time for a single taks: either write or read to/from it. Read and write tasks will not be performed simultaneously to avoid that the performances of the storage system are degraded. The current estimates for the needed computing resources are based on the current DAQ prototype system. A need of 400 cores for the readout of the southern array of telescopes is estimated.

C.0.6 PBS 3.4: ACTL-TRIG

The computing needs for the software array trigger (PBS 3.4.1) is estimated from the current prototype to 16 cores in total for the southern installation. An additional machine is required for the CDTS server of the clock distribution and trigger-timestamping system (PBS 3.4.2). These estimates do not include cold swaps.

C.0.7 PBS 3.5: ACTL-ONSITE

The computing needs for the IT services (login, user authentication, workgroup servers, central services etc.) is estimated from current experiences of large-scale data centres and results in 96 cores, or 6 machines respectively, for the southern installation.

C.0.8 PBS 3.6: ACTL-SCHED

The computing needs for ACTL-SCHED is estimated in 12 cores at each observatory site and running on the ACTL infrastructure. An additional server with 12 cores is required at the CTA central premises to compute the long-term plan for a full season and running as a support tool for the observatory operations units.

C.0.9 DATA management: level-A and level-B analysis

From scaling of level-A and level-B analysis software at previous experiments such as H.E.S.S. and MAGIC as well as experience with current analysis software prototypes, a total computing need of 800 cores was estimated [41].

D Remote Control

A remote control of the CTA arrays would be technically possible with the ACTL system architecture. In this appendix, the guiding principles and assumptions for such operations are described:

1. A remote control of a CTA array will only be provided and allowed once sufficient experience with local, almost automatic, routine operation with a CTA installation has been gained.
2. The primary objective of remote operation is *not* science data taking but situations where maintenance, debugging or upgrades (of, for example, observation modes) should be performed without sending expert personnel to a CTA site.
3. Remote operation requires coordination, trained personnel and a stable network connection with good performance. The remote array control centre is therefore the primary site to conduct remote observations while operation from an expert at a desktop PC is considered as an exceptional case.
4. At all times, remote observations are at the discretion of the person/body in CTA overseeing science operations.

The following organizational and technical principles should be applied:

1. Only experienced operators are allowed, even if the remote control is done via the remote array control centre. Remote operators must be competent to create schedules, initialise and execute observations without support. They must be capable of recognising problems and exceptional situations (e.g. ToO, alerts from other observatories) and know how to handle them.
2. Remote control will only be granted to experts who have observed in person at one of the CTA observatories or at the remote array control centre. Important changes to the array configuration (e.g. adding telescopes) or the ACTL software system will invalidate permits.
3. If remote control has been approved, the on-duty local operators must be contacted and confirm that a remote control will be happening.
4. Remote control can only start once the local operators have performed safety checks and actively handed over control.
5. Remote control will take advantage of a GUI based on web browser technologies. Remote operation is performed either by (i) the remote array control centre which mimics the operator room on a CTA site, and is configurable to be used for South or North observatories, or by (ii) a dedicated GUI with a tailored interface well suited for the operation from a typical desktop PC setup with a single computer screen.
6. For safety reasons only one GUI instance can have the control of a CTA array at a time.
7. At any time, the local operators at a CTA site can override and take control from either the remote array control centre or a desktop PC again if required (e.g. in case of an incident). The remote user will be immediately notified that he/she has lost the control.
8. During remote operation, strict monitoring (e.g. of weather conditions) will be in place and will override the remote access in case any abnormalities are detected.

In case that remote observations are to be conducted by an expert at a desktop PC the following conditions must be met:

1. The experts need to demonstrate (before remote control is approved) that their computer and internet connection have sufficient power, responsiveness and connectivity to adequately run the required software.
2. Experts must be able to initiate or receive phone calls to/from the CTA array and the remote array control centre.
3. At any time, the remote array control centre can override and take away the remote control from a desktop PC if the situation requires so. The expert will be notified.
4. In order to eliminate the risk that a virus or other malware may reach the CTA site's network, the remote connection must be made via a Virtual Private Network configured to lock out access to the remote user's LAN.

E Risk Register

F List of Authors

Role	Name	Institute
Editor	M. Füßling	DESY
	I. Oya	DESY
	U. Schwanke	HU Berlin
	P. Wegner	DESY
Author	J. Colomé	IEEC-CSIC
	T. Le Flour	LAPP Annecy
	R. Lindemann	DESY
	E. Lyard	UNIGE-ISDC
	A. Weinstein	U Iowa State
	A. Balzer	University of Amsterdam
	D. Berge	University of Amsterdam
	J. Borkowski	CAMK Torun
	J. Campreciós	IEEC-CSIC
	S. Colognes	APC
	D. Melkumyan	DESY
	M. Punch	APC
	C. Tanci	INAF
	G. Tosti	University of Perugia
	J. Schwarz	INAF - OAB Milano
	R. Wischnewski	DESY

Table F.1 – List of editors and contributing authors.

References

[1] *ACS as core software framework for ACTL*. ACTL-PM/140224a

[2] Wootten A. & Thompson A.R. (2009). *The Atacama Large Millimeter/Submillimeter Array*. IEEE Proceedings, **97**, 1463

[3] *OPC UA Use for CTA*. ACTL-PM/140224

[4] *Device integration into CTA-ACTL*. ACTL-PM/140115

[5] *OPC UA Guidelines*. ACTL-PM/140401

[6] Bernloehr K. *Mc studies based on prod2 simulations*. private communication

[7] Bernloehr K. *et al.*. *Report of the trigger sub-task group*. CTA Internal note, 5/1172009

[8] *Data management technical design report*. DATA-TDR/140513

[9] Tosti G. & the ASTRI Collaboration. *The actl aspects of the astri software environment*. https://www.cta-observatory.org/indico/contributionDisplay.py?contribId=118&confId=516

[10] Chiozzi G., Jeram B., Sommer H. *et al.* (2004). *The ALMA common software: a developer-friendly CORBA-based framework*. In H. Lewis & G. Raffi (editors), *Advanced Software, Control, and Communication Systems for Astronomy*, volume 5496 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pp. 205–218

[11] *Acs community*. https://github.com/ACS-Community

[12] *ACTL Architectural Design document*. In preparation

[13] *ACTL requirements*. MAN-PO/121201

[14] *ACTL Preliminary Design Verification Document*. ACTL-PM/130117

[15] *ACTL Use Cases*. ACTL-OPS/140527

[16] *CTA Data Acquisition Architectural Design Document*. ACTL-DAQ/140508

[17] *Data Volume Reduction in CTA*. MAN-PO/130828

[18] Serrano J., Alvarez P., Cattin M. *et al.* (2009). *Icalepcs: The white rabbit project*. The 12th ICALEPCS, Kobe, Japan, pp. 1–3

[19] *White rabbit*. http://www.ohwr.org/projects/white-rabbit. Accessed: 2015-02-28

[20] J. S., Lipinski M. *et al.* (2013). *White rabbit status and prospects*. In *14th International Conference on Accelerator & Large Experimental Physics Control Systems, San Francisco, CA, USA*

[21] Moreira P., Serrano J., Wlostowski T. *et al.* (2009). *White rabbit: Sub-nanosecond timing distribution over ethernet*. In *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, pp. 1–5

[22] Włostowski T. (2011). *Precise time and frequency transfer in a white rabbit network*. Warsaw University of Technology

[23] *White rabbit wiki*. http://www.ohwr.org/projects/white-rabbit/wiki. Accessed: 2015-02-28

[24] Brueckner M., Wischnewski R., Berezhnev S. *et al.* (2013). *Results from the whiterabbit sub-nsec time synchronization setup at hiscore-tunka*. Proceedings of the 33rd ICRC, Rio de Janeiro, paper, **1158**

[25] Brueckner M. & Wischnewski R. (2013). *A white rabbit setup for sub-nsec synchronization, timestamping and time calibration in large scale astroparticle physics experiments*. Proceedings of the 33rd ICRC, Rio de Janeiro, paper, **1146**

[26] —. *First results from the hiscore/siberia wr-setup and plans for cta*. eventh White Rabbit Workshop, Madrid/Spain, 2012, http://www.ohwr.org/projects/white-rabbit/wiki/Nov2012Meeting

[27] Borkowski J. *Update on software central trigger*. https://www.cta-observatory.org/indico/getFile.py/access?contribId=161&sessionId=19&resId=0&materialId=slides&confId=115

[28] Colome J., Colomer P., Guàrdia J. *et al.* (2012). *Research on schedulers for astronomical observatories*. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 8448 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, p. 1

[29] *Interface Management Plan*. SYS-PLANS/140721

[30] *Seven solutions*. http://www.sevensols.com/en/products/spec.html

[31] *Creotec*. http://www.creotech.pl/en/

[32] Colomé J., Colomer P., Campreciós J. *et al.* (2014). *Artificial intelligence for the CTA Observatory scheduler*. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 9149 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, p. 0

[33] Dubus G., Contreras J.L., Funk S. *et al.* (2013). *Surveys with the Cherenkov Telescope Array*. Astroparticle Physics, **43**, 317

[34] *CTA Software Policy*

[35] Panzer-Steindel B. *Data integrity*. 2007, http://indico.cern.ch/event/13797/session/0/material/paper/1?contribId=3

[36] Seaman R., Pence W. & Rots A. (2012). *FITS Checksum Proposal*. ArXiv e-prints

[37] for Systems C. & Engineering S. *Cocomo ii*. http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html

[38] Kemball A.J. & Cornwell T.J. (2004). *A Simple Model Of Software Costs For The Square Kilometre Array*. Experimental Astronomy, **17**, 317

[39] *Infrastructure technical design report*. INFRA-TDR/140502

[40] *Actl risk register*. ACTL-PM/150228

[41] Bulgarelli A. *Data management on-site computing needs*. private communication

Glossary

ACL	Access Control List
ACS	Alma Common Software
ACTL	Array Control and Data Acquisition
CDB	Configuration DataBase
CDTS	Clock Distribution and Trigger Timestamping
CORBA	Common Object Request Broker Architecture
CTA	Cherenkov Telescope Array
CTAO	Cherenkov Telescope Array Observatory
DAQ	Data Acquisition
ESO	European Southern Observatory
FRAM	(Ph)Fotometric Atmospheric Robotic Monitor
IACT	Imaging Atmospheric Cherenkov Telescope
ICT	Information and Communications Technology
LIDAR	Light Detection And Ranging
LST	Large Size Telescope
LTP	Long-Term Planner
MB, GB, TB	Mega/Giga/Tera-Byte
Mb, Gb, Tb	Mega/Giga/Tera-Bit
MC	Monte-Carlo
MST	Medium Size Telescope
OPC-UA	OLE for Process Control - Unified Architecture
OSDC	On-Site Data Centre
PBS	Product Breakdown Structure
PCB	Printed Circuit Board
PLC	Programmable Logic Controller
PPS	Pulse per Second
RAMS	Reliability Availability Maintainability Safety
RTA	Real-Time Analysis (level-A analysis)
SCT	Schwarzschild-Couder telescope
SCTA	Scheduler for CTA
SDK	Software Development Kit
SST	Small Size Telescope

STP	Short-Term Planner
SV	Science Verification
SVO	Science Verification Observation
SWAT	Software Array Trigger
TDR	Technical Design Report
TMCDB	Telescope Monitoring and Configuration Database
ToO	Target of Opportunity
UPS	Uninterruptible Power Supply
VHE	Very High Energy
WBS	Work Breakdown Structure
WP	Work Package