17th Int. Conf. on Acc. and Large Exp. Physics Control Systems ICALEPCS2019, New York, NY, USA JACoW Publishing
ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL006

THE MINISCULE ELT CONTROL SOFTWARE: DESIGN, ARCHITECTURE AND HW INTEGRATION

C. Diaz Cano, N. Kornweibel, R.Abuter, J.Sagatowski, H.Tischer, T.R.Grudzien, European Organization for Astronomical Research in the Southern Hemisphere (ESO), Garching bei Muenchen, Germany

Abstract

This paper presents the development of the MELT (Mini ELT) Control System, to be used for testing and validating key functionalities of the Extremely Large Telescope (ELT) during AIV/commissioning and operation phase. MELT is an optical test bench with a turbulence generator, whose main objective is to deploy and validate the Central Control System (CCS) and the Wavefront control strategies. The subsystems under control are: a segmented primary mirror, a secondary mirror on a hexapod, an adaptive fourth mirror, a fast tip/tilt mirror, phasing sensor, a light source, a Wavefront sensor, a IR camera, together with their control interfaces that emulate the ELT conditions. The CCS integration layer, the Core Integration Infrastructure (CII), will be deployed to MELT for their verification and testing strategy, producing feedback to their requirements and design.

This paper describes the Control SW distributed architecture, communication patterns, user interfaces and SW infrastructure. The control algorithms are being developed separately and will be integrated into the control loop via MATLAB script API.

INTRODUCTION

MELT is a table-top emulator of the ELT (see Fig. 1), the European Extremely Large Telescope, the next generation Telescope developed by ESO [1]. It will be used for testing and validating key functionalities of the ELT, during the periods of system verification, wavefront control commissioning, through the handover to science, up to regular diagnostic, monitoring, or validation tasks during operations.

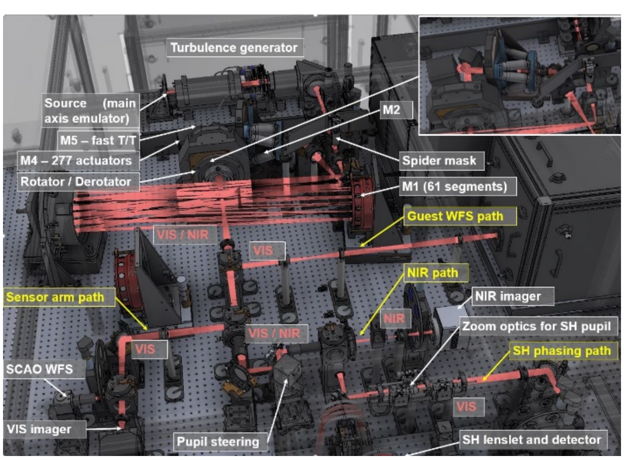

Figure 1: MELT optical test bench layout.

Another expected outcome of MELT would be to produce and validate requirements for the phasing and diagnostic station (PDS) of the ELT.

The MELT Control System (CS) Architecture follows the principles of the ELT Control Software and its Common Development Standards. Basically, the system is divided into hierarchical layers, i.e. into individual control systems associated with Telescope subsystems, collectively termed Local Control Systems, and the system that integrates these, termed the Central Control System. There are several products that have already been integrated within the bench: The network infrastructure (physical and data link layer interfaces); the messaging protocols through Core Integration Infrastructure (CII) middleware abstraction layer (MAL); the Instrument Control Framework (IFW); and the ELT Development Environment. The overall Software counts more than 550 files and 65K LOC, split in different programming languages, e.g.: C++/C (35K), Java (27K) and Python (11K).

SYSTEM DESCRIPTION

General Layout

MELT has been used as a precursor to the definition of user requirements, functional analysis, and define the most relevant functions. The CS block diagram (Fig. 2) describe the components functions throughout the optical path.

- Source: Laser driven incoherent white light in the wavelength of 500-1700nm, though a 25um multimode fiber.
- M1 active segmented mirror: consisting of 61 segments, each driven by 3 piezos to control piston, tip, and tilt with a free mechanical stroke of 15 um for wavefront control.
- M2 hexapod: hexapod is a compact 6DOF parallel kinematics system for the positioning and adjustment of precision elements with a resolution of 50 nm
- M4 Deformable mirror: ALPAO 277 actuator deformable mirror with a clear aperture of 24.5 mm, based on electro-magnetic actuators.
- Sensor arm: Fast tip/tilt (M5) and VIS imager, SCAO SH WFS 256x256 pixel with 207 um lenslets, 16 x 16 subapertures on a 3.3 x 3.3 mm pupil.
- IR Path: Before entering the IR path, the beam passes by the pupil stabilization tip/tilt mirror, with a fast full

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL006

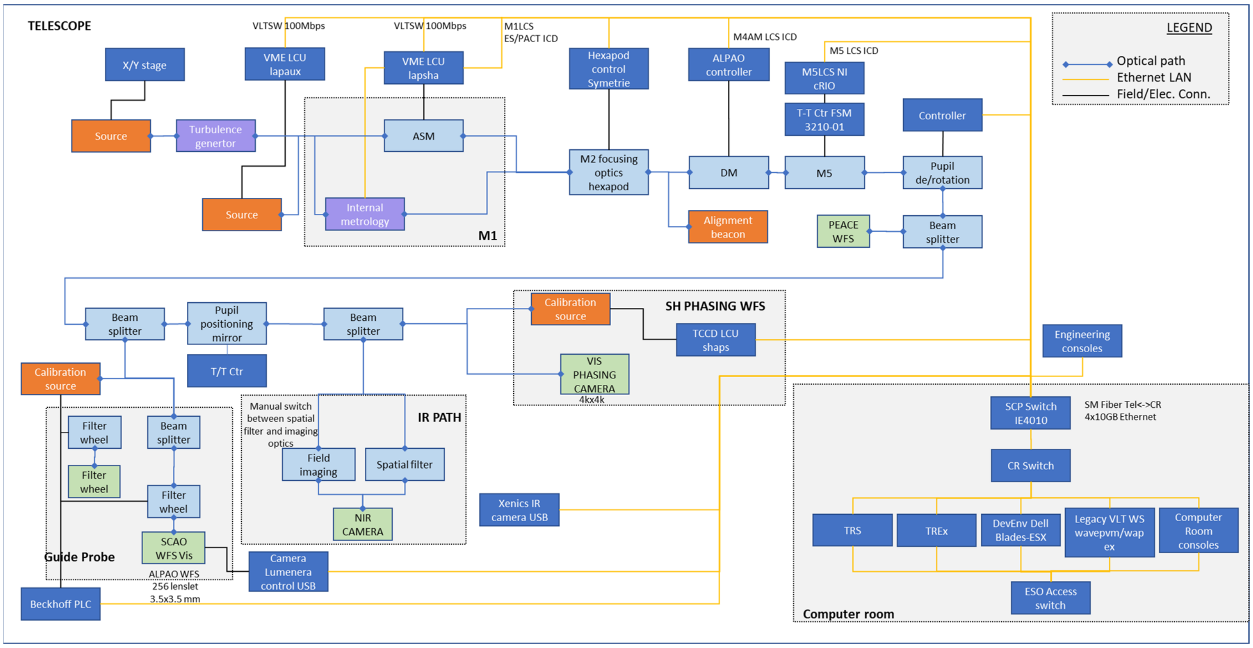

Figure 2: MELT CS block diagram.

frame readout performed by a 240x320 pixel IR camera.

- SH Phasing: Reused WFS (called SHAPS) with additional optics to adjust to the lenslet array. Additionally an automated calibration source and filter wheel is part of this path. Finally, a TCCD from the VLT program, with 512 x 512 pixels The entire M1 will be visible on the detector.
- Motors and Power control: Interface to two Beckhoff PLC, mostly restricted to moving motors of translation stages or filter wheels.

CONTROL SYSTEM IMPLEMENTATION

Control System Goals

Apart from the validation of the control algorithms, MELT aims to facilitate the validation of core SW products and technologies baselined for the ELT Control System. In order to do so, MELT will

- provide Ethernet based interfaces, between the CCS and the subsystem control systems, in-line with ELT (OPC/UA[2]), MUDPI, DDS[3], ZMQ[4] and protocol buffers[5]).
- Use the ELT Software development environment, Network infrastructure architecture and Time Reference System.
- Include Core Integration Infrastructure SW products as they become released, i.e., Middleware abstraction layer, Configuration, Online database, Telemetry and Alarm system.
- Include Instrument Framework (IFW): developed by ESO and intended as toolkit to help instrument developers to implement their control systems. It includes a set of PLC standard libraries controlling common devices (motors, lamps, shutters, sensors, ADCs).
- Enable closed loop and distributed control across subsystems (e.g. between wave front sensors and mirror control systems).
- MELT does not include any of the aspects of the telescope safety system.

Future Goals

- Integrate TREx: sub-assembly of the ELT Control System, that manages the communication infrastructure between the control equipment and the distributed real-time computers (at instrument side)
- Integrate new CII products, e.g., Online Database, Configuration, Alarm system and Telemetry service

Software Architecture

MELT control Software comprises multiple applications that run the required logic to command and measure the different devices. Due to the diverse nature of these devices, a wide set of programming languages and computer architectures are used.

The Software stack is designed so that they all use a middleware abstraction layer (MAL, part of CII) that enables the exchange of commands/measurements via ZPB, DDS or OPC/UA for three programming languages: Java, C++ and Python. Figure 3 shows the layered stack and main components.

PLC controlled devices share the so called Instrument Framework to be accessed via OPC data Access and RPC paradigms. IFW standardize the PLC libraries used for common devices, such as lamps, motors or timers. Additionally, it also offers a nice GUI where all devices are shown and operated.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL006

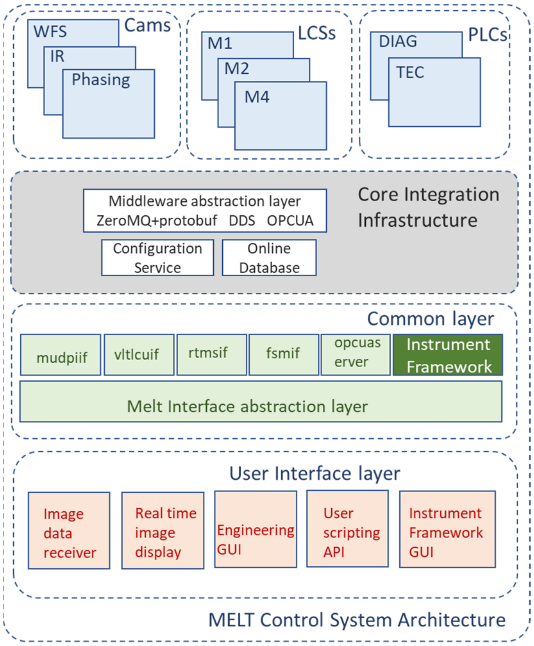

Figure 3: MELT control system SW stack.

Multiple Threaded Real Time Python Image Viewer

A Real time display viewer has been developed as a Python tool to view the images provided by the three cameras (SHAPS, Xenics IR, and Lumenera WFS). The program is multithreaded, splitting the processing of the:

- Reception of MUDPI/RTMS packages
- Unpacking of packages and image composition
- Display of images: OpenCV[6] was used to finally achieve a frame rate >30Fps

WFS and IR Camera Control

Three states (Idle, Configuring, Acquiring) handle the way the camera is controlled.

- Idle: Initial state. In the entry function it initializes the camera, and dumps its properties. Then waits until an event is dispatched. It reacts to the events.
- Configuring: Used to set any of the available properties of the Xenics camera
- Acquiring: It acquires N images from the camera.

In order to do an end-to-end control, two modules are required (Fig. 4).

Matlab user interface: subscribes to the images being published through MUDPI, and waits for N images to be received. After the N images are received, they are stored in an array. For receiving the housekeeping and sending commands, meltccs module is used. Meltccs: Initially waits until it receives TM from the camera. Upon TM reception commands can then be sent.

Camera LCS: Initially program creates three threads:

a. State Machine and camera control: enters into idle state and searches for a camera attached. Once the camera is found, its properties are dumped.

b. Command subscription: Waits until a command is received. Acquisition command triggers the camera thermal control (only for IR camera) during acquisition.

c. Telemetry publishing: As soon as the camera is initialized, the program sends telemetry (1Hz)

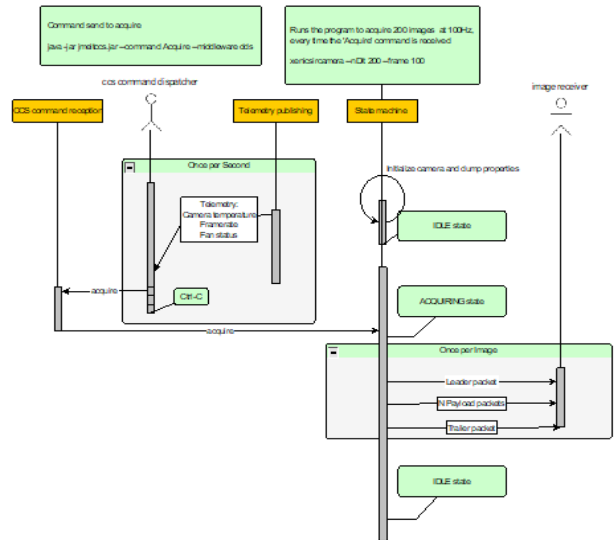

Figure 4: Sequence diagram for camera control.

MELT Graphical User Interface

An engineering graphical user interface (Fig. 5) has been developed to help the operator maintain or improve the system. It is based on QT [7], using the tab widget, and separates into several threads the display, the publishing of commands and the subscription to the housekeeping measurements.

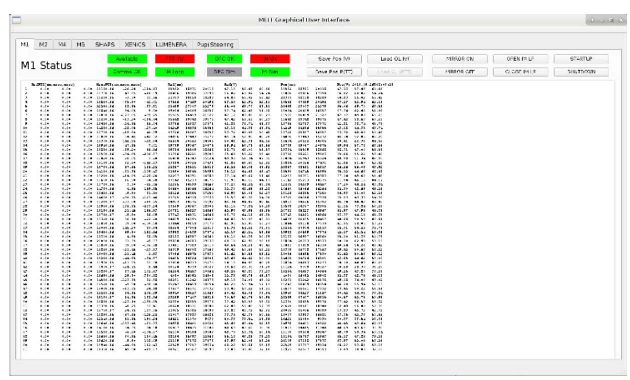

Figure 5: MELT graphical user interface.

It provides the operator with mechanisms to send low level commands, and sets of setpoints, when applicable.

Matlab User Object Oriented API

The Matlab interface permits interaction with the control system once it is fully available. The interface provides some functions to initialize certain subsystems. The operator GUI and user scripts are used to bring the desired subsystems of MELT to a state where they are available, after which the Matlab interface is usable. Availability is on a subsystem basis and the Matlab interface may be used in

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL006

whichever subsystem are available, while others may be down/offline.

The basic principal is that an object may be created for each subsystem, and the operations (sending requests or reading measurements) are available as methods of the command.

The basic object life cycle is:

- Initialize MATLAB session (melt_init)
- Create the object
  >> m4=melt_m4;
- Connect to the subsystem
  >> m4.connect;
- Interact with the subsystem
  >> m4.status;
  >> m4.sendSetPoints(array);
- When finished, disconnect from the subsystem
  >> m4.disconnect

Melt_init is the first command to be called, and it is called once only per Matlab session. This command sets up the java environment, identifies the network card connected to the MELT network, and sets up various communication middleware variables.

Many (ideally all) commands are interruptible if they do not conclude in a short time. There are no long-running commands. Commands may not return promptly when the control system software is not running. Ctrl-C may be used to interrupt such commands.

Network Infrastructure

MELT network infrastructure (Fig. 6) uses the architecture baselined for ELT. The LAN closely follows the telescope LAN design, with the Nexus switch planned for the Service Connection Points (SCP) in the field, and connected back to the computer room via single mode optical fibre. some characteristics are:

- SCP switch: IE4010: high-performance non-blocking switching capacity with 28 Gigabit Ethernet ports
- IGMP snooping enabled: listening to Internet Group Management Protocol (IGMP) network traffic to control delivery of IP multicasts

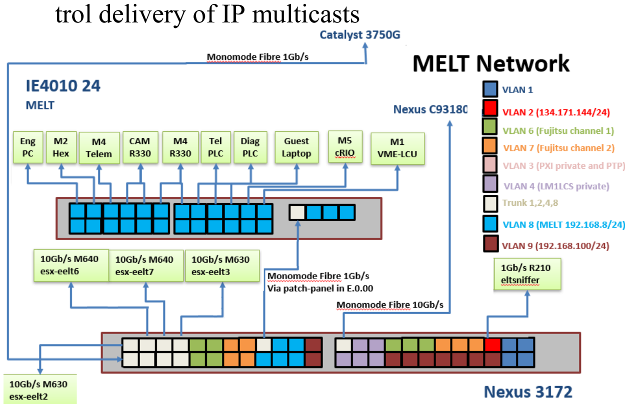

Figure 6: MELT switch layout.

To avoid accidental multicast flooding on the ESO network a Linux gateway is configured such that all outbound multicast can be blocked. The gateway Linux instance will be run as a VM one ESX server.

Infrastructure and Deployment

MELT Control System is deployed in a distributed environment comprised of different servers and machines (Fig. 7): 3x Dell PowerEdge r330 16RAM, XEON® CPU E3-1270 3.8Ghz: 1xCentOS 7.4, 1xCentOs7.4 with RT patch, 1xWindows, VME crater for the M1 LCU, 2x Beckhoff CX2030 PLC, 4xVLT like LCUs.

The two Linux hosts run the ELT Linux development Environment [8], comprised of:

- Support for C/C++, Java, Python programming languages and QT5.
- Build system: WAF
- Unit tests: googe tests, nosetests, testing
- Integration tests: Robot Framework
- Continuous integration: Jenkins
- OS: Linux CentOS7.4

Each PLC run TwinCAT3, which is a realtime kernel and development environment from Beckhoff automation, and implements I/O communication through EtherCAT, and has a motion module taking care of the calculations for the different axis doing motor control.

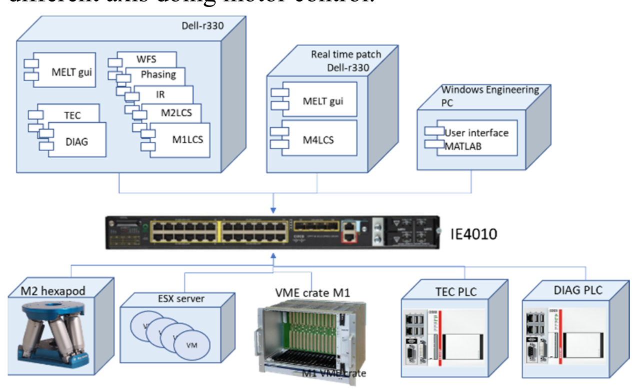

Figure 7: CS Infrastructure and deployment view.

Communication Patterns

As in ELT Control System, two communication patterns are used: publish/subscribe and request/reply, then they are mapped to the underlaying communication middleware software stack, all abstracted within the Core Integration Infrastructure Middleware abstraction layer (Fig. 8).

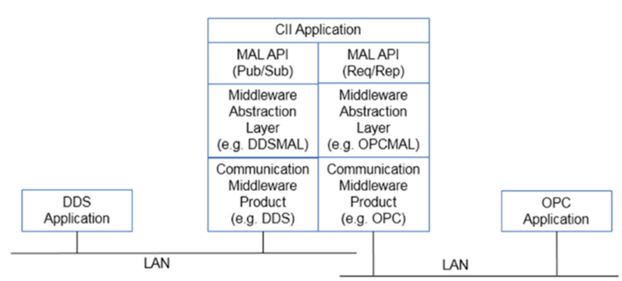

Figure 8: MELT communication stack

Specifics of the MELT's usage of MAL:

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEMPL006

- publish/subscribe via DDSMAL: Specific QoS libraries and profiles created for setting reliable_reliability_qos (DDS will attempt to deliver all samples in its history. Missed samples may be retried) and transient_local_durability_qos (can be delivered to any potential late-joining) properties.
-
- Request/Reply via OPCMAL [2]: Usage of Remote Procedure Calls (RPCs) allow MELT CS to control e.g. the PLC controlled motors or Flipper stage tip tilt.
- Publish/subscribe via MUDPI MAL: Multicast UDP Interface (MUDPI) contains no transaction or session requirements, it is little more than a standardized wrapper for UDP payloads with some additional constraints. This very simplicity, however, is one of its key requirements, making it suitable for use in Ethernet-based distributed control loops and high performance interfaces of the ELT Control System across languages and architectures

Additionally, Real-Time MUDPI Stream (RTMS) protocol is used for the exchange of images between the cameras and the MELT abstraction layer. RTMS is a low-latency, deterministic communication meant for the Adaptive Optics (AO) Real-Time Computer (RTC) Hard Real-Time Core (HRTC) of the ELT.

SYSTEM PERFORMANCE

Several equipment running at different rates are integrated in the bench. The control loop runs subsystems at different rates, collecting observable data at frequencies up to 1KHz. The most demanding devices, in terms of performance required, are the ASM and M4DM. ASM performance test measurements show that the difference between the expected send time and the actual one is 0.1uSec.

Regarding the deformable mirror, it is shown in Fig. 9, the latency of 130 us from first byte reception to the application of setpoints, with the system running at 1KHz. Image is obtained with an oscilloscope.

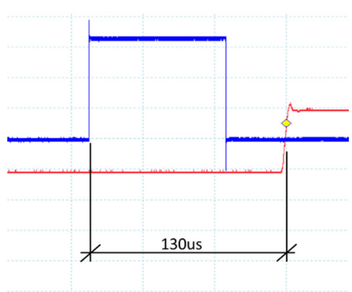

Figure 9: M4 setpoint application performance.

CONCLUSION

We have presented the Control system and detailed Software design, used for the MELT project within the EL program. Similarities (technology wise) to the ELT control approach has been shown and discussed. MELT is now ready to be used to develop the design strategy for the PDS and will in the future help to derive its technical specifications.

With its capability to adapt to other wavefront control strategies, MELT enables us to find the best starting strategy, when this task is to be used at the ELT. In addition, the central control system of the ELT can already now interface with real hardware and validate software work on the bench that is outsourced. Over the following years, the presented design will most certainly not stay static, but exhibits changes to the needs that result from the usage of MELT. We hope that this learning experience will help us prepare for the ELT commissioning, as discussed at the beginning.

ACKNOWLEDGEMENTS

The authors would like to thank the entire APE team for their work, which provides the grounds used to extend on with the present project. Additionally, the support received from the CII and IFW Software development teams has been of great help, and has allowed a successful procurement of the MELT Control System.

REFERENCES

[1] T. Pfrommer *et al.*, "MELT: an optomechanical emulation testbench for ELT wavefront control and phasing strategy", in *Proc. SPIE 10700, Ground-based and Airborne Telescopes* VII, no. 107003F, Jul. 2018.

[2] OPC Unified Architecture, `https://opcfoundattion.org/about/opc-technologies/opc-ua/`

[3] Rti DDS, `https://community.rti.com/rti-doc/500/ndds.5.0.0/doc/html/api_java/index.html`

[4] ZeroMQ, `https://zeromq.org`

[5] Protocol buffer, `https://developers.google.com/protocol-buf`

[6] OpenCV, `https://pypi.org/project/opencv-python/`

[7] QT framework, `https://www.qt.io/`

[8] F. Pellegrin, and C. Rosenquist, "The ELT Linux Development Environment", in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Control Systems (ICALEPCS'17), Barcelona, Spain, Oct..2017*, doi:10.18429/JACoW-ICALEPCS2017-THBPL05