ELT Instrument control software framework
Development status and future challenges.

TETIS Workshop October 29th 2020
Mario Kiekebusch on behalf the development team, ESO
Agenda

- Overview
- Project Status & Planning
- Software Engineering
- Summary
Overview
Future ELT/VLT Instruments

- ELT
  - HARMONI
  - MICADO and MAORY
  - METIS
  - HIRES
  - MOSAIC
  - PCS

- VLT
  - FORSUP
  - MAVIS
  - CUBES
The ELT Instrument Control Software Framework (IFW) is a toolkit aimed to help instrument developers implementing the control software for their instruments.

The IFW reuses the proven architectural and design patterns from the VLT instrument framework but is implemented using the new technologies defined by the ELT development standards.

The IFW implement generic solutions that can be configured and extended for each application.
ELT Control SW

Applications

Base SW stack

Dependencies

IFW

RTC Tk

NGC2

CII-MAL + Services

DevEnv

INS1

INS2

AO Module
Technologies

PLCs:

- Industrial standard widely used. Solving many of the control needs of instruments.
- Supports multiples fieldbus protocols (EtherCAT, serial, canbus, ethernet, etc.)
- Development Environment: MS Visual Studio and TwinCAT
- Programming Languages: Structured Text (ST) and C++
- Communication Interface OPC-UA.
  - Remote Procedure Calls
  - Read/Write and subscriptions
Technologies

WS part

Server:
  • Dell M8xx Blade chassis, (IT Standards, TBD).
  • OS: Linux (CentOS 7.6)

Programming Language: C++17 and Python.
Python Binding: pybind11
Middleware: ZMQ
Graphical Interfaces: Qt

Testing Frameworks: Google Unit Tests, Robot Framework.

PTP time protocol
Technologies

- **Redis DB**

- **Nomad**
  - Nomad is a flexible deployment tool used normally in data centers.

- **Nix**
  - A powerful package manager for Linux and other Unix systems that makes package management reliable and reproducible.
Existing IFW components (V2)

- **Application Framework (RAD)**
  - Enables the development of event-driven applications based on call-backs and state machines.

- **Test Framework (ETR)**
  - Standardizes the way to set up and execute tests using existing test runners.

- **Core Libraries (CDT and DIT)**
  - General purpose libraries.

- **Function Control Framework (FCF)**
  - Controls and monitor instrument hardware functions and sensing systems.

- **Online Data Processing (ODP)**
  - A simple component aiming to provide a data processing toolkit, flexible and well integrated with the Instrument Control Software.

- **Sequencer (SEQ)**
  - A generic tool for the execution of Observation Blocks (OB) and engineering scripts.
A generic tool for the execution of Observation Blocks (OB) and engineering scripts.

Being implemented in Python. All scripts will be python scripts.

Decouples execution engine from graphical tool.

A template library will simplify the interface with subsystems.
Sequences are modeled as a Directed Acyclic Graph (DAG).

Nodes state machine
Sequencer

Simple example of a sequencer script

```python
import asyncio
from seqlib.nodes import Sequence, Action

def do_a():
    """An Action does not need to be a coroutine""
    print("A")

async def do_b():
    """But coroutines are pretty neat""
    await asyncio.sleep(1)
    print("B")

def create_sequence():
    """Creates a simple sequence""
    return Sequence.create(
        Action(do_a, name="A"),
        do_b, # syntax sugar — automagically converts to Action
        name="Tut_01",
    )
```
Sequencer
A set of *PLC libraries* implementing the supported device controllers, simulators and their HMIs for local control.

A *Device Manager* controlling a configurable number of devices from a standard ELT WS.

A set of *Device Simulators* capable of emulating the behavior of a device controller and its interface within a WS.

A generic *GUI* for the Device Manager that allow users to control devices graphically.
Function Control Framework

Device Manager

- State Machine Engine
- Façade Class
- Device Factory

Device Classes

Linux Host (ELT DevEnv)

OPCUA Server

- RPC calls
- DA Read
- DA Subscriptions

Beckhoff IPC

Win OS

OPCUA

ADS

EtherCAT

Master

I/O

TwinCAT Kernel

TwinCAT PLC

redis

ØMQ

- Request/Reply
- Pub/sub

pymotogui

protobuf

fcfgui

Simulator

OPCUA Server

Hardware Adapter, e.g. Motion Controller.

Hardware Function, e.g. Filter Wheel with its actuator.
Function Control Framework
Data Display Tool

- The DDT is a framework to implement quicklook tools for different data types.
- Is being implemented in Qt with python bindings.
- The DDT software is split into four major components
  - Data Transfer
  - Image Handling
  - Data Visualization
  - Python Components
The DDT reference application (early version).
Data Display Tool

- Example of other data types.
- Actuator values are mapped to a pixel display which represents its physical shape.
Project Status & Planning
Timeline

- Jun 16: Start
- Apr 17: IFW development
- May 19: IFW Version 1
- May 20: IFW Version 2

- Prototyping work/wtools/DevEnv
- Requirements, design and software development
- Software development CII integration

- 4 years of development
- 12 FTEs spent.
- More than 100k lines of code
# Status & Planning

<table>
<thead>
<tr>
<th>Software Component</th>
<th>2019 30/05</th>
<th>2020 30/05</th>
<th>2021 30/05</th>
<th>2022 30/05</th>
<th>2023 30/05</th>
<th>2024 30/05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Framework</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
</tr>
<tr>
<td>Function Control Framework</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
</tr>
<tr>
<td>Widget Library</td>
<td></td>
<td></td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
</tr>
<tr>
<td>Observation Coordination Framework</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera Control Framework</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Display Tool</td>
<td>delayed</td>
<td></td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
</tr>
<tr>
<td>Sequencer</td>
<td>A</td>
<td></td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
</tr>
<tr>
<td>Template Library</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calibration Framework</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>V1</td>
<td>V2</td>
</tr>
<tr>
<td>Online Data Processing</td>
<td>A</td>
<td></td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
</tr>
<tr>
<td>Test Framework</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
</tr>
<tr>
<td>Miscellaneous Libraries</td>
<td>V1</td>
<td>V2</td>
<td>V3</td>
<td>V4</td>
<td>V5</td>
<td>V6</td>
</tr>
</tbody>
</table>
Software Engineering Process
Work is based on well defined iterations called sprints.

Stories are the features of the system to be implemented.

Backlog is the repository of features (stories)
Development Process
Test & Verification

- Set of Jenkins Pipelines
  - CI Pipeline (build + execution of unit tests)
  - CI Pipeline of DevEnv beta
  - Integration Pipeline (build + unit tests + integration tests)
  - Daily IFW documentation (build + doxygen & sphinx)
  - Daily and Weekly IFW statistics (SLOC, static analysis, code coverage)
  - Daily and Weekly IFW valgrind
Simulation of hardware devices.

- Simulators can be controlled externally to emulate certain behavior on demand.

Test Instrument with hardware devices.

Deployment in other test facilities, e.g. MELT
Major release lifecycle

- One major version will be released per year, following the IFW Release Plan.
- Unstable versions might be created at the end of a sprint.
Patch release lifecycle

A patch release can also be created when urgent bugs need to be fixed in a previous release delivered to externals.
IFW problems reported in existing JIRA project for instrument follow-up.

This allows a filtering process and a more dedicated effort from follow-up team to deliver the answers to Consortia.
ESO will continue developing the IFW according to its development plan.

IFW started to be used by instrument developers specially to control hardware devices.

A significant work is ahead to adapt to CII services.

DDT is delayed but first version is expected for next IFW version. An alpha version may be made available in December.