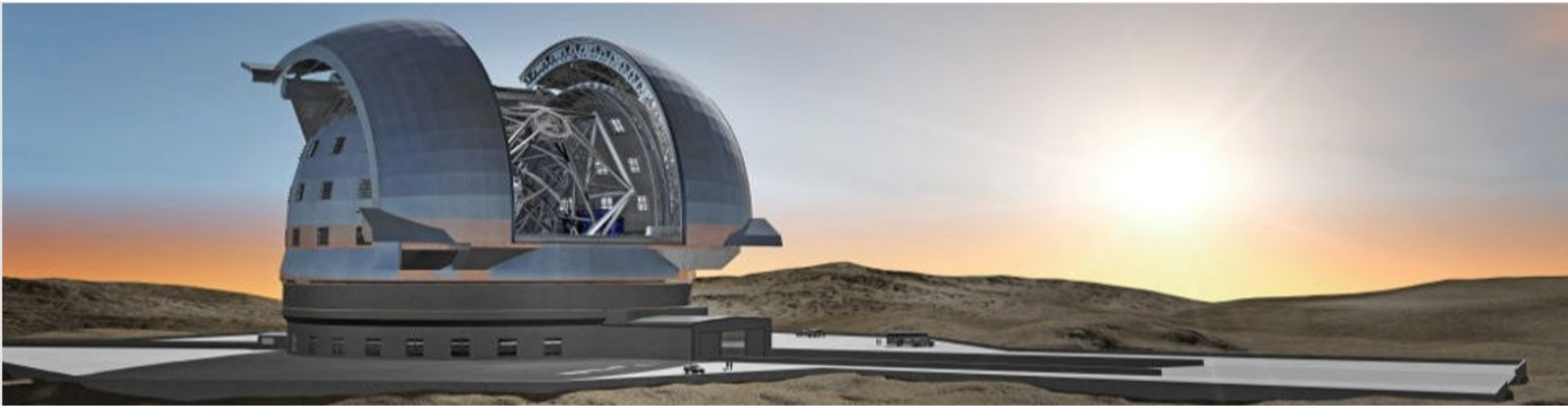
ELT Instrument control software framework

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

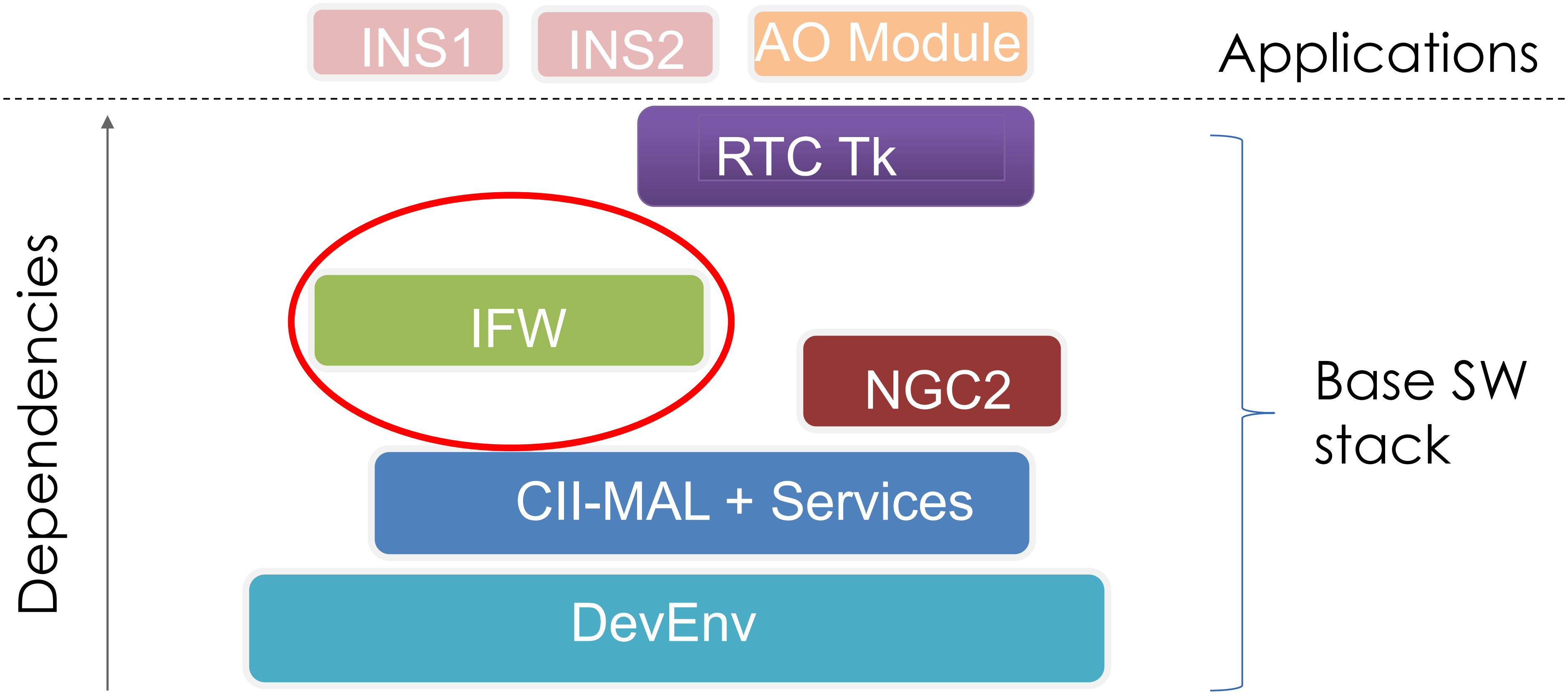
Future ELT/VLT Instruments

- ELT
  - **HARMONI**
  - **MICADO and MAORY**
  - **METIS**
  - HIRES
  - MOSAIC
  - PCS
- VLT
  - FORSUP
  - MAVIS
  - CUBES

Definition

- 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



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)

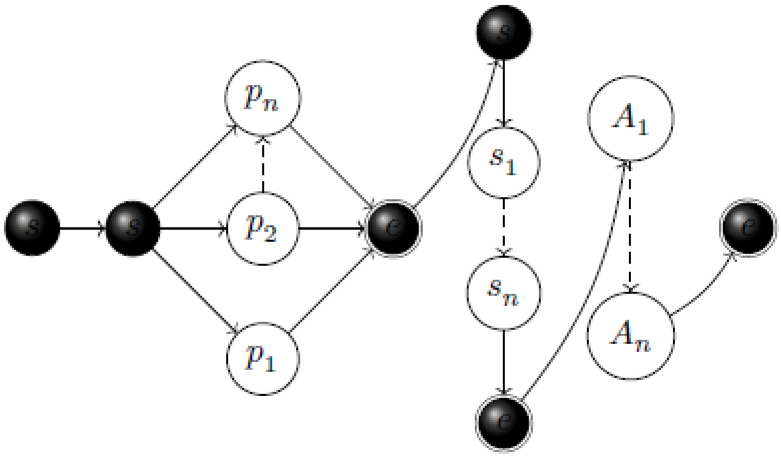
- Application Framework (RAD)
  - Enables the development of event-driven applications based on call-backs and state machines.
- Test Framework (ETR)
  - Standardizes the way to set up and execute tests using existing test runners.
- Core Libraries (CDT and DIT)
  - General purpose libraries.
- Function 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.

Sequencer

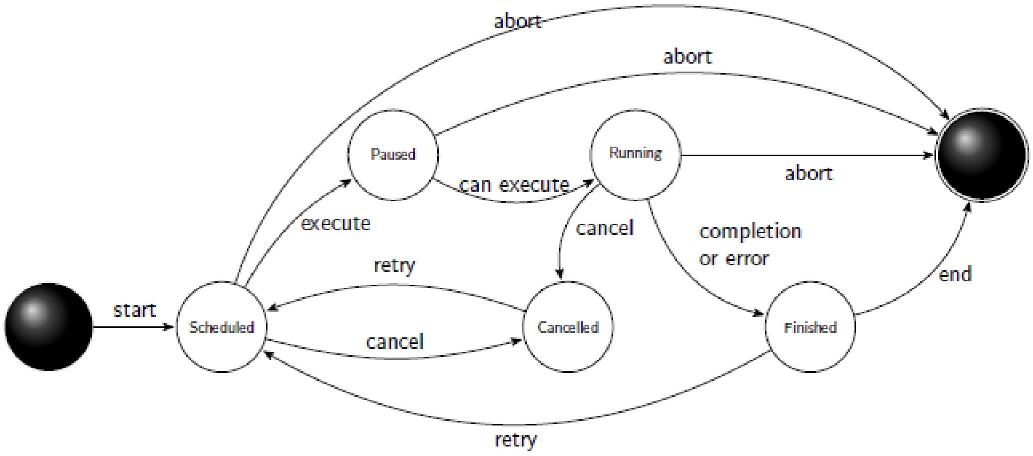
- A generic tool for the execution of Observation Blocks (OB) and engineering scripts.
- Being implemented in Python. All scripts will be python scripts.
- Decouples execution engine from graphical tool.
- A template library will simplify the interface with subsystems.

Sequencer

- Sequences are modeled as a Directed Acyclic Graph (DAG).

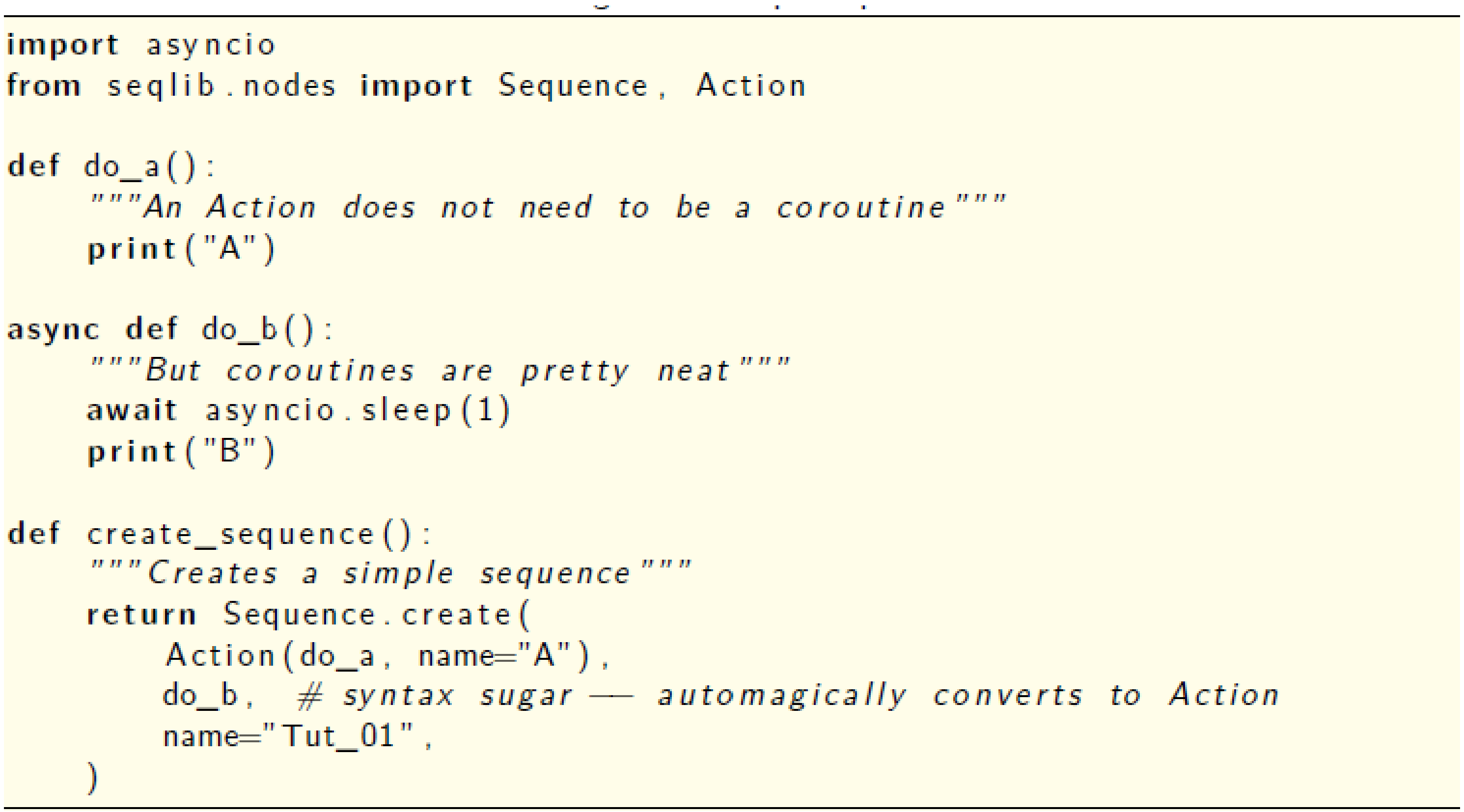


- Nodes state machine



Sequencer

- Simple example of a sequencer script



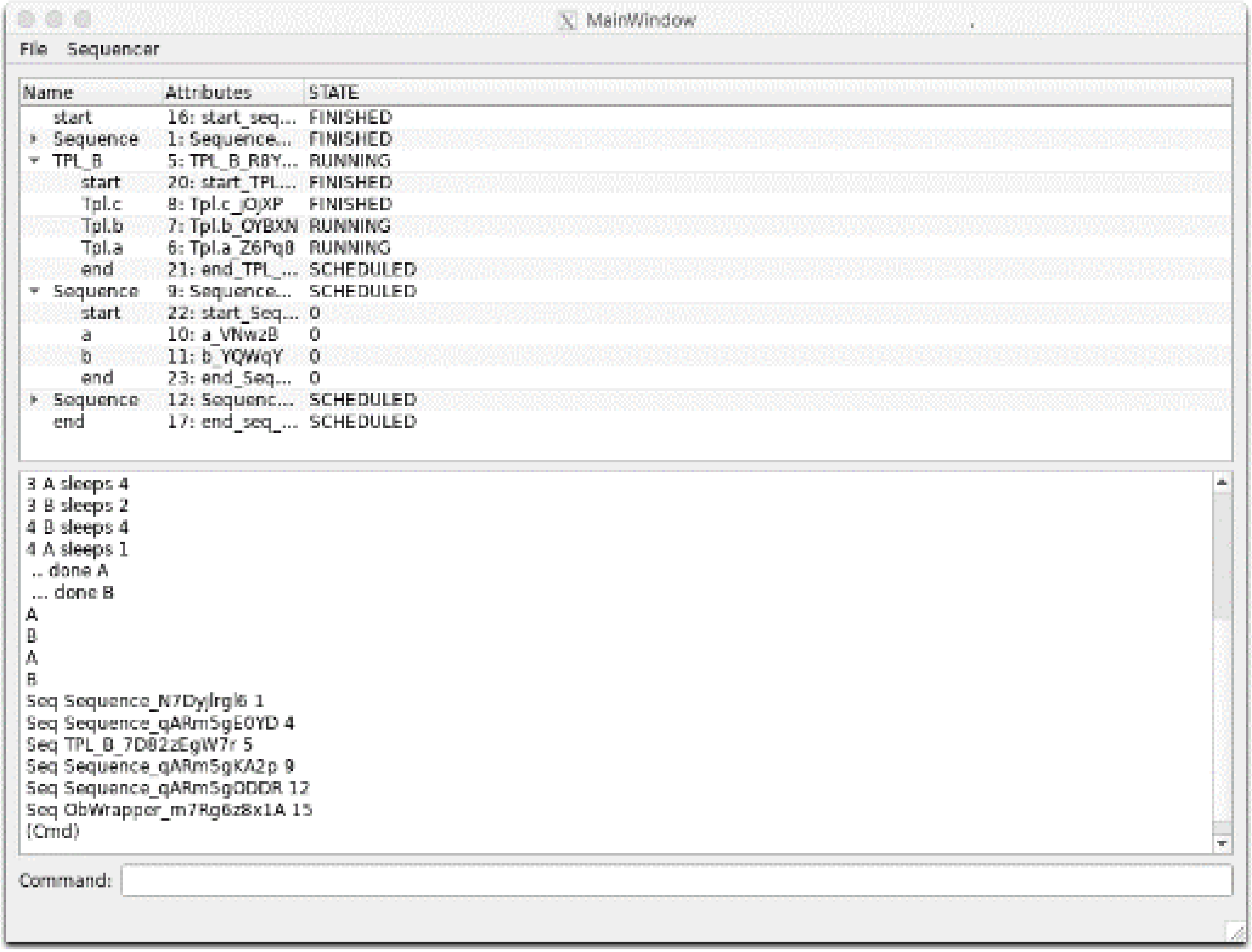
```
import asyncio
from seqlib.nodes import Sequence, Action

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



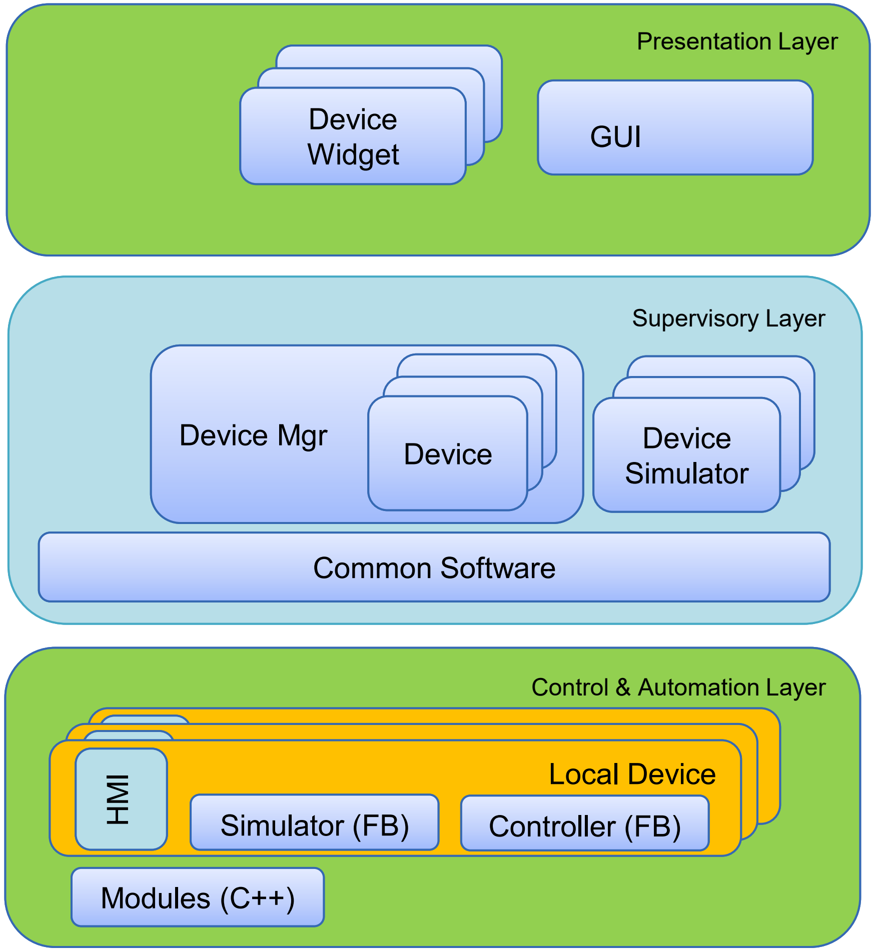
Name	Attributes	STATE
start	16: start_seq...	FINISHED
Sequence	1: Sequence...	FINISHED
TPL_B	5: TPL_B_R8Y...	RUNNING
start	20: start_TPL...	FINISHED
Tpl.c	8: Tpl.c_jOjXP	FINISHED
Tpl.b	7: Tpl.b_OYBXN	RUNNING
Tpl.a	6: Tpl.a_Z6Pq0	RUNNING
end	21: end_TPL_...	SCHEDULED
Sequence	9: Sequence...	SCHEDULED
start	22: start_Seq...	0
a	10: a_VNwzB	0
b	11: b_YQWqY	0
end	23: end_Seq...	0
Sequence	12: Sequenc...	SCHEDULED
end	17: end_seq_...	SCHEDULED

```
3 A sleeps 4
3 B sleeps 2
4 B sleeps 4
4 A sleeps 1
 .. done A
 ... done B
A
B
A
B
Seq Sequence_N7DyjIrgl6 1
Seq Sequence_qARm5gE0YD 4
Seq TPL_B_7D02zEgW7r 5
Seq Sequence_qARm5gKA2p 9
Seq Sequence_qARm5gODDR 12
Seq ObWrapper_m7Rg6z8x1A 15
(Cmd)
```

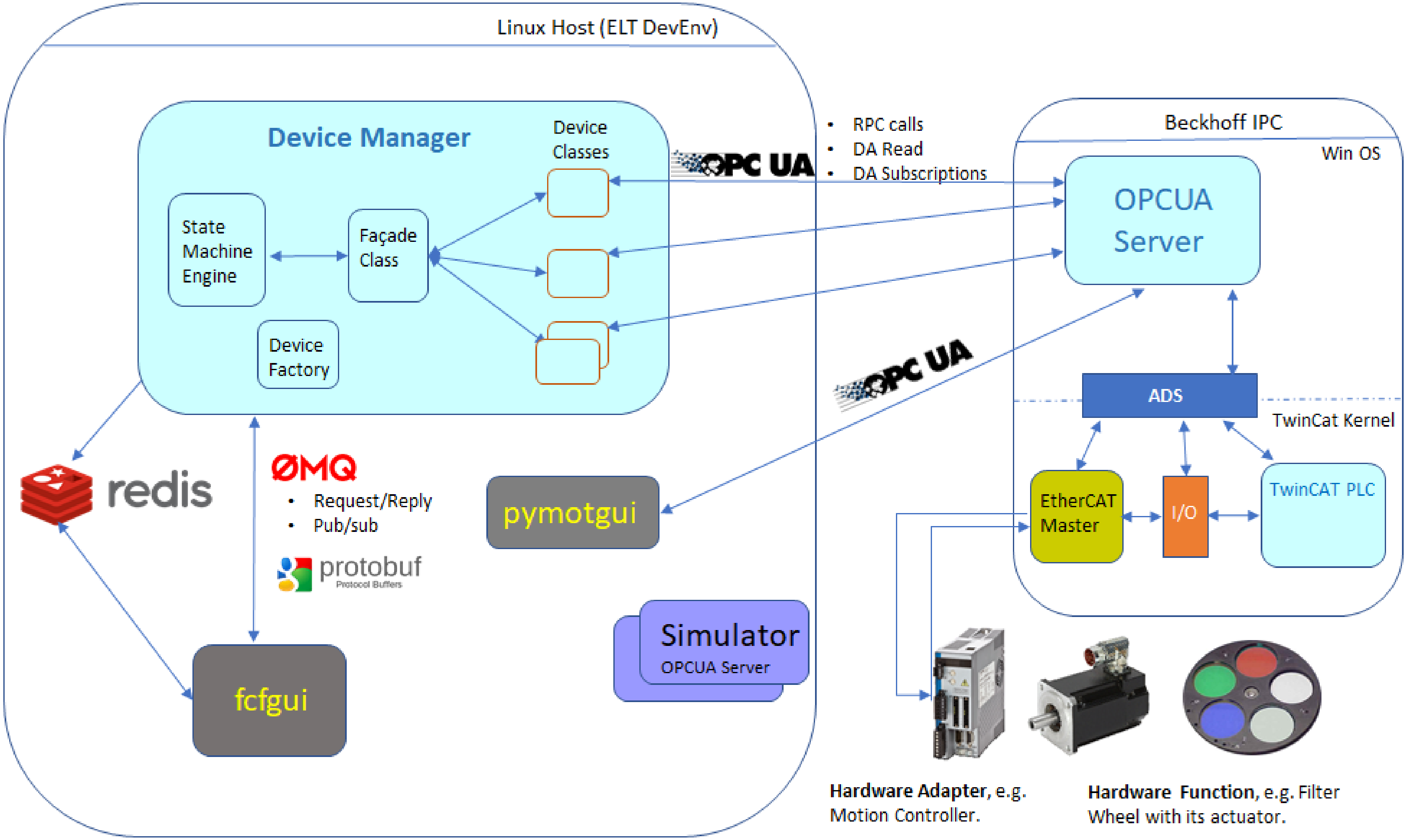
Command:

Function Control Framework

- 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.



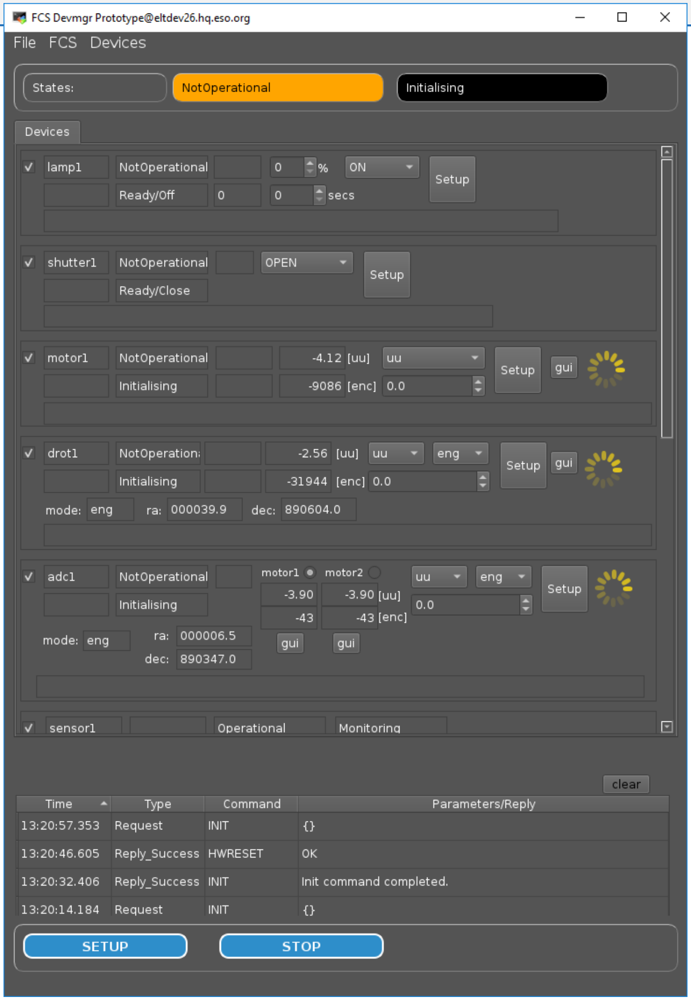
Function Control Framework



Hardware Adapter, e.g. Motion Controller.

Hardware Function, e.g. Filter Wheel with its actuator.

Function Control Framework



FCS Devmgr Prototype@eltdev26.hq.eso.org

States: NotOperational Initialising

☑ lamp1 NotOperational 0 % ON Setup
Ready/Off 0 0 secs

☑ shutter1 NotOperational OPEN Setup
Ready/Close

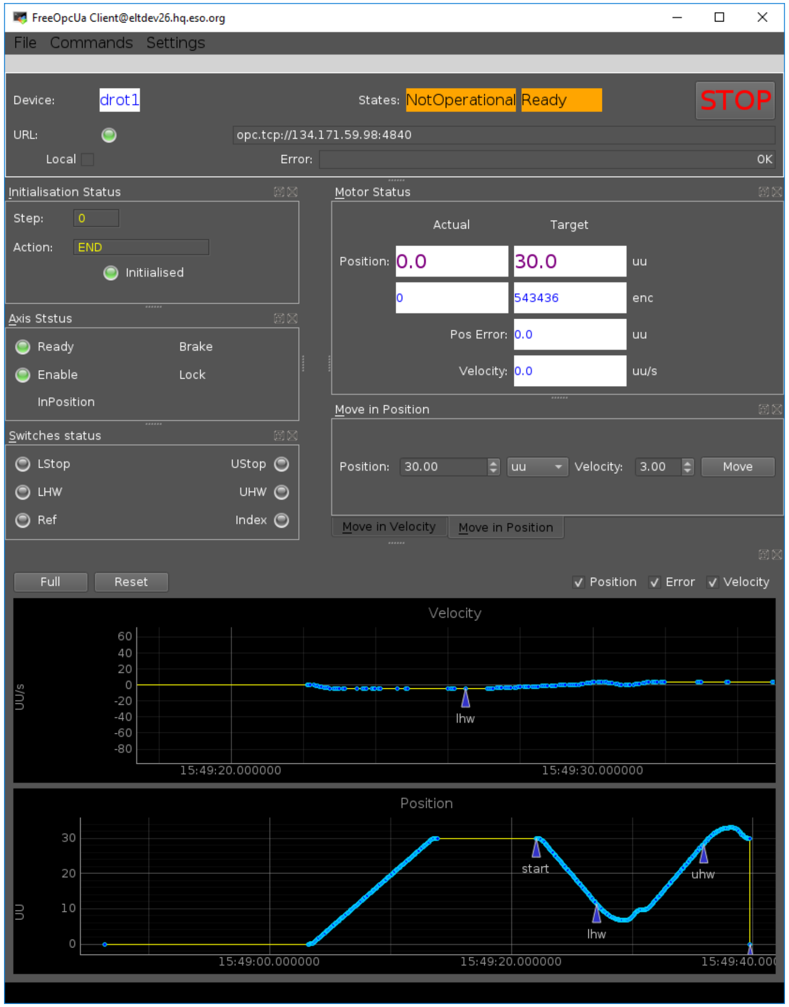
☑ motor1 NotOperational -4.12 [uu] uu Setup gui
Initialising -9086 [enc] 0.0

☑ drot1 NotOperation: -2.56 [uu] uu eng Setup gui
Initialising -31944 [enc] 0.0
mode: eng ra: 000039.9 dec: 890604.0

☑ adc1 NotOperational motor1 ● motor2 ○ uu eng Setup
Initialising -3.90 -3.90 [uu] 0.0
-43 -43 [enc]
mode: eng ra: 000006.5 gui gui
dec: 890347.0

☑ sensor1 Operational Monitoring

Time	Type	Command	Parameters/Reply
13:20:57.353	Request	INIT	{}
13:20:46.605	Reply_Success	HWRESET	OK
13:20:32.406	Reply_Success	INIT	Init command completed.
13:20:14.184	Request	INIT	{}



FreeOpcUa Client@eltdev26.hq.eso.org

Device: drot1 States: NotOperational Ready STOP
URL: ● opc.tcp://134.171.59.98:4840
Local □ Error: OK

Initialisation Status
Step: 0
Action: END
● Initiialised

Axis Ststus
● Ready Brake
● Enable Lock
InPosition

Switches status
○ LStop UStop ○
○ LHW UHW ○
○ Ref Index ○

Motor Status

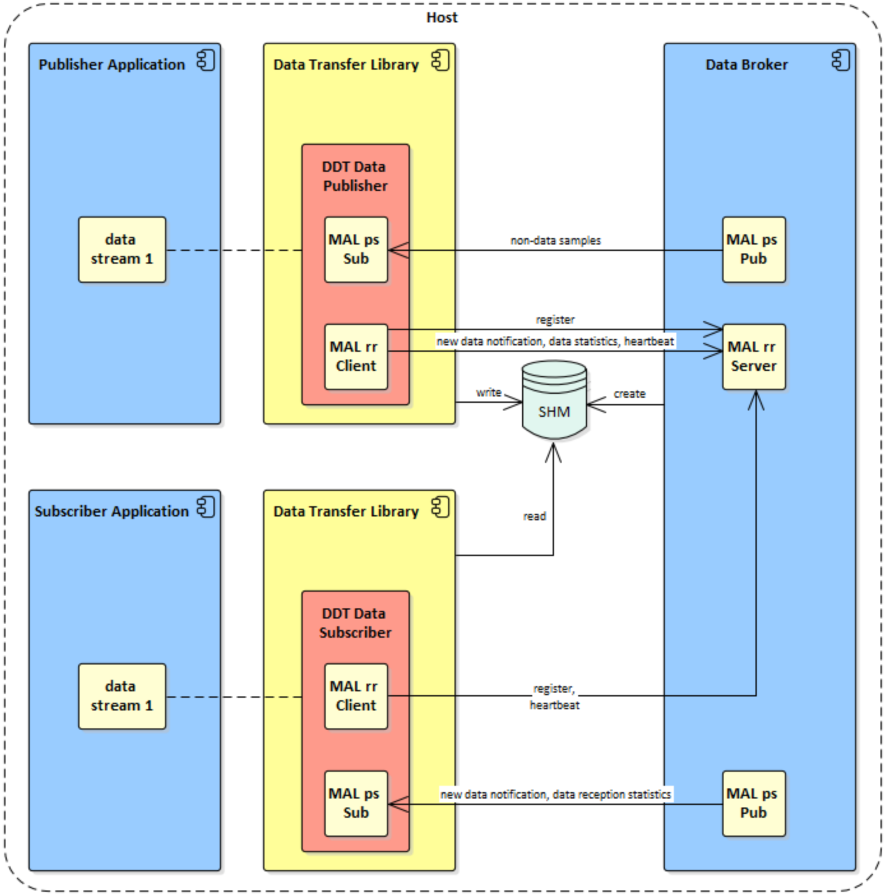
	Actual	Target	
Position:	0.0	30.0	uu
	0	543436	enc
Pos Error:		0.0	uu
Velocity:		0.0	uu/s

Move in Position
Position: 30.00 uu Velocity: 3.00 Move

Data Display Tool

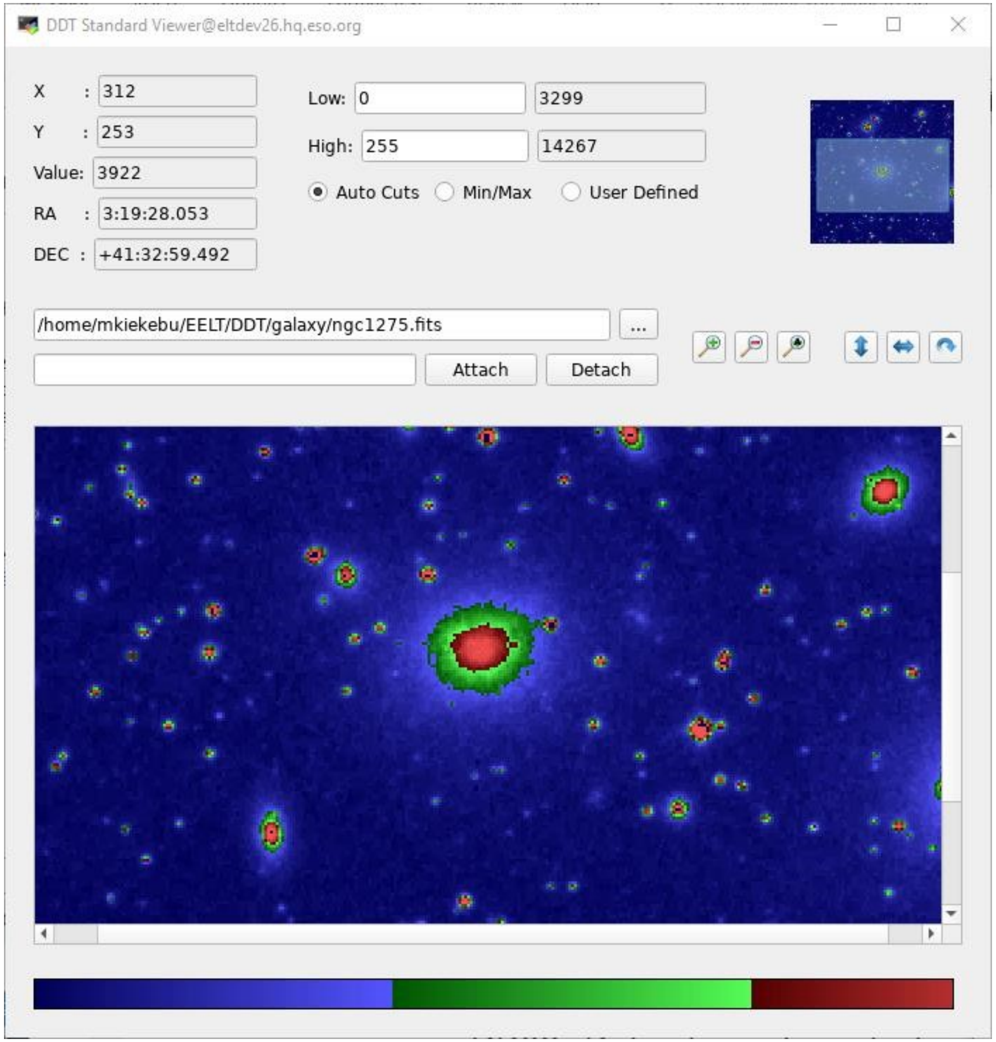
- The DDT is a framework to implement quicklook tools for different data types.
- Is being implemented in Qt with python bindings.
- The DDT software is split into four major components
  - Data Transfer
  - Image Handling
  - Data Visualization
  - Python Components

Data Display Tool



Data Display Tool

- The DDT reference application (early version).



DDT Standard Viewer@eltdev26.hq.eso.org

X : 312

Y : 253

Value: 3922

RA : 3:19:28.053

DEC : +41:32:59.492

Low: 0 3299

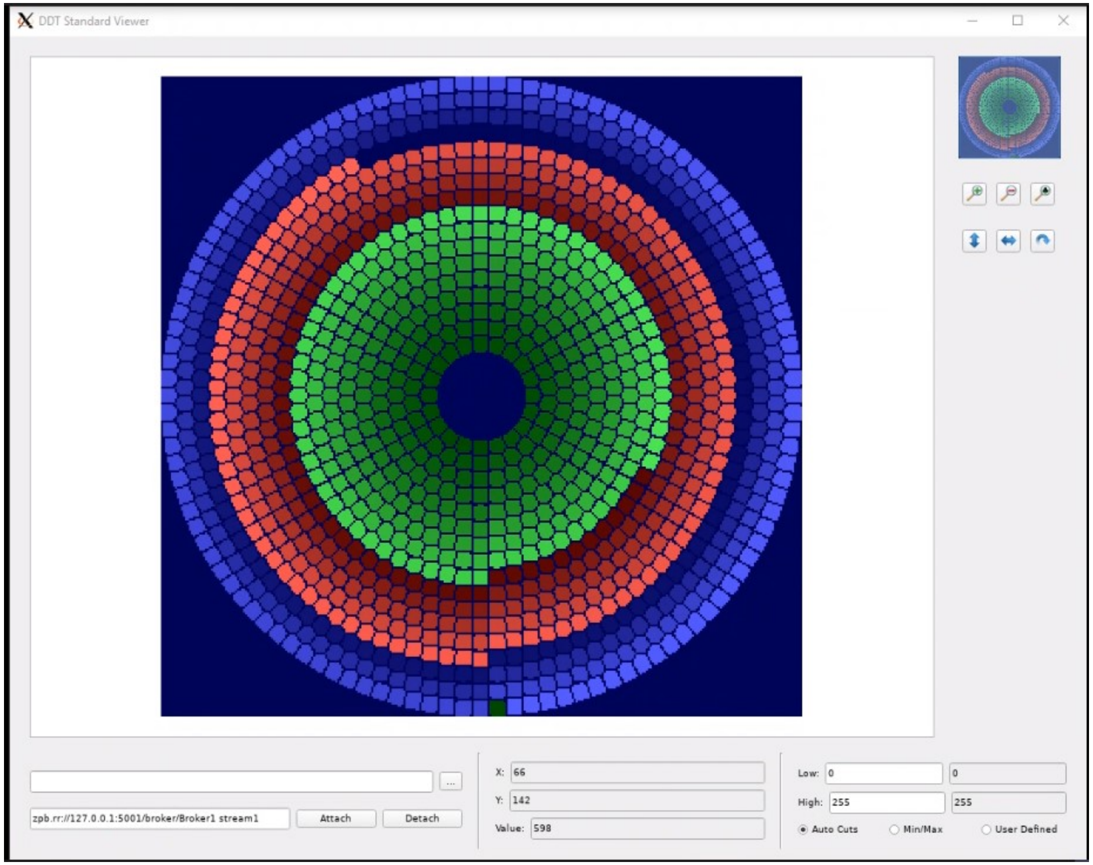
High: 255 14267

● Auto Cuts ○ Min/Max ○ User Defined

/home/mkiekebu/EELT/DDT/galaxy/ngc1275.fits

Data Display Tool

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



zpb.rr://127.0.0.1:5001/broker/Broker1 stream1

X: 66

Y: 142

Value: 598

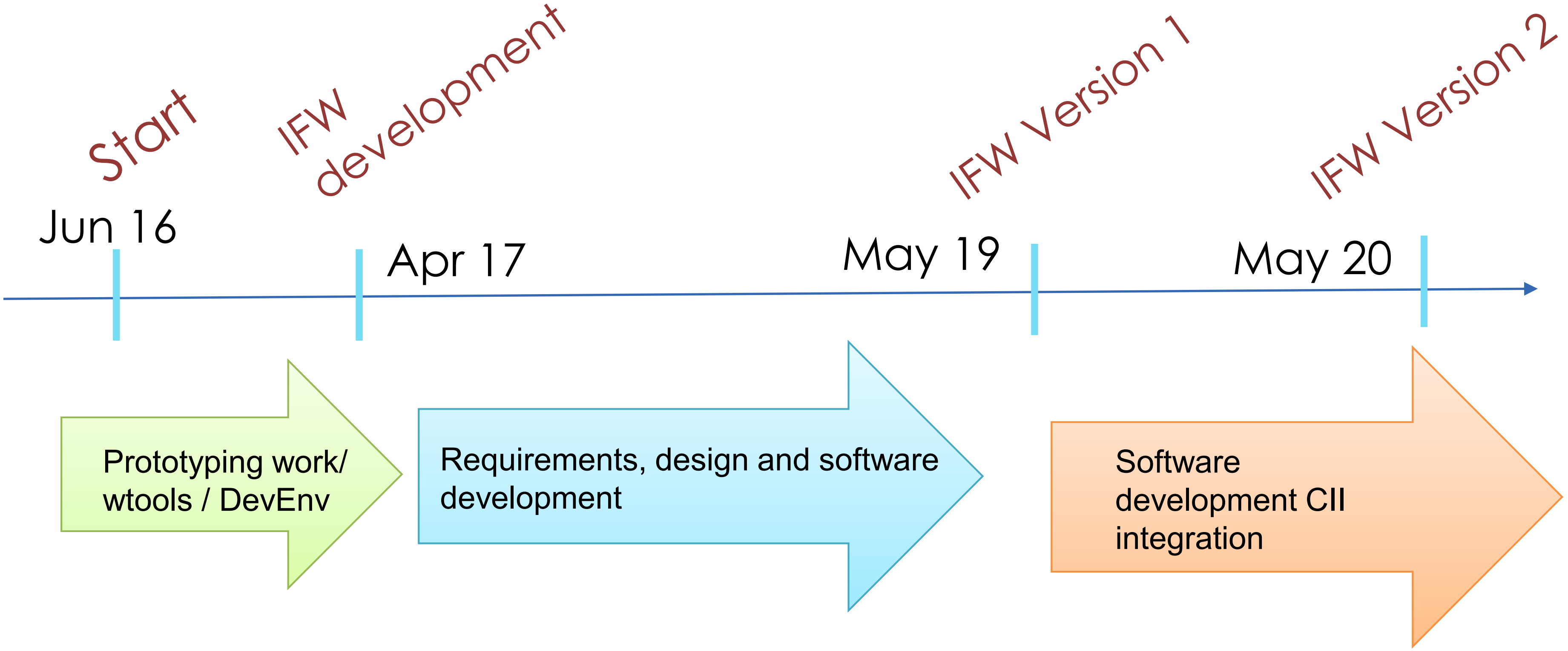
Low: 0 0

High: 255 255

● Auto Cuts ○ Min/Max ○ User Defined

Project Status & Planning

Timeline



Start

Jun 16

IFW development

Apr 17

IFW Version 1

May 19

IFW Version 2

May 20

Prototyping work/ wtools / DevEnv

Requirements, design and software development

Software development CII integration

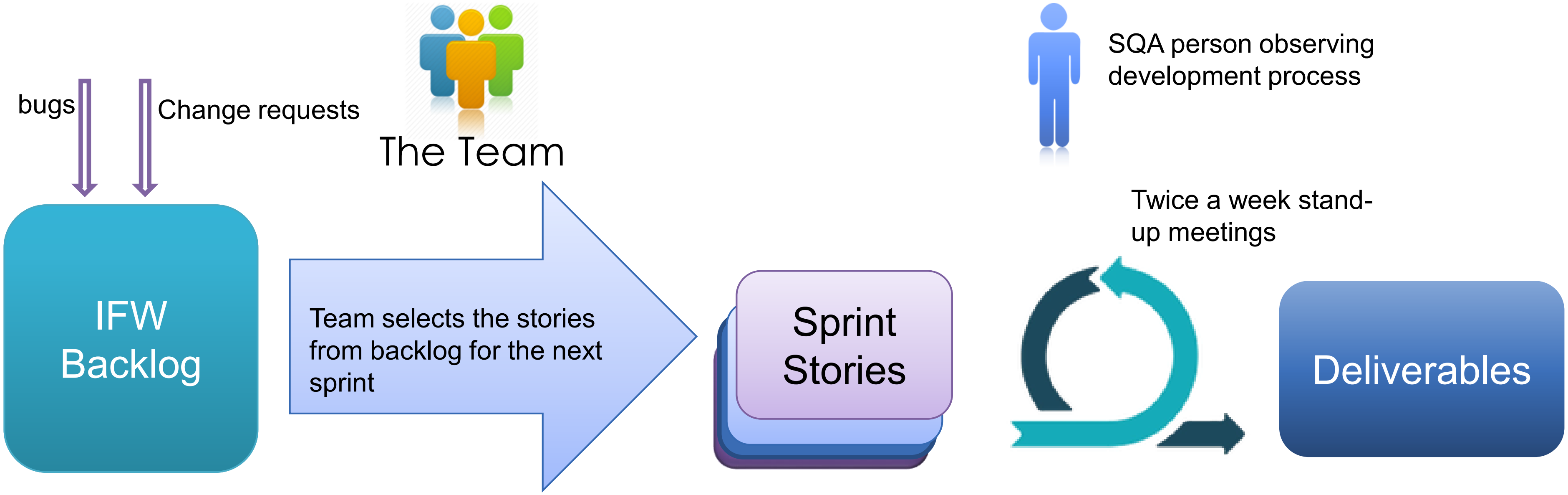
- 4 years of development
- 12 FTEs spent.
- More than 100k lines of code

Status & Planning

	2019 30/05	2020 30/05	2021 30/05	2022 30/05	2023 30/05	2024 30/05
Application Framework	V1	V2	V3	V4	V5	V6
Function Control Framework	V1	V2	V3	V4	V5	V6
Widget Library			V1	V2	V3	V4
Observation Coordination Framework			V1	V2	V3	V4
Camera Control Framework			V1	V2	V3	V4
Data Display Tool		delayed	V1	V2	V3	V4
Sequencer		A	V2	V3	V4	V5
Template Library			V1	V2	V3	V4
Calibration Framework					V1	V2
Online Data Processing		A	V1	V2	V3	V4
Test Framework	V1	V2	V3	V4	V5	V6
Miscellaneous Libraries	V1	V2	V3	V4	V5	V6

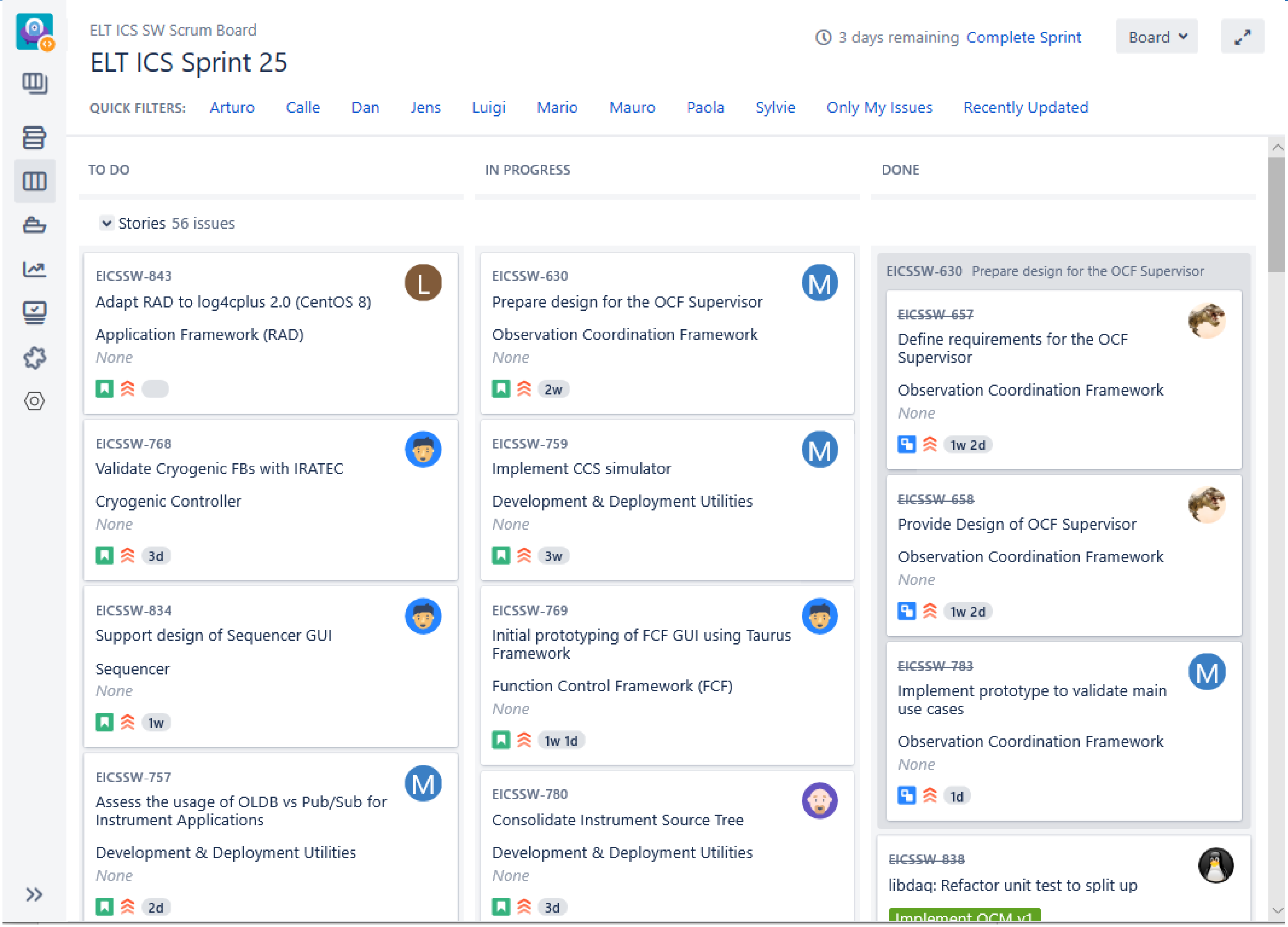
Software Engineering Process

Development Process


bugs
Change requests
The Team
SQA person observing development process
IFW Backlog
Team selects the stories from backlog for the next sprint
Sprint Stories
Twice a week stand-up meetings
Deliverables

- 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



ELT ICS SW Scrum Board

ELT ICS Sprint 25

3 days remaining Complete Sprint Board

QUICK FILTERS: Arturo Calle Dan Jens Luigi Mario Mauro Paola Sylvie Only My Issues Recently Updated

TO DO | IN PROGRESS | DONE

Stories 56 issues

TO DO

EICSSW-843
Adapt RAD to log4cplus 2.0 (CentOS 8)
Application Framework (RAD)
None

EICSSW-768
Validate Cryogenic FBs with IRATEC
Cryogenic Controller
None
3d

EICSSW-834
Support design of Sequencer GUI
Sequencer
None
1w

EICSSW-757
Assess the usage of OLDB vs Pub/Sub for Instrument Applications
Development & Deployment Utilities
None
2d

IN PROGRESS

EICSSW-630
Prepare design for the OCF Supervisor
Observation Coordination Framework
None
2w

EICSSW-759
Implement CCS simulator
Development & Deployment Utilities
None
3w

EICSSW-769
Initial prototyping of FCF GUI using Taurus Framework
Function Control Framework (FCF)
None
1w 1d

EICSSW-780
Consolidate Instrument Source Tree
Development & Deployment Utilities
None
3d

DONE

EICSSW-630 Prepare design for the OCF Supervisor

~~EICSSW-657~~
Define requirements for the OCF Supervisor
Observation Coordination Framework
None
1w 2d

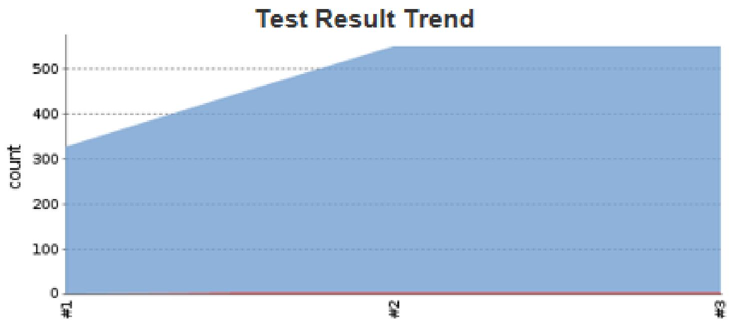
~~EICSSW-658~~
Provide Design of OCF Supervisor
Observation Coordination Framework
None
1w 2d

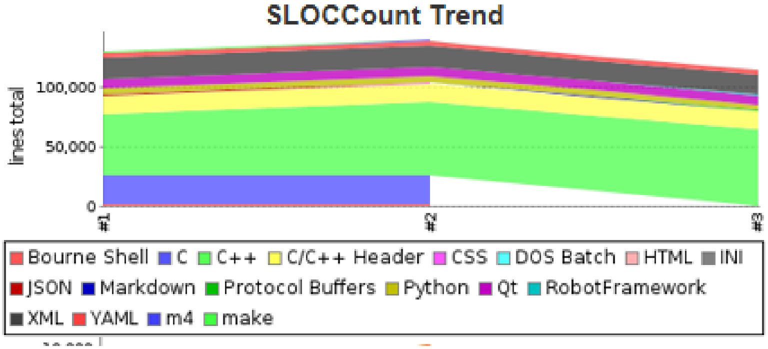
~~EICSSW-783~~
Implement prototype to validate main use cases
Observation Coordination Framework
None
1d

~~EICSSW-838~~
libdaq: Refactor unit test to split up
Implement OCM v1

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



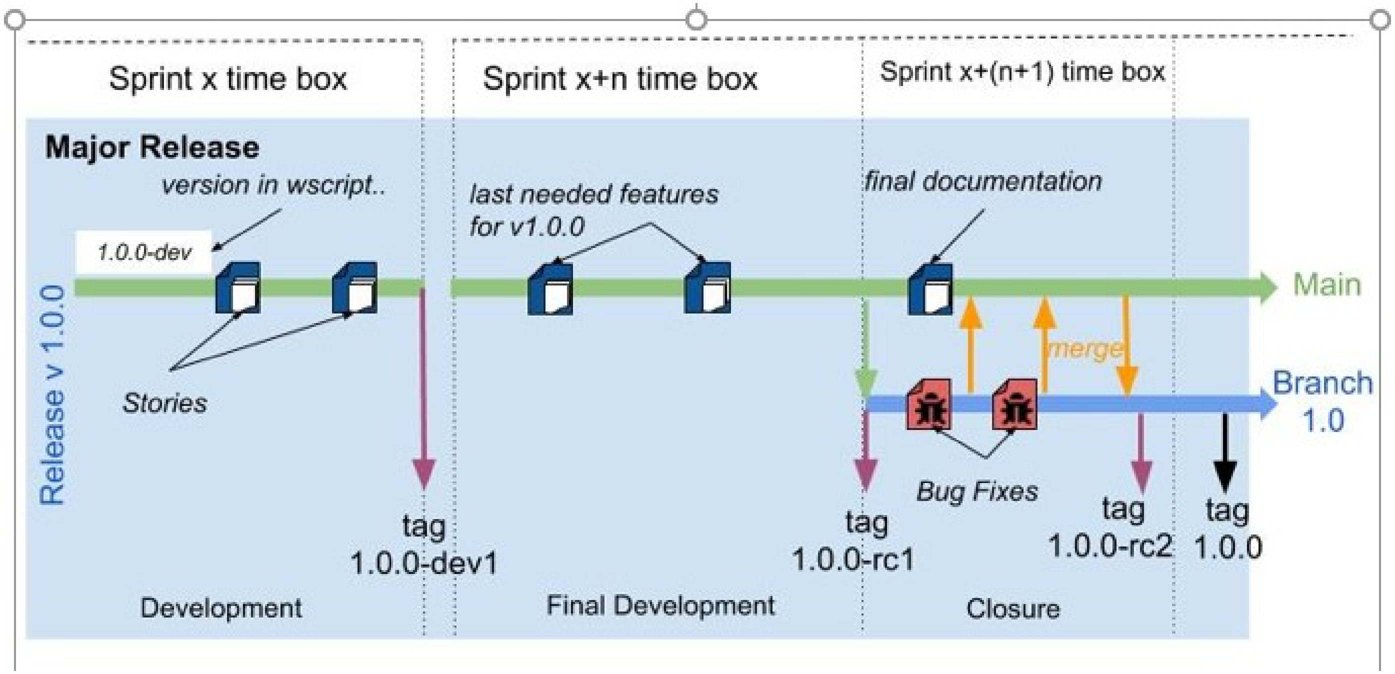


Test & Verification

- 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

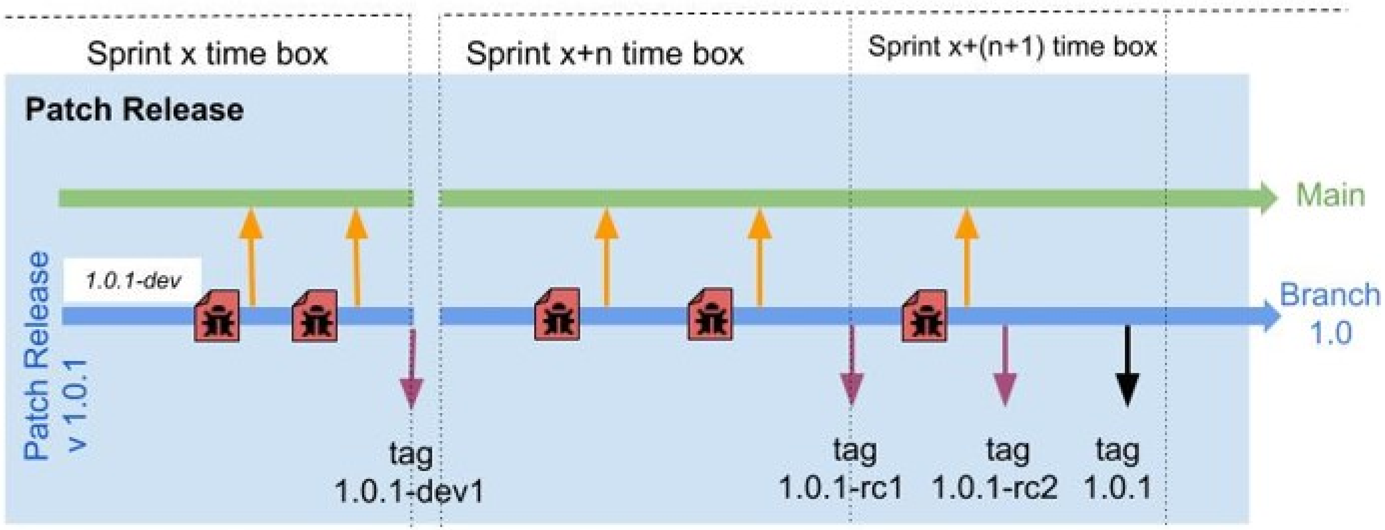
Release Management

- 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.



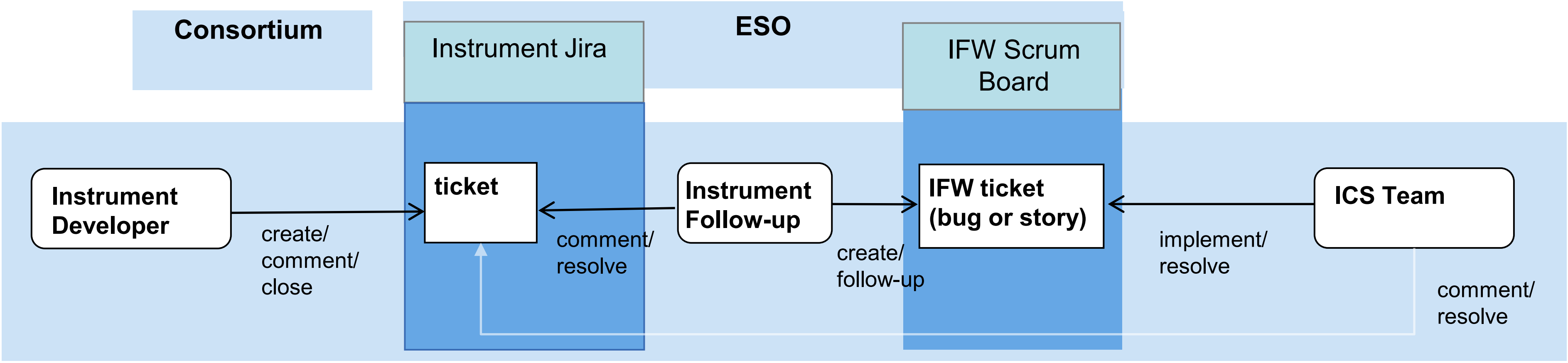
Release Management

- Patch release lifecycle
  - A patch release can also be created when urgent bugs need to be fixed in a previous release delivered to externals.



User Support

- 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.



Summary

- 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.