

Case-study: CERN Sequencer



Introduction to accelerator operations automation software

“Efficiency through Automation” Workshop

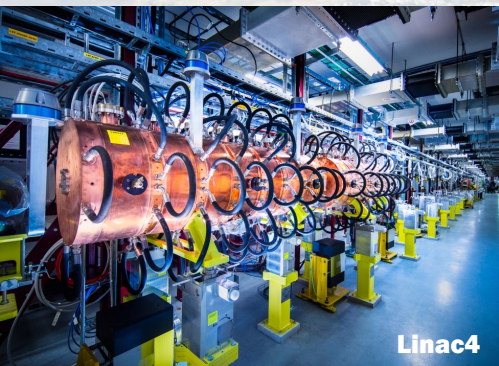
Lukasz Burdzanowski | CERN

8th October 2023

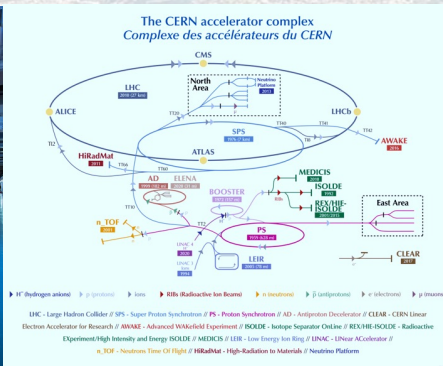
Why Sequencer

To reduce the number of manual actions needed to control the increasingly advancing machines and goals.

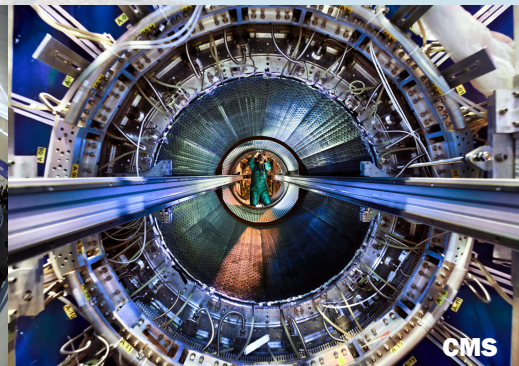
To shorten the turn-around times.



Linac4



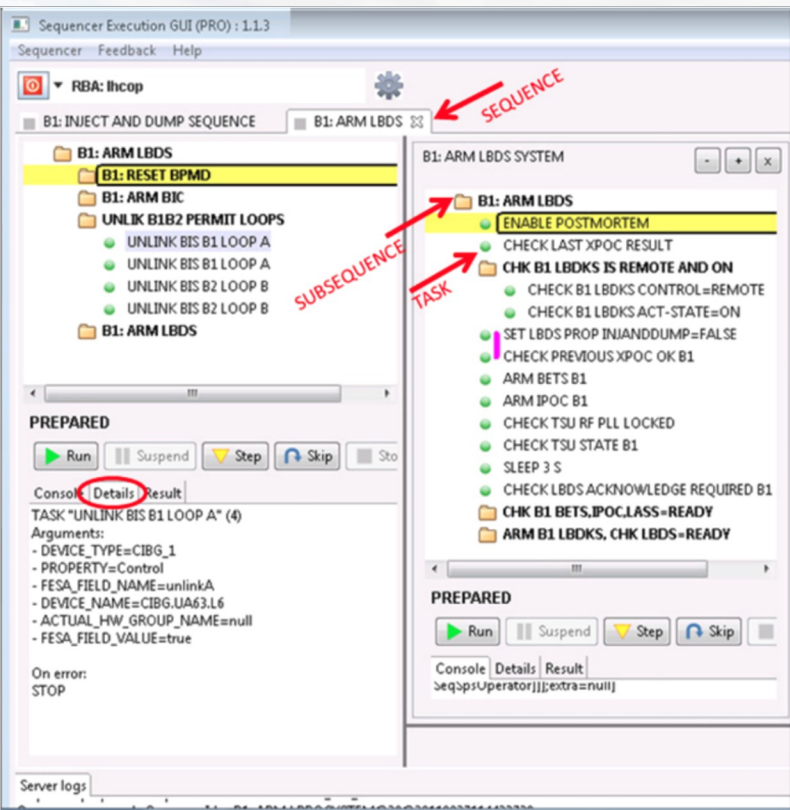
CERN Control Centre



CMS

The well-defined and deterministic problem to address:
efficiently execute n-steps to achieve the given state;
repeatedly and reproducibly.

Sequencer - automation of operations

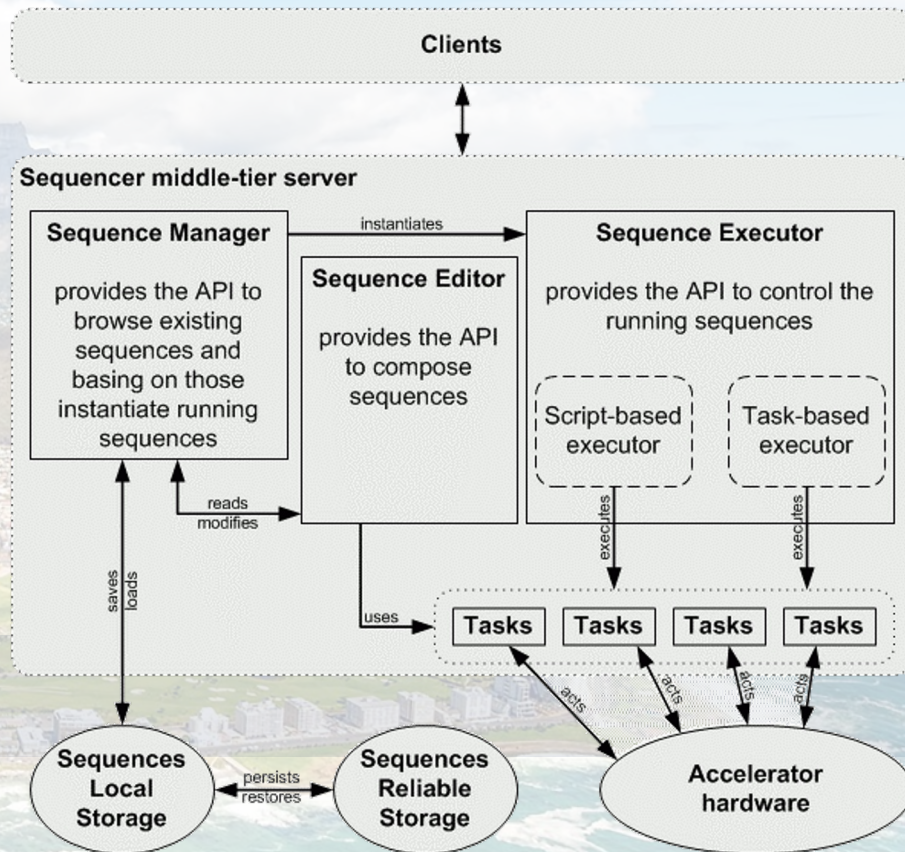
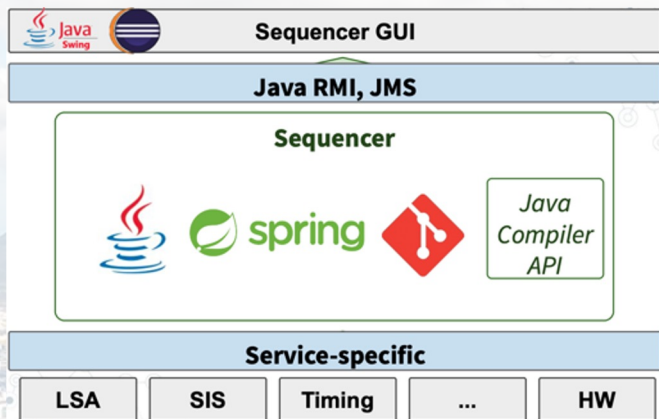


The Sequencer system: framework and dedicated applications, used to enable **automation of operations** and to help **drive the accelerators** through long **sequence of tasks** needed to produce the beam.

Provides full control over task execution, allowing:

- break-points,
- stepping, skipping, repeating, etc.
- executing the tasks in parallel.

System architecture



Technologies:

- Java (Spring, RMI), JMS for client-server (ActiveMQ),
- Eclipse SWT + JFace for GUI
- Sequences = Java classes in Git, *no RDBMS*

Except the authorisation (RBAC), Sequencer framework is not CERN specific

Sequencer in numbers

Framework: 1250 Java files, 105k LOC
Tasks utils & commons: 191 Java files, 14.5k LOC

Sequencer - a task

```
@Attrib(name = DEVICE_TYPE, description = "LSA device type", typeEnum = TypeEnum.LSA_DEVICE_TYPE)
@Attrib(name = LSA_HW_COMMAND, description = "HW command", typeEnum = TypeEnum.LSA_HW_COMMAND, value = HWC_LHC_COLL_LOAD_THRESHOLDS)
@TaskMethod(name = "LOAD COLL THRESHOLDS", description = "Load collimator thresholds", displayName = "Load collimator thresholds")
public static ArrayResult<?> loadCollThresholds(
    @Param(name = DEVICE, description = "Device name", typeEnum = TypeEnum.LSA_DEVICE) String deviceName,
    @Param(name = LSA_DEVICE_GROUP, description = "Device group name", typeEnum = TypeEnum.LSA_DEVICE_GROUP) String deviceGroupName,
    @Param(name = TIMING_USER, description = "Timing user (context)", typeEnum = TypeEnum.JAPC_SELECTOR) String timingUser)
    throws Exception {
    return loadCollThresholdsIgnoringResidency(deviceName, deviceGroupName, timingUser, false);
}
```

Takeaways:

- **tasks**, from interacting with HW equipment, to running the algorithms, are **procedures programmed in Java**
- code-completion in IDE for developers, **creating a task = writing code**
- **task meta-data** (@Attrib) can be used to **enable validation**, and to limit generic tasks to specific scenarios (e.g. machines)
- like any code, **tasks reside in VCS** (version control system)

Sequencer – a sequence

The screenshot displays the CERN Sequencer interface. On the left, the 'Edit sequence properties' dialog is open for a sequence named 'TEST LBU' with the description 'Test for ICALEPCS'. A dropdown menu for 'Type*' is open, showing options like 'CURRENT_A', 'ENERGY_GEV', 'LANDAU_DAMPING', 'JAPC_DEVICE_NAME', 'LSA_DEVICE_NAME', and 'LSA_DEVICE_GROUP'. The main window shows a tree view of tasks under 'TEST LBU', including 'Load collimator thresholds', 'Wait for any update from sparameterName', and 'Load crystal collimator thresholds'. On the right, the 'Edit task properties' dialog is open for the 'Load collimator thresholds' task, showing fields for 'Display Name', 'Execution directive', 'Attributes', and 'Arguments'.

The **sequence**, and a **sub-sequence**, is a named list of tasks.

- Sequences are created with a dedicated editor
- Tasks & sub-sequences can be parameterised
- End-result, the sequence is generated Java code
- There are no limits on sequences length

```
@SequenceInfo(displayName="PREPARE COLLISIONS", description="PREPARE COLLISIONS FROM 3.5 M SQUEEZED OPTICS", categories="DEVELOPMENT")
public class PREPARE_20COLLISIONS {
    public void exec() throws Exception {
        _displayName("PREPARE FEEDBACKS FOR PHYSICS");new PREPARE_20FEEDBACKS_20FOR_20PHYSICS().exec();
        _displayName("ENSURE START_COLLISIONS TABLE LOADED");CBCM.ensureEventTableLoaded("Start_Collisions");
        _displayName("MOVE STATE/BEAM_MODE = ADJUST");new MOVE_20STATE_20BEAM_5MODE_20_3d_20ADJUST().exec();
        _displayName("PREPARE SEPARATION BUMPS COLLAPSE");new PREPARE_20SEPARATION_20BUMPS_20COLLAPSE().exec();
        _displayName("DRIVE COLLISIONS BP FOR PC's AND COLLIMATORS");new DRIVE_20COLLISIONS_20BP_20TO_20COLLIDE_20HW().exec();
        _displayName("END SUBSEQUENCE BREAK");_break();SEQ.sendLogMessage("End subsequence break", (java.lang.Boolean)null);
    }
}
```

The Sequencer GUIs: Editor of sequences and the Executor (to control and monitor the execution).

PREPARE COLLISIONS

- PREPARE COLLISIONS
 - PREPARE FEEDBACKS FOR PHYSICS
 - DISARM FEEDBACKS
 - SWITCH ORBIT AND ENERGY FB OFF
 - SET QFB OFF
 - DISABLE RT TRIMS
 - ENSURE START_COLLISIONS TABLE LOADED
 - MOVE STATE/BEAM_MODE = ADJUST
 - MOVE TO STATE=ADJUST
 - SET BEAM MODE=ADJUST
 - PREPARE SEPARATION BUMPS COLLAPSE
 - INCORPORATE 1.5M TRIMS INTO COLLISIONS
 - INCORPORATE RF FREQ 1.5M TRIMS INTO COLLISIONS
 - REGENERATE ACTUAL END BP FOR PHYSICS
 - MAKE LHC.USER.COLLISIONS RESIDENT
 - LOAD COLLISION FUNC FOR ART GAINS AND PHASE SUI

TASK "INCORPORATE 1.5M TRIMS INTO COLLISIONS" (10)

Arguments:

- SRC_USER (VALUE)
 - type name: java.lang.String
 - description: Accelerator user mapped to the source beam process
 - type enum: JAPC_SELECTOR
 - value: LHC.USER.SQUEEZE-END
- DST_USER (VALUE)
 - type name: java.lang.String
 - description: Accelerator user mapped to the destination beam process

Sequencer Editor

ALL RF TASKS TEST DIALOG TEST CHANGING TIME

- ALL RF TASKS
 - LOAD RAMP SETTINGS IN ALL RF FGC
 - SWITCH RF POWER TO STANDBY
 - SWITCH RF OFF
 - TRIM ALL CAVITY Q TO 20000
 - TRIM ALL CAVITY Q TO 60000
 - TRIM RF TOTAL VOLTAGE AND CAVITY Q FOR
 - TRIM RF TOTAL VOLTAGE AND CAVITY Q FOR
 - ARM LONGITUDINAL BLOW-UP
 - B1: RESET LONGITUDINAL BLOW-UP
 - B2: RESET LONGITUDINAL BLOW-UP
 - LOAD RF BLOW_UP SETTINGS
 - B1: ARM LONGITUDINAL BLOW-UP
 - B1: ENABLE LONG BLOW-UP FEEDBACK
 - B1: SET LONG BLOW-UP PAYLOAD 23
 - B2: ARM LONGITUDINAL BLOW-UP

Run Suspend Step Skip Stop

PREPARED

Console Details Result

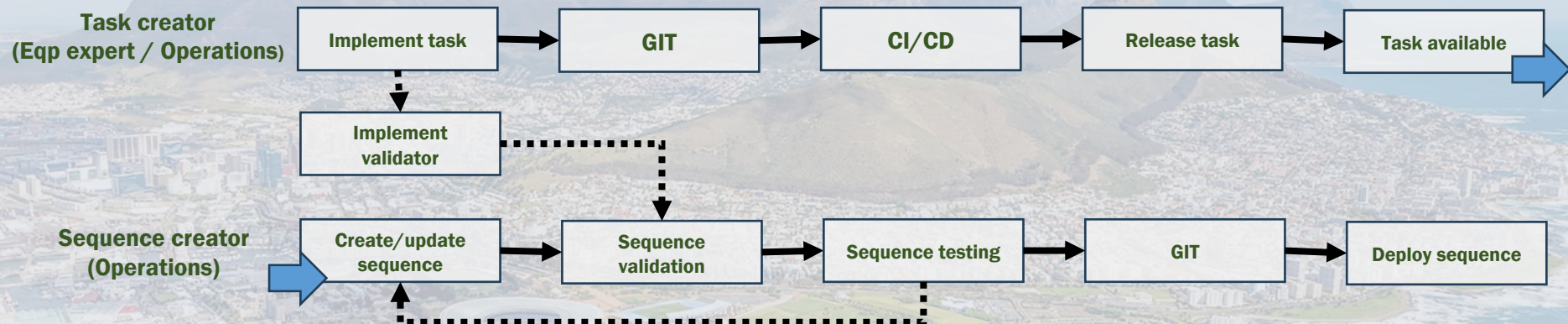
Server logs

Sequence prepared : SequenceId = TEST CHANGING TIME TASKS@2

Sequencer Executor

User workflow

Two distinct stages when using the system:
implementation of a task, and assembly of the sequence.



The tasks and sequences can be created by any user: equipment experts, operations, controls engineers, others. Once available in system repository, both entities can be used and require no detailed knowledge about the implementation details.

State of adoption

The Sequencer is **used by several CERN machines**, primarily in LHC, as well as across the injectors (SPS, PS, PSB, LEIR) and during HW Commissioning campaigns (HWC).

- For HWC fully programmatic approach is used (Sequencer via API access)
- The system is used in a variety of scenarios:
 - ✓ Hardware and machine commissioning
 - ✓ Machine Development specific procedures
 - ✓ Complete operational stages, e.g. RAMP, SQUEEZE in LHC
 - ✓ and more...
- The type of tasks usually falls into one of the categories: check/wait, set/ensure e.g.
 - “check if all Power Converters are ready” → call HW status property*
 - “ensure Power Converters ready” → check if PCs are ON, if not, set ON and wait for confirmation*

The portfolio of tasks and sequences is growing close to 2000 (sub) sequences for LHC; over 350 task types; LHC nominal sequence → close to 2100 tasks

Lessons learned

What works well

- **Modular** approach with **configurable** entities such as: Task/Sub-Sequences/Sequences
 - Enable re-use of entities, limiting copy & paste proliferation of task and sequences
 - Sub-sequences simplify workflow de-composition, from small HW-oriented procedures to rich operational sequences
- Separation of Implementation (tasks), from Edition and Execution of sequences
- With hindsight – direct use of Java (**no DSL or scripting**)

and... what is missing to empower the users

- Sequences with loops, conditions, parallelism of sub-sequences
- Tasks to return results or depend on each-other → *a double-edge sword (rapid growth of complexity)*
- Complete API for programmatic interactions / embedding to other systems
- (possibly) Python task, and more...

The outlook

In the **context of efficiency**, and the objective to further automate and shorten the turn-around times of CERN machines, the Sequencer acts as a **building block**.

How to **automatically fill** LHC?
How to **automatically recover** from HW/SW issues?
... and many more specific questions to answer

In this context, we plan to:

*Invest more into it, re-think GUI and **modernize**, addressing known technological risks, to consider **opening it**, when justified by the **external interest**, **rearchitected**.*

The work to begin in 2024++ horizon, driven by the Efficiency working group@CERN.

How **other labs could benefit** from the Sequencer?

Sequencer is the **main automation software solution** for CERN accelerators operations.

It is a well-established system and the problems it addresses are **not CERN specific**.

Planned **evolution and extensions** aim to further increase the efficiency by helping to **shorten the turn-around times**.

We are embarking on making it more modular, extended, modernized.
An **opportunity to collaborate** and let other labs profit.