Elettra
Sincrotrone
Trieste

A FRAMEWORK FOR HIGH LEVEL MACHINE AUTOMATION BASED ON BEHAVIOR TREES

G. Gaio
on behalf the Sequencer TEAM

www.elettra.eu

Elettra Sincrotrone Trieste

Introduction

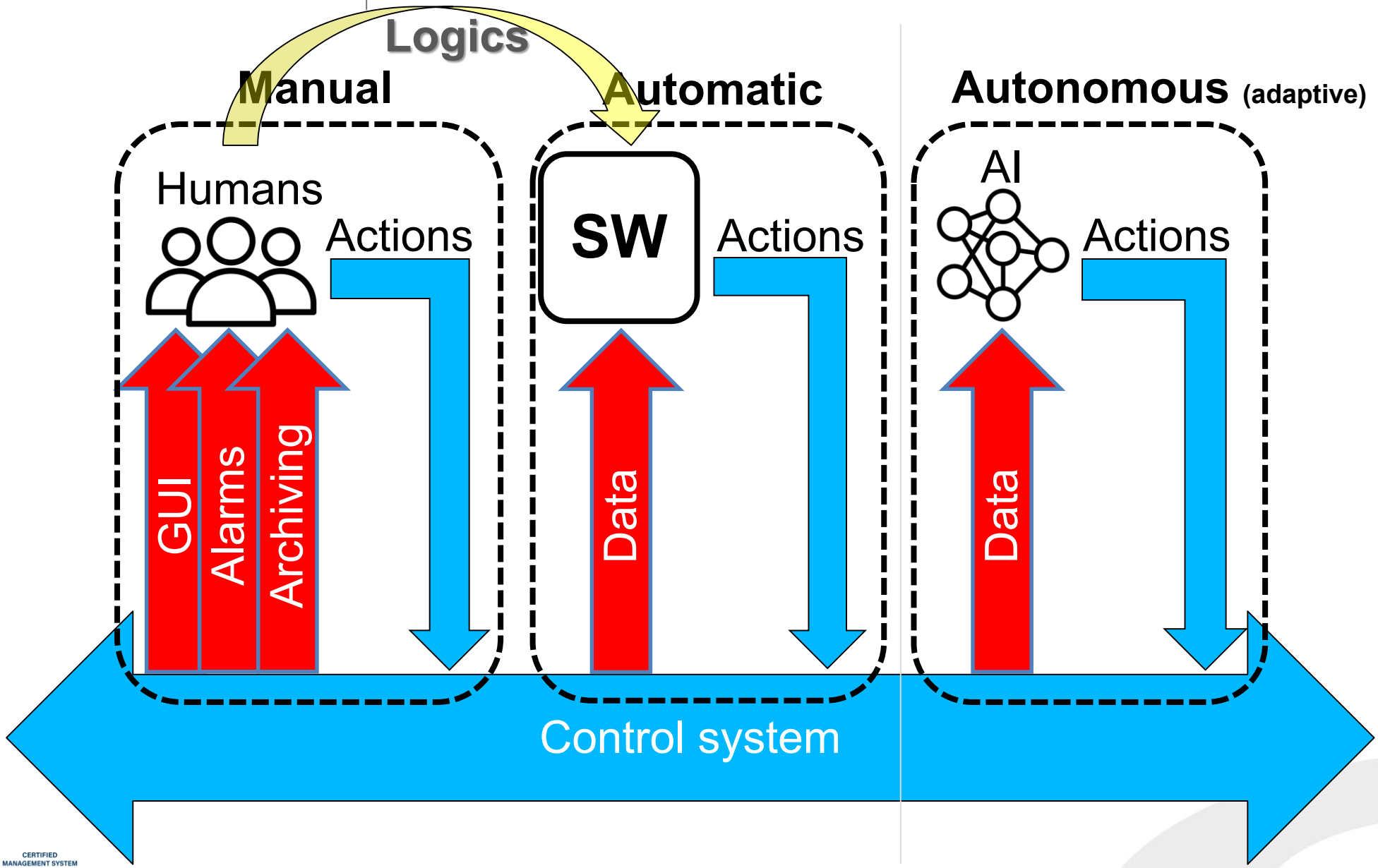
✓ Automation frameworks:

- *The Colliding Beam Sequencer (FNAL - PAC89)*
- *Automating ELETTRA Operation with One Button Machine (Elettra - IPAC97)*
- *The RIHC Sequencer (BNL - PAC01)*
- *A sequencer for LHC ERA (CERN - ICALEPCS09)*
- *Automated operation of ITER using behavior tree semantics (ITER – this conference **WEPV006**)*
- ……
- EPICS sequencer
- …….

SOLEIL, DESY....and many many others

Sorry if I haven't mentioned your work!

Toward full automation



Automation metric

- ✓ For a user facility like a synchrotron **full automation** means:
  - Recover from a beamdump, reinject and give a stable beam to the users without any human intervention
- ✓ A metric to measure human intervention is:
  - ***Quantum of human-computer interaction (Qhci)****:* one click on a keyboard/mouse in the control room during user and machine tuning shifts

$$\mathbf{AL(U)} = \Sigma\,[Q_{hci(U)}]_{\text{per day}}$$
$$\mathbf{AL(T)} = \Sigma\,[Q_{hci(T)}]_{\text{per day}}$$

Automation **L**evel = AL(U) + AL(T)

AL = 0 autonomous machine

Scripting language

- ✓ Move logics from GUIs and stand alone scripts to server side applications (tango servers)
- ✓ Speed up the knowledge transfer by involving more people
- ✓ Minimize bugs by privileging configuration instead of programming
- ✓ **Home-made scripting language** (based on C++ Boost.Spirit parser)
  - Implement IF/ELSE (conditional ternary operator "? :") and the "harmful" GOTO statements

Language syntax

stepN;[expression];[step description];[error message];[timeout ms.];[catch exceptions]

Sequence example (reset and turn ON a power supply):

step1;read(sr/ps/ch_s1.1/State)==FAULT **?** command(sr/ps/ch_s1.1/Reset) && **goto(2) : goto(3)**;Reset PS;Error resetting PS;3000
step2;read(sr/ps/ch_s1.1/State) != OFF **?** sleep(1) && **goto(2) : goto(3)**;Waiting OFF state;Timeout waiting OFF state;6000
step3;read(sr/ps/ch_s1.1/State) != ON **?** command(sr/ps/ch_s1.1/On) && **goto(4) : goto(5)**;Turn PS ON;Error turning ON PS;3000
step4;read(sr/ps/ch_s1.1/State) != ON **?** sleep(1) && **goto(4)** : **goto(5)**;Waiting ON state;Timeout waiting ON state;6000

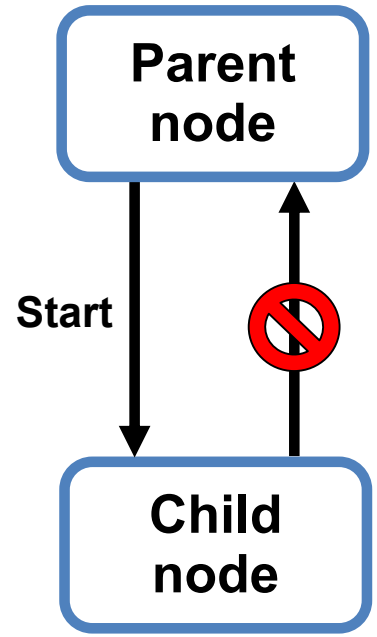
Description:

step1; If PS is in FAULT state then Reset and **go to step2**, otherwise **go to step3**
step2; if PS is not OFF then sleep one sec and check again (max 6 sec.), otherwise **goto step3**
step3; if PS is not ON turn PS ON and **go to step4**, otherwise **go to step5** (exit)
step4;if PS is not ON then sleep one sec and check again (max 6 sec.), otherwise **go to step5** (exit)

Behavior Trees

✓ **Used by AI in video games (Unreal Engine), UAV…**

Direct Acyclic Graph



Node states
SUCCESS
RUNNING
FAULT

The **parent node starts** the execution of the **child node** and **waits** it to complete the task. If the **child node** goes in **FAULT** state then also the **parent node** will end in **FAULT** state.

Structural nodes:

sequence, composer, fallback, selector, decorator

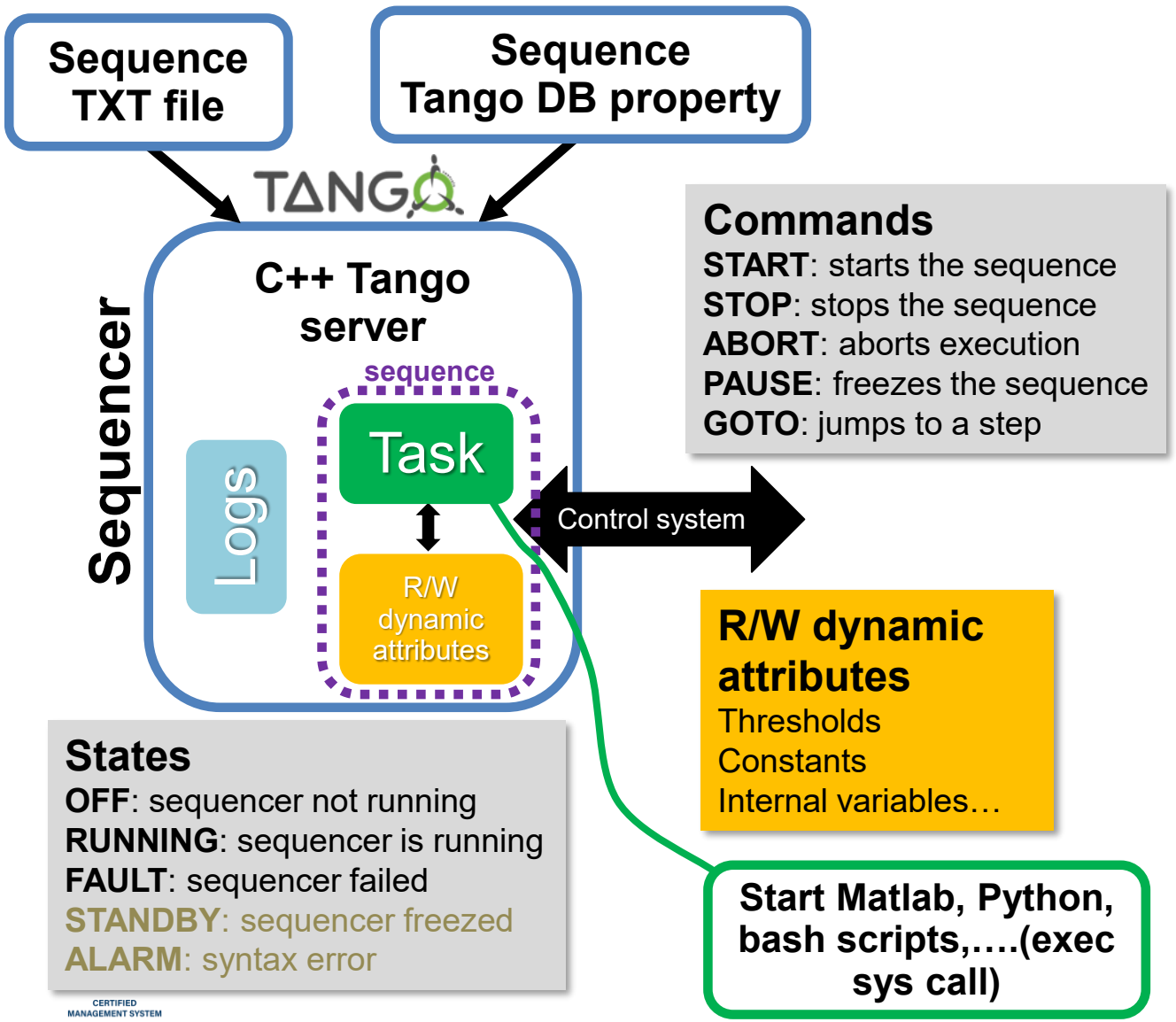
A node executes **a real action** or implements **the structure of the BT:**

- start child nodes in series
- start child nodes in parallel
- invert the returning state of a child node
- Execute at least one child node from a multitude
- Conditional execution between two child nodes
- …….

All nodes are implemented with the scripting language

Tree node: the Sequencer

The Sequencer is a Tango device implementing a task

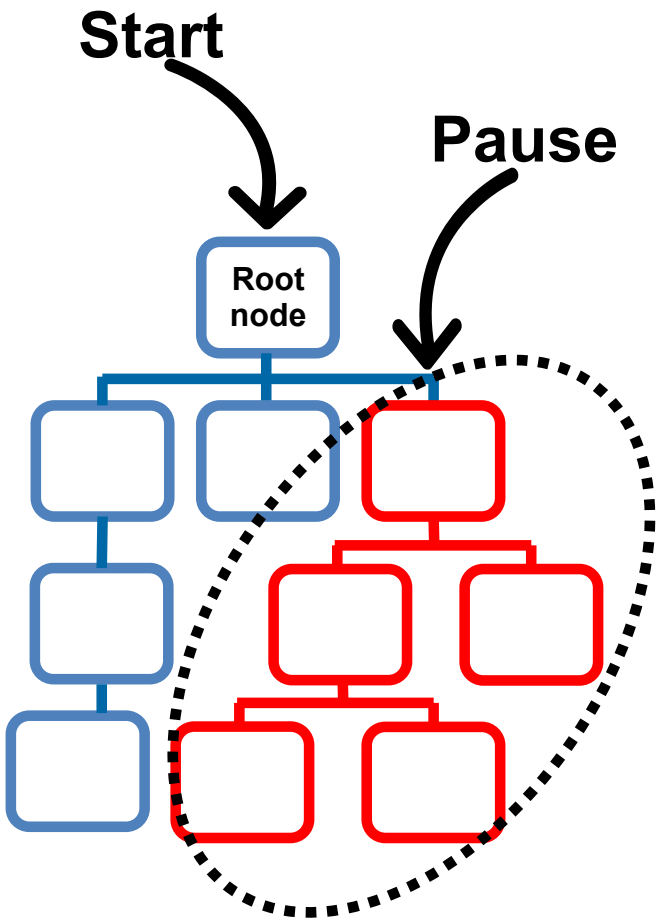


Commands
START: starts the sequence
STOP: stops the sequence
ABORT: aborts execution
PAUSE: freezes the sequence
GOTO: jumps to a step

R/W dynamic attributes
Thresholds
Constants
Internal variables…

States
OFF: sequencer not running
RUNNING: sequencer is running
FAULT: sequencer failed
STANDBY: sequencer freezed
ALARM: syntax error

Start Matlab, Python, bash scripts,….(exec sys call)



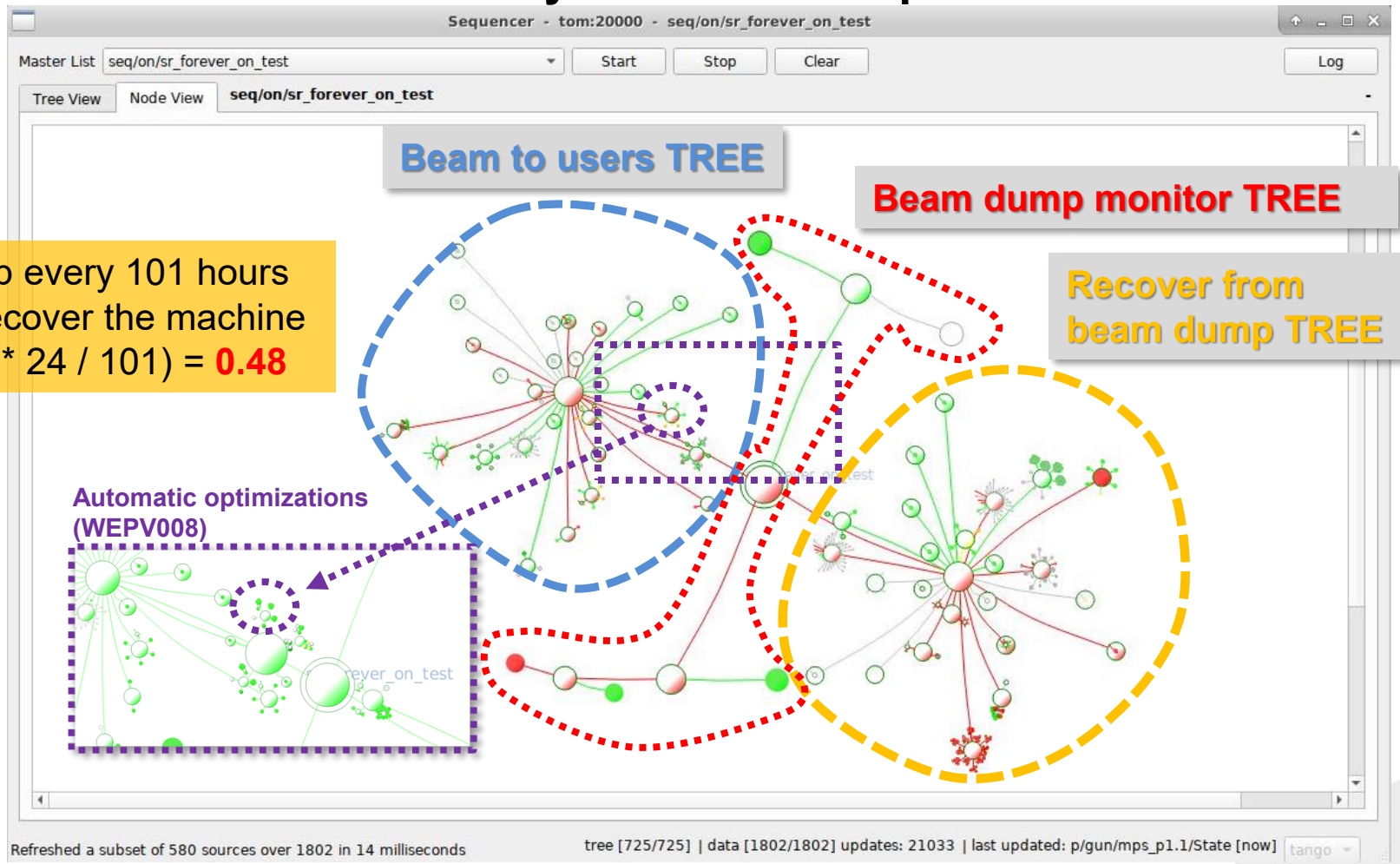
Can propagate **Stop**, **Abort** and **Pause** command through a sub-tree

Sequencer GUI (Node view)

All grafical interfaces (Qt-Cumbia) are automatically generated by getting information from the Tango Database and by dynamically exploring the Behavior Tree.

Elettra synchrotron autopilot

1 beamdump every 101 hours
2 clicks to recover the machine
AL(U) >= (2 * 24 / 101) = **0.48**



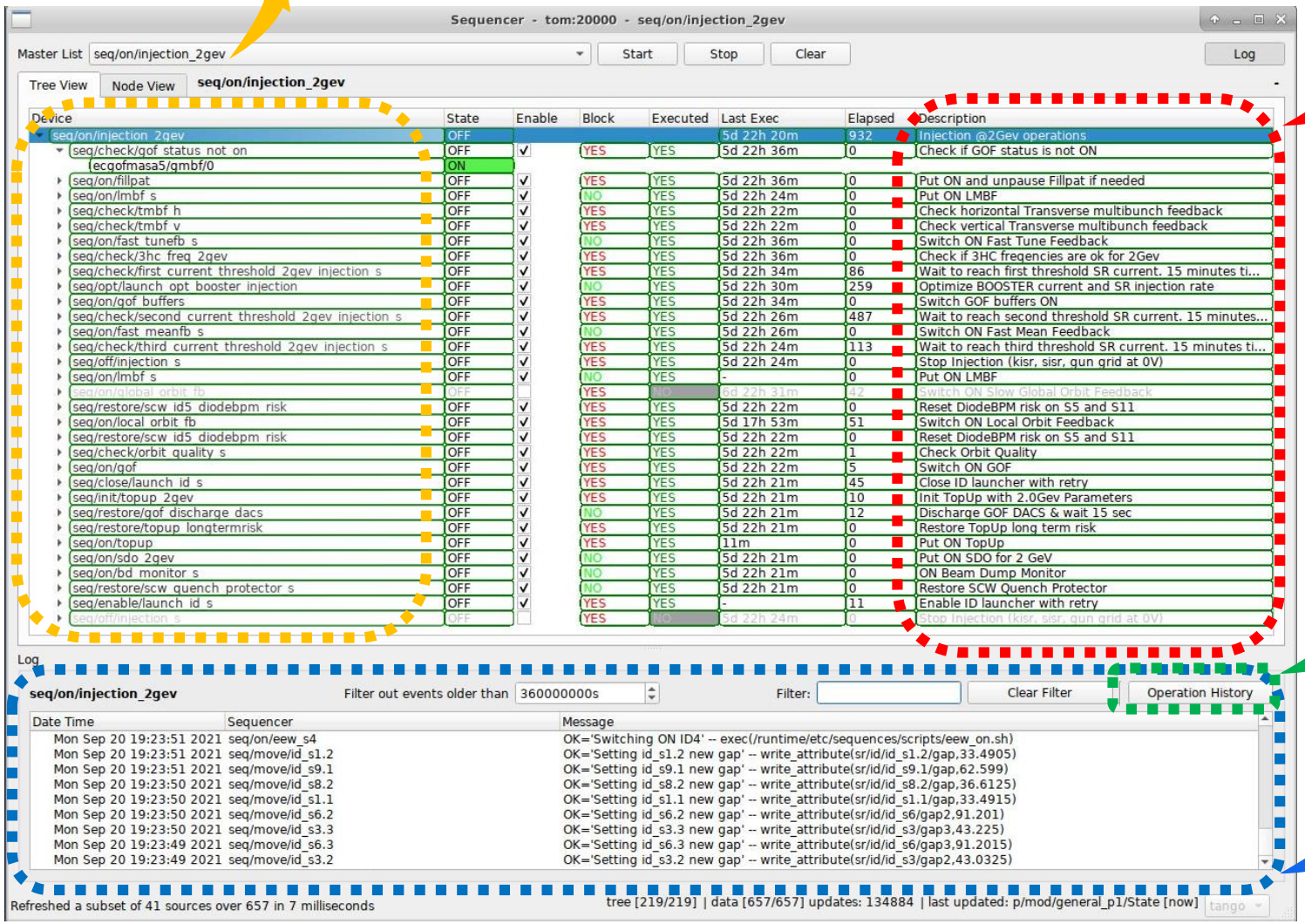
725 nodes **1802 attributes**

Giulio Gaio, giulio.gaio@elettra.eu

Sequencer GUI (Tree View)

Elettra Sincrotrone Trieste

Sequencer that refills the storage ring till giving stable beam to the users



Sequence description

Historical logging

RT-logging based on events

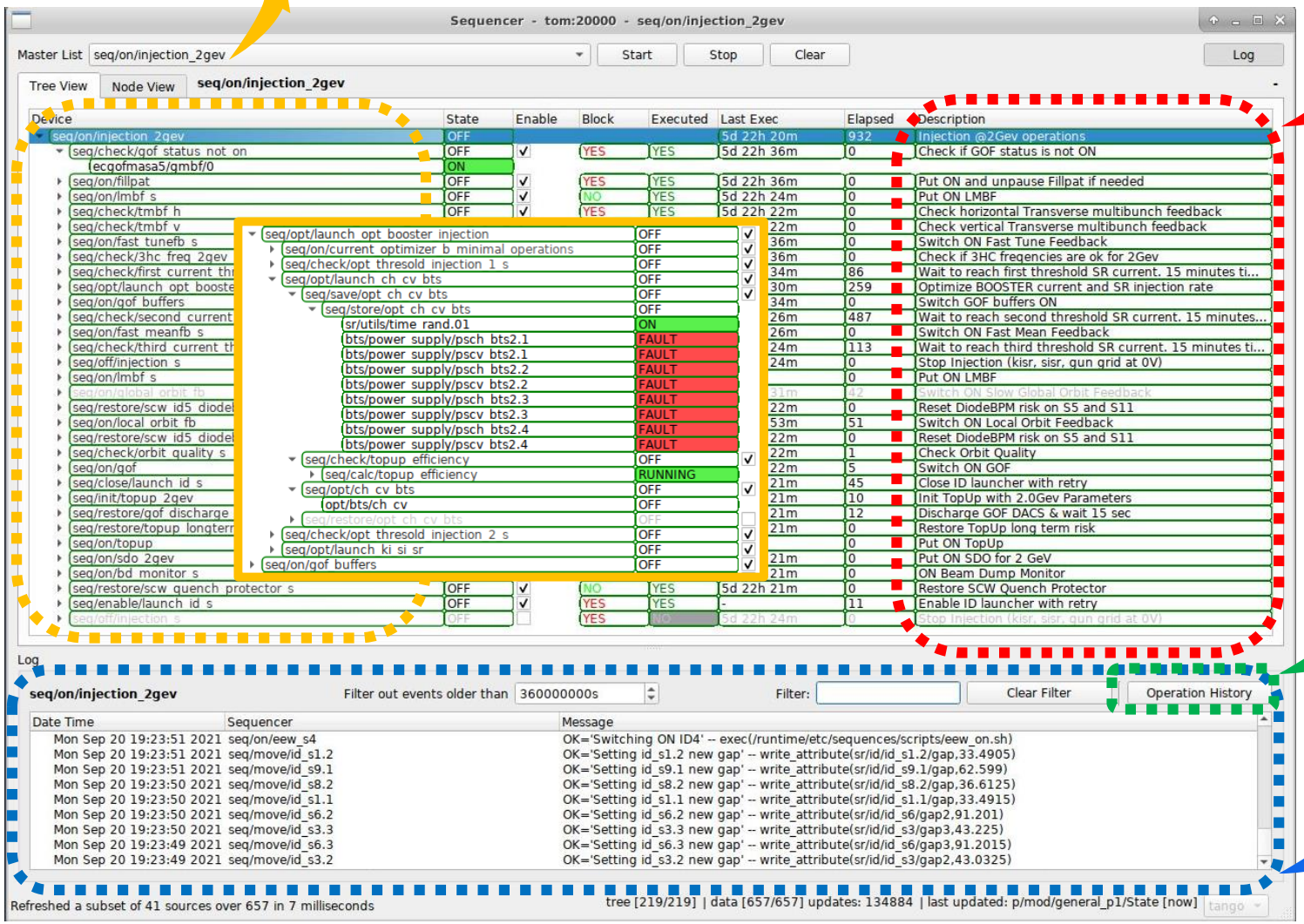
Giulio Gaio, giulio.gaio@elettra.eu

20/10/2021

Sequencer GUI (Tree View)

Elettra Sincrotrone Trieste

Sequencer that refills the storage ring till giving stable beam to the users



Sequence description

Historical logging

RT-logging based on events

Giulio Gaio, giulio.gaio@elettra.eu

Conclusions

- ✓ 995 sequencers in FERMI, 958 sequencers in Elettra
- ✓ 8 people have been involved (Controls + Operators + Physicists), now 1 FTE
- ✓ In operation in Elettra since 2019, almost all high level operations in control room driven by sequencers
- ✓ Introduced in FERMI in 2021
- ✓ Developer doesn't have to care of GUIs, logging, documentation…all out of the box
- ✓ "Framework" based on one C++ Tango server, two Qt-Cumbia panels and a sequence template that eases the implementation of the BT structure
- ✓ Short term to-do list:
  - Versioning of an entire BT logic (md5)
  - Log analysis to detect anomalies in terms of execution time and fault rate

Elettra
Sincrotrone
Trieste

Thank you!

P. Cinquegrana, G. Gaio, S. Krecic, G. Scalamera, G. Strangolino, F. Tripaldi, M. Trovo', L. Zambon