Settings Management: status and plans

Michi Hostettler

on behalf of the Settings Management Working Group / EPA WP 6

and with great thanks to all OP teams, the LSA team,
and the EPA project core team

towards efficient settings management

"efficient settings management" is a long term process

Settings Management Working Group (OP + LSA team + equipment groups)

Objectives:

- Homogenize as far as possible the operational procedures and primciples concerning settings management for accelerators and facilities
- Get toword more physics parameters oriented operation
- Get rid of 'cultural' differences of the operations teams, deal with the technical differences and specific constraints of the different accelerators
- Solve issues (common or specific) and improve the usage of LSA-INCA in all facilities

other EPA work packages with setting management requirements

WP1
Dynamic Beam Scheduling
Lead: F. Irannejad (BE-CSS)

WP3
Automated Parameter Control & Optimisation
Lead: M. Schenk (BE-CSS)

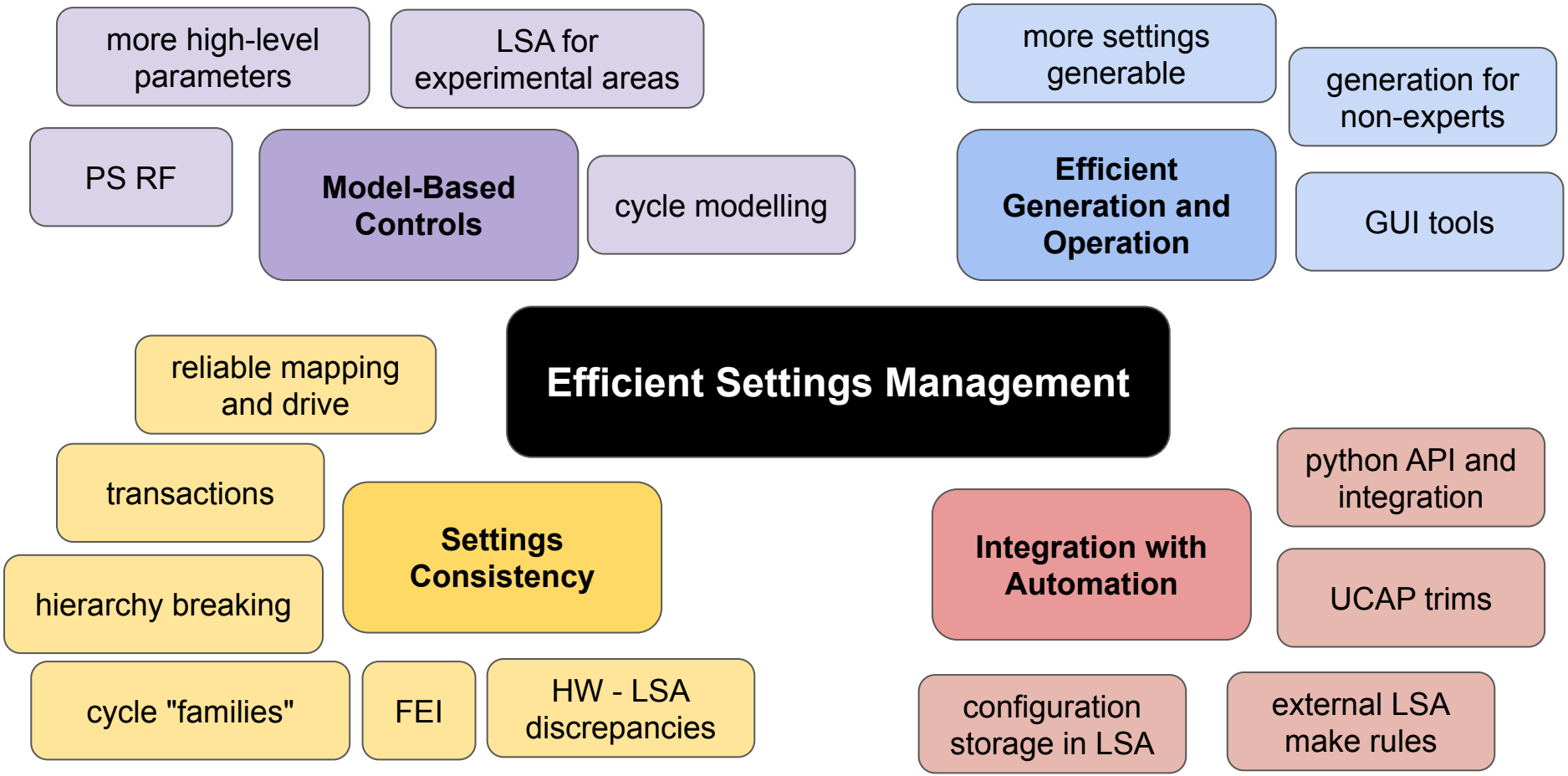
WP8
Automating Equipment
Lead: K. Papastergiou, F. Velotti (SY-ABT)

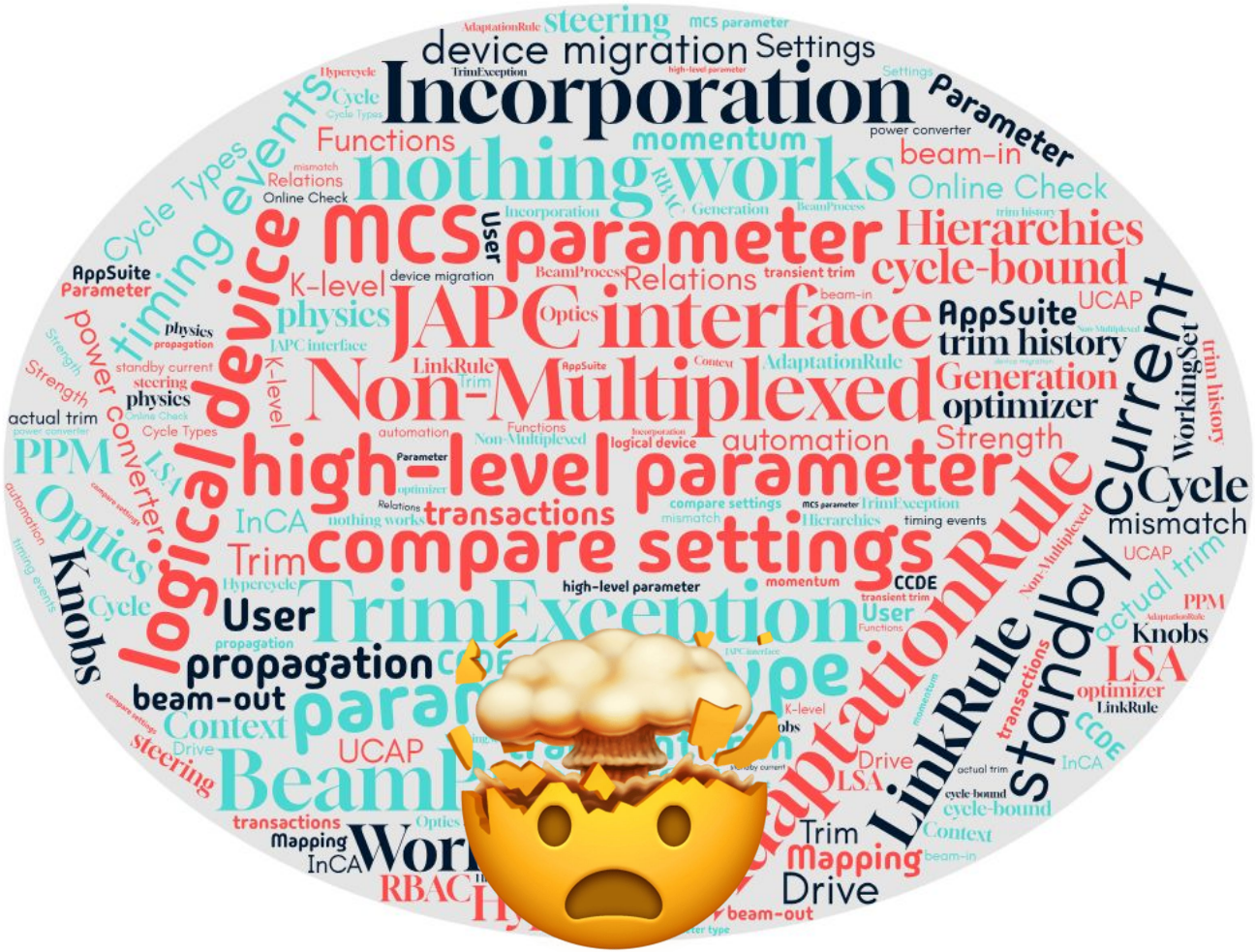
WP2
Automated LHC Filling
Lead: G. Trad, A. Huschauer (BE-OP)

improvements and solutions for existing operational use cases & problems

requirements on settings management for future novel operational approaches

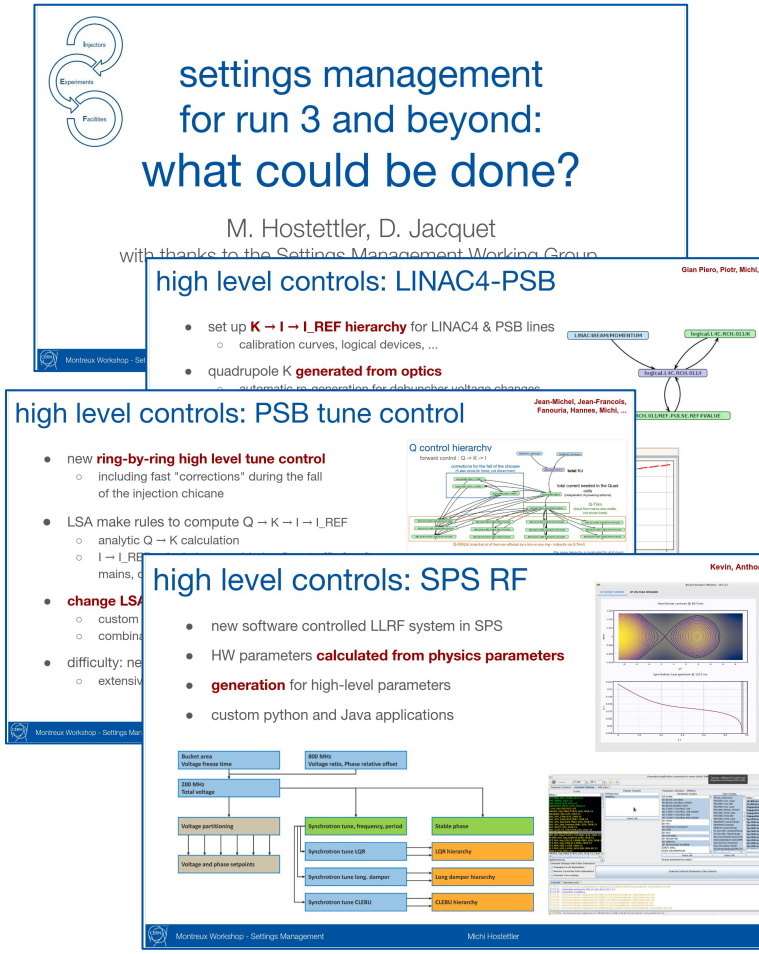
EPA WP6 - efficient settings management


more high-level parameters
LSA for experimental areas
PS RF
Model-Based Controls
cycle modelling
more settings generable
generation for non-experts
Efficient Generation and Operation
GUI tools
Efficient Settings Management
reliable mapping and drive
transactions
hierarchy breaking
Settings Consistency
cycle "families"
FEI
HW - LSA discrepancies
python API and integration
Integration with Automation
UCAP trims
configuration storage in LSA
external LSA make rules


steering
device migration
Settings
Incorporation
Parameter
Functions
momentum
nothing
works
beam-in
Online Check
Cycle Types
timing events
MCS
parameter
Hierarchies
cycle-bound
K-level
physics
JAPC
interface
AppSuite
trim history
Non-Multiplexed
Generation
optimizer
high-level parameter
automation
Strength
PPM
Cycle
mismatch
logical device
compare settings
transactions
InCA
Trim
TrimException
User
propagation
beam-out
Context
UCAP
BeamProcess
Drive
LSA
Knobs
Optics
LinkRule
standby current
AdaptationRule
Mapping
RBAC
Relations
actual trim
power converter

where are we?

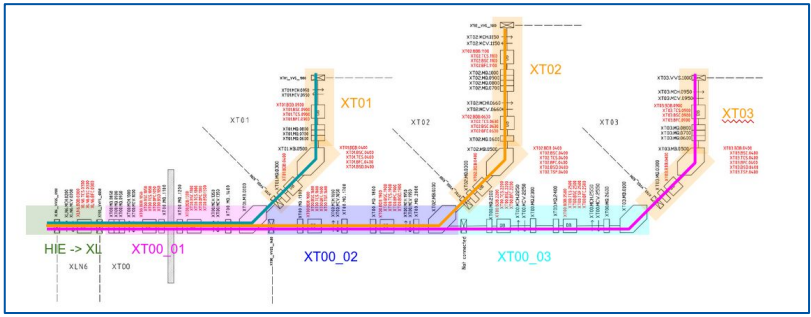
- significant effort on **high-level controls for injectors during LS2** (in the view of LIU)
  - driven by the joint CO + OP + Equip Settings Management Working Group
  - mostly successful: in operation since 2021
  - next: experimental areas!
- optics + steering for all machines
- generation - could still be improved
- settings (in)consistency - being addressed
- more and more integration outside LSA
  - measure & correct, optimization, …

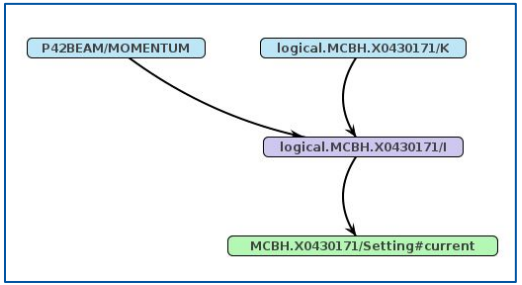


LSA for experimental areas

- CESAR renovation + NA consolidation
  - unify settings management
  - model based controls
- non-PPM, partially cycle bound
- **model: stand-alone beam processes**
  - like LHC, but **no hyper-cycles**
  - possibility to map beam processes per particle transfer / beam line
- first implementations:
  - ISOLDE - HIE part
  - P42 line - first test in November

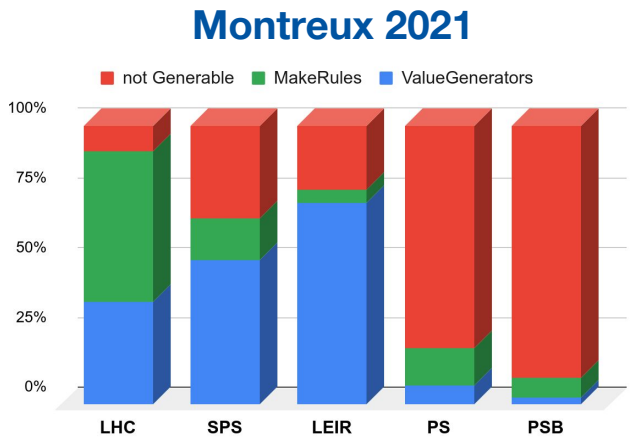
Beam Process	Particle Transfer
H2A.NA61.910	NA_H2Transfer
H4D.HERD.002	NA_H4Transfer
H8A.SBA.061	NA_H8Transfer
M2A.SBA.040	NA_M2Transfer
P0A.SBA.000	NA_P0SurveyTransfer
P42A.SBA.040	NA_P42Transfer
	NA_H6Transfer
	NA_K12Transfer

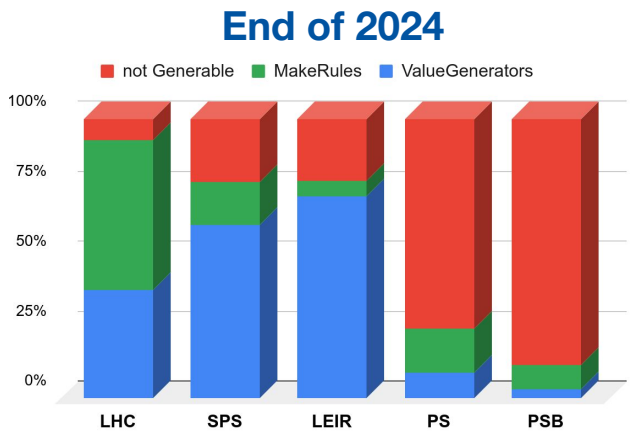




generation

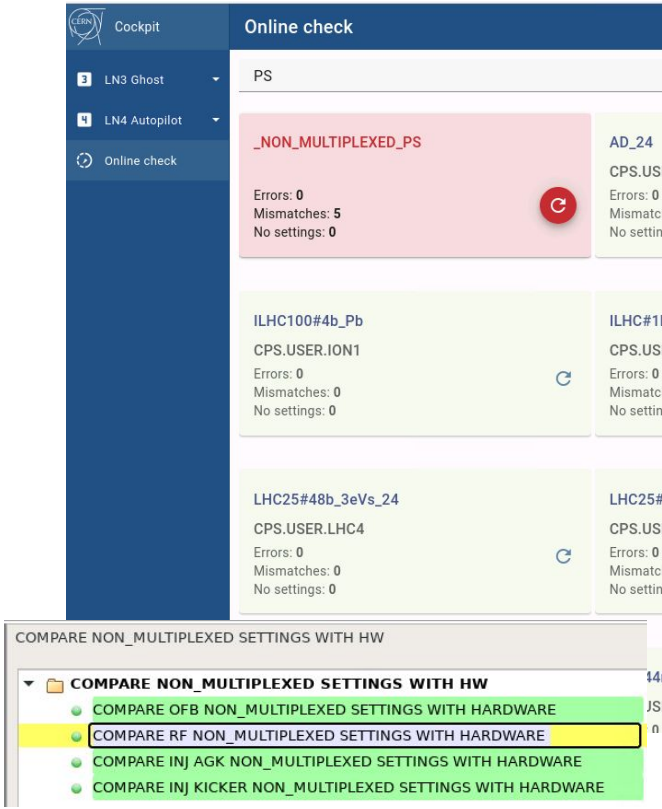
- generation: preferred way to initialize settings
  - largely automated
  - code / logic in a central place
  - avoid losing knowledge / single point of failure
- PS and PSB still need some of work …
  - cycle model also fully ideal
  - work planned in LS3
- new: **bulk generation**
  - N parameters * M contexts





(in)consistencies

- at **different levels**:
  - missing settings / discontinuities (new cycles / BPs)
  - broken LSA hierarchies
  - LSA - HW inconsistencies
- predominant cause of **settings related downtime**
  - e.g. LHC - LLRF settings issue during MD3
    "reloading" of a tag took RF out > 1 day
- LSA - HW: new **continuous checks** in place
  - injectors: web application provided by LSA team
  - LHC: sequencer tasks during preparation
  - optional tolerance (HW read-back issues)
- not solved - but mitigated!



automation integration

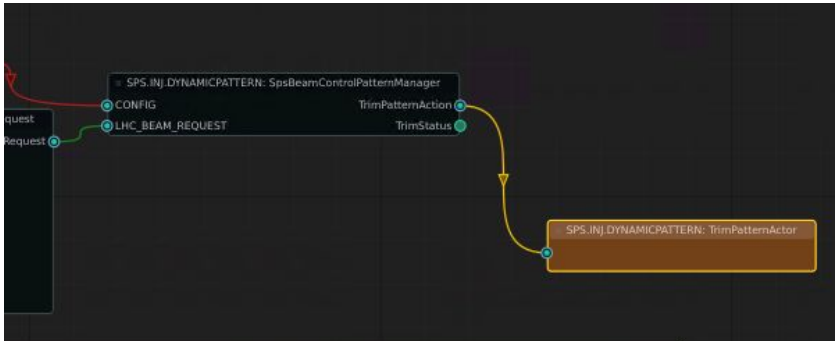
- "transient" trims
  - problem: automated trims fill up the trim history
    - difficult tracking, performance, …
  - transient trims: cleaned up after 14 days
    - do we need more flexible time to live?
  - same application can produce transient and permanent trims

Time	Descr
22-11-2024 15:07:43	LumiServer trim [ManualSeparationLev
22-11-2024 15:12:05	LumiServer trim
22-11-2024 15:16:48	LumiServer trim
22-11-2024 15:18:31	LumiServer trim
22-11-2024 15:18:55	LumiServer trim [ManualSeparationLev
22-11-2024 15:21:23	LumiServer trim [OptimizationAutopilot
22-11-2024 15:22:04	LumiServer trim [OptimizationAutopilot
22-11-2024 15:22:46	LumiServer trim
22-11-2024 15:29:06	LumiServer trim
22-11-2024 15:29:48	LumiServer trim [OptimizationAutopilot
22-11-2024 15:30:29	LumiServer trim [OptimizationAutopilot

permanent

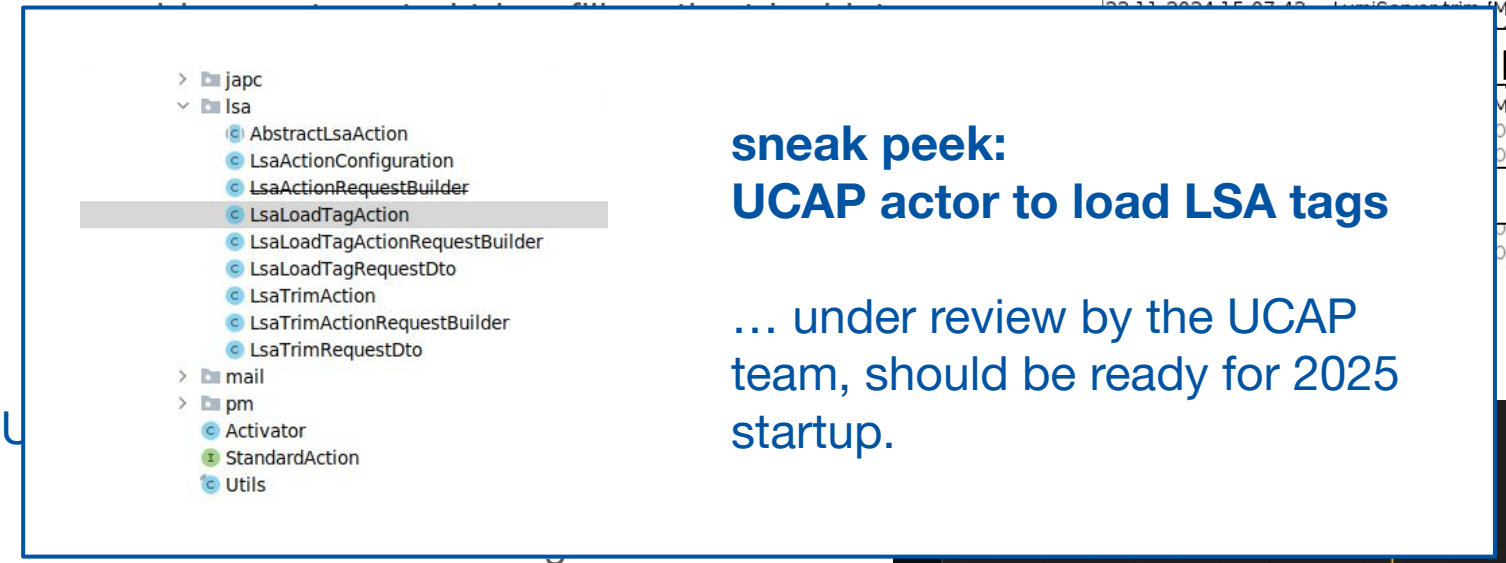
transient

- UCAP LSA trim actor
  - now in ucap-actions-standard
  - trim from UCAP - avoiding JAPC API
  - used by GeOFF, optimizers, …
  - possibility for transient trims
  - traceability (actor name in history)

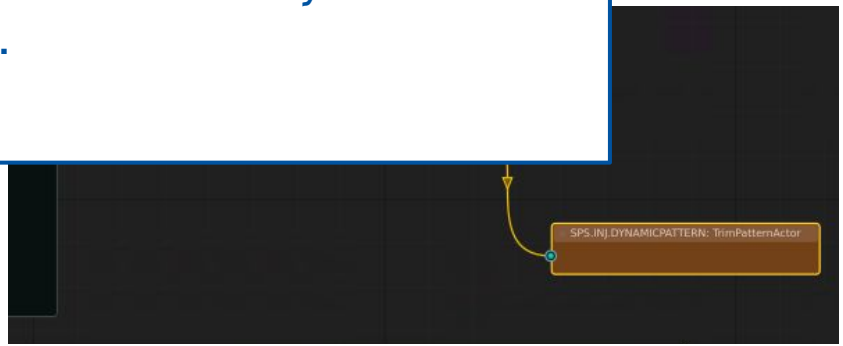


automation integration

- "transient" trims

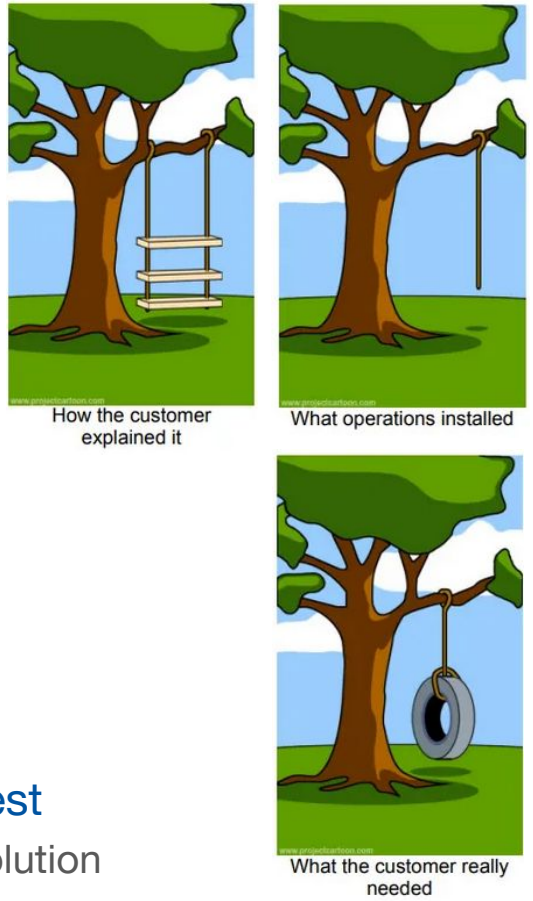


- U
  - used by GeOFF, optimizers, …
  - possibility for transient trims
  - traceability (actor name in history)



what do we still need?

- we are fine in the short term, but …
- novel controls and optimization concepts (ML, …)
  - softening the border between online and offline
  - Java-only is too rigid
  - one setting per cycle may be too rigid
- improved modelling for exp. areas, PS, …
- HL-LHC integration
- settings management is more than device/properties
  - modelling, domain-driven objects, …
- the (apparently) simplest solution may or may not be the best
  - the truth lies between the quick hack and the most generic solution



custom logic in LSA - beyond Java

- custom code in LSA: **make rules, value generators**, …
  - runs online (mapped/resident context) or offline
  - currently: monolithic, Java only
- python libraries superior in certain fields
  - machine learning
  - numerics & scientific computing

➜ Machine Learning Platform - promising solution
  - "stand-alone" model deployment
  - remote call from LSA code
  - planned for next-generation PS PFW control
  - investigated for post-LS3 PS LLRF control

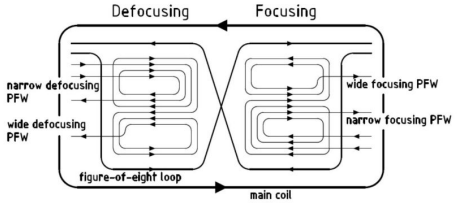
- LSA core will remain Java native

```
RealMatrix K0;
RealMatrix KH1, KH2;
RealMatrix SH1, SH2;

int iterationCount = 0;
RealMatrix P_ = initialP;

try {
    KH1 = B.transpose().multiply(P_).multiply(B);
    KH2 = B.transpose().multiply(P_).multiply(A);
    K = inverse(R.add(KH1)).multiply(KH2);
} catch (NullPointerException ex) {
    System.out.println(P_);
    System.out.println(A);
}

SH1 = A.transpose().multiply(P_).multiply(A);
  SH2 = K.transpose().multiply(B.transpose()).multiply(P_).
SH2 = A.transpose().multiply(P_).multiply(B).multiply(K);
P_ = (Q.add(SH1)).subtract(SH2);

K0 = K.scalarMultiply(0);

while (K.subtract(K0).getNorm() > EPSILON * K.getNorm()) {
    K0 = K;
```

scipy.linalg.solve_discrete_are

scipy.linalg.solve_discrete_are(a, b, q, r, e=None, s=None,

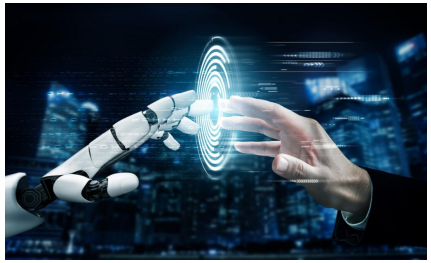
Solves the discrete-time algebraic Riccati equation (DARE).



interaction with automatic controllers

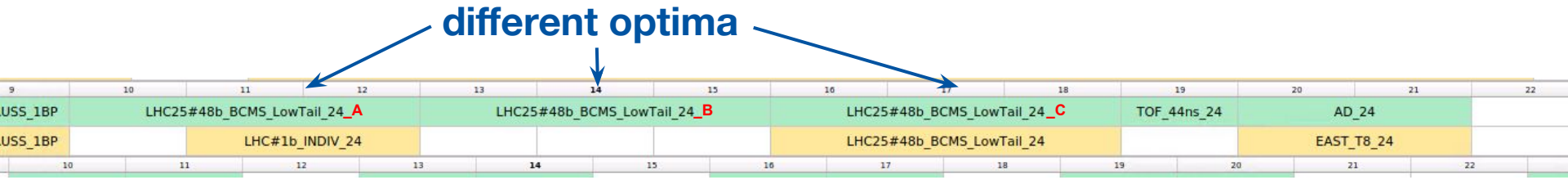
- increasing amount of automatic optimizers / feedbacks
  - ensure continuous optimization
  - reduce OP workload / repetitive tasks
- **OP** (and other optimizers) **need an overview**
  - which optimizers are running?
  - which parameters are touched by which optimizer?
  - how to monitor, interact, stop, and resume optimizers?
  - goes beyond settings management

➜ concept of "parameter ownership"?
  - automatic controllers need to "reserve" parameters
  - warning for OP if trimmed manually



➜ optimizers are there to help us - OP
  - ensure a good human-machine collaboration

per cycle-instance settings



- **different cycle executions** may have different optimal settings
  - depending on intensity, preceding cycles, hysteresis, …
  - e.g. splitting, trajectories, …
- **avoid: trims** / sending settings for every cycle execution
  - overflowing trim history, even with transient trims
  - optimizers and LSA do not have real-time guarantees

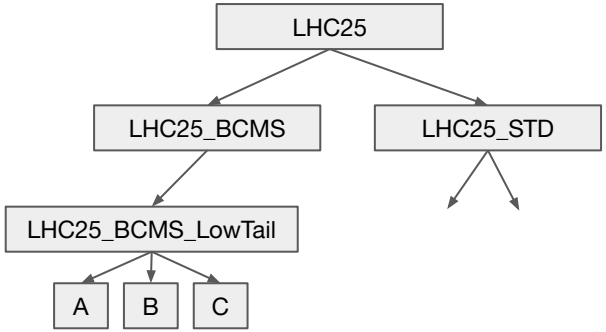
➜ "real" hardware real-time channels (needs HW support)

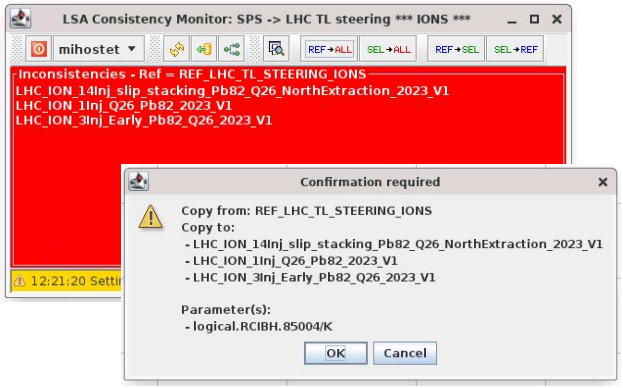
➜ different timing user per execution for different programmed settings?

do we need super-settings?

- scenario: different cycles which are **50-90% similar**
  - cycles for the same destination in injectors
  - SPS-LHC injection steering
  - different users per cycle execution

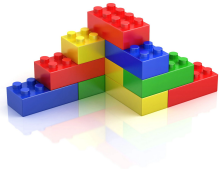
    … if we are going for this solution
- **common settings** should be kept in sync
  - not only initially, but also during operation
- "**super settings**" - propagation between cycles
  - technical challenge … looks solvable
  - operational complexity is the main concern (understand propagation, debugging, …)
- lightweight approach: reference cycles
  - used throughout 2024 for LHC injection steering

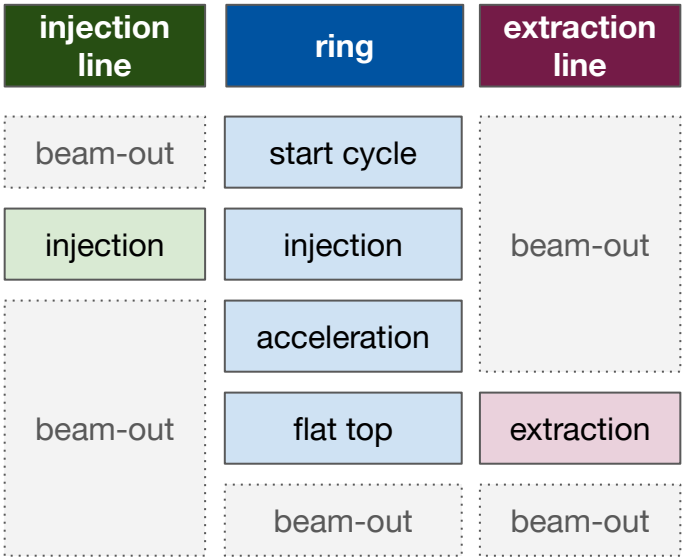




LS3: re-thinking injector cycle modelling

- a LSA **cycle type** can schedule multiple **sequential beam process types**
- beam process types are "**building blocks**" that can be reused (e.g. ramps, plateaus, …)
- **beam-in** and **beam-out** beam processes
- current situation:
  - SPS: full cycle model
  - LEIR: 1 beam-in and 1 beam-out beam process
  - PS / PSB / AD / ELENA: 1 beam process (beam-in only) per cycle and particle transfer
- **full model could be useful for other machines**
  - PS full model under study, including RF processes
  - PSB could at least profit from beam-in/out

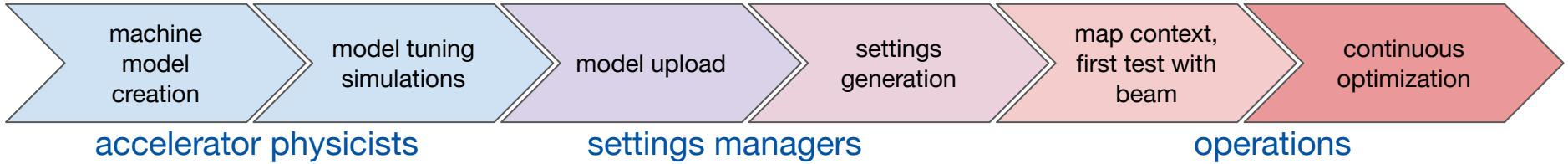




LS3: LHC moving towards the HL-LHC era

- standard transaction system
  - transactions are crucial in LHC to execute synchronized transitions (ramp, …)
  - currently each system has its own transaction protocol
  - goal: use common transaction protocol introduced in injectors during LS2
- new inner triplet assemblies in LSA
  - IT-STRING as a test facility - unique opportunity!
  - beyond magnetic elements: FRAS, QPS, …
- QPS settings management
  - expose QPS settings in a more standard way (FESA classes, interfaces)
  - store settings in LSA (possibly with extra protection)
- XSuite for optics creation and upload

model-based controls, in an ideal world …



- smooth and standardized path from physics models to operations
- models allow to generate initial settings to make a beam pass
- handle what can be handled with physics-backed models
  - pre-correct / feed-forward known effects
- feedback loops and optimizers correct for dynamic effects and imperfections

➜ do not spend beam time on problems that can be solved offline

conclusions

- steady evolution towards model-based controls
  - significant improvement for injectors in LS2
  - experimental areas upcoming
  - settings generation and cycle model being improved
- settings inconsistencies still an issue
  - LSA internal / LSA-HW
  - main cause of settings related downtime
  - mitigations in place
- integration with automation
  - automated trims: transient to avoid overflowing history
  - optimizer management: part of settings management?
  - per cycle-execution settings?