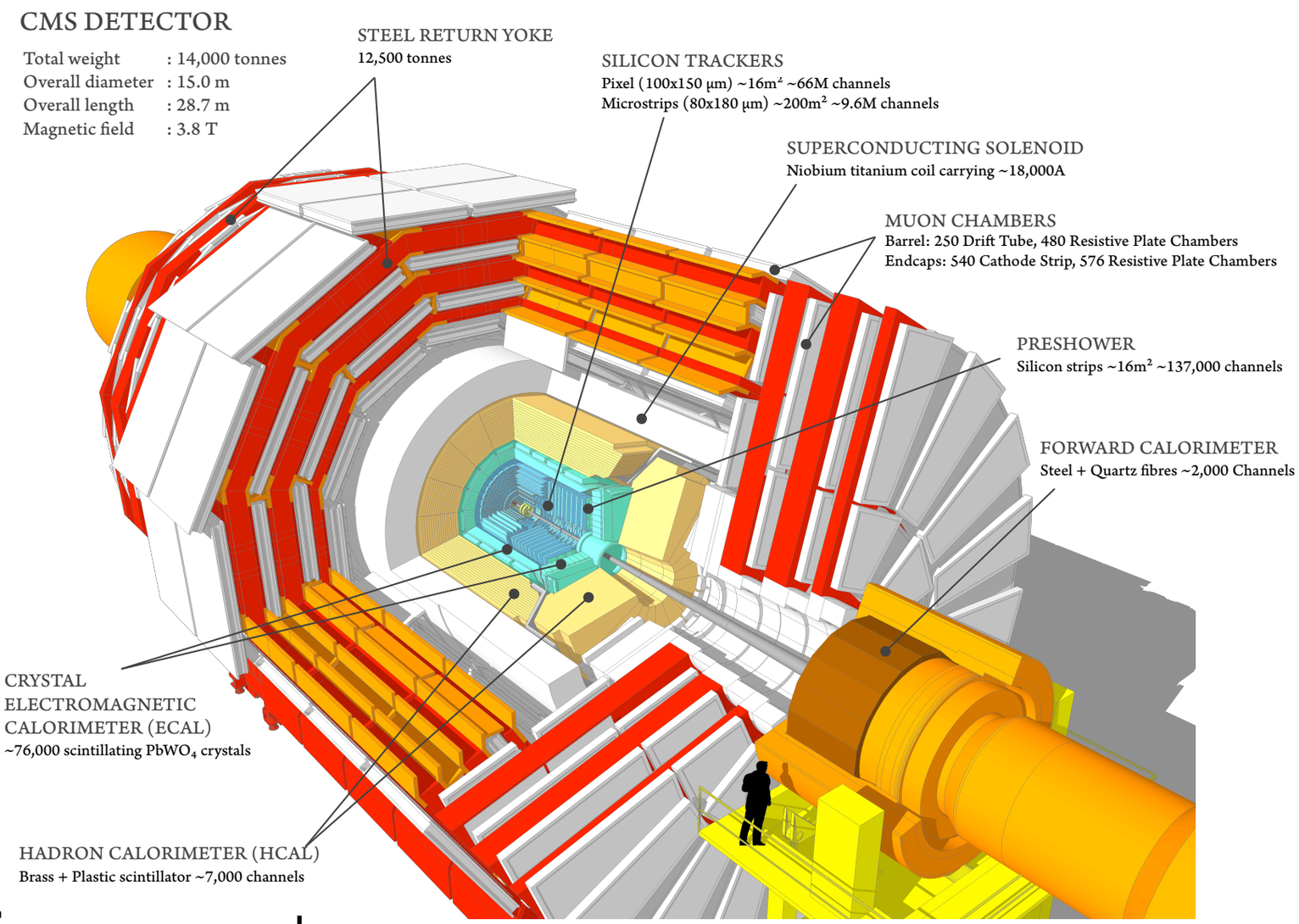
SWATCH: Common software for controlling and monitoring the upgraded Level-1 trigger of the CMS experiment

S. Bologna, G. Codispoti, G. Dirkx, L. Kreczko,
C. Lazaridis, E. Paradas, A. Rose, A. Thea, T. Williams

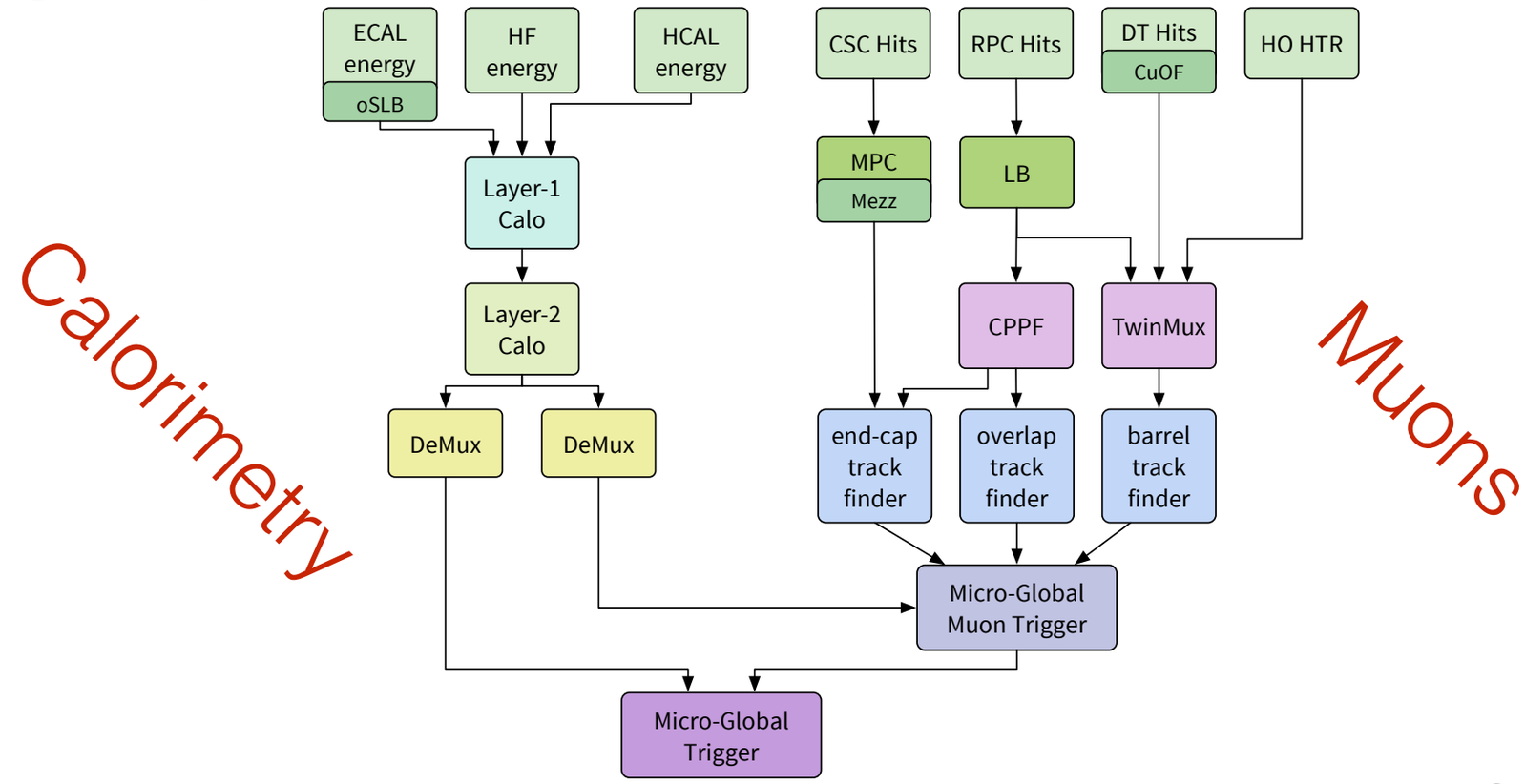
CMS Level-1 trigger system



- Level-1 trigger system:
  - Selects 100 kHz of interesting events from 40MHz pp collisions
  - Fixed latency: Decision within 3.8 us, whilst full-resolution data held in pipeline memories
  - Input: coarse data from calorimeters & muon detectors

Level-1 trigger upgrade

- 2015: LHC restarts after Long Shutdown 1
  - Higher energy & luminosity; trigger efficiency must remain same / improve
  - Trigger upgrade from 2015 (partial) to 2016 (full)
- 2016 trigger system:

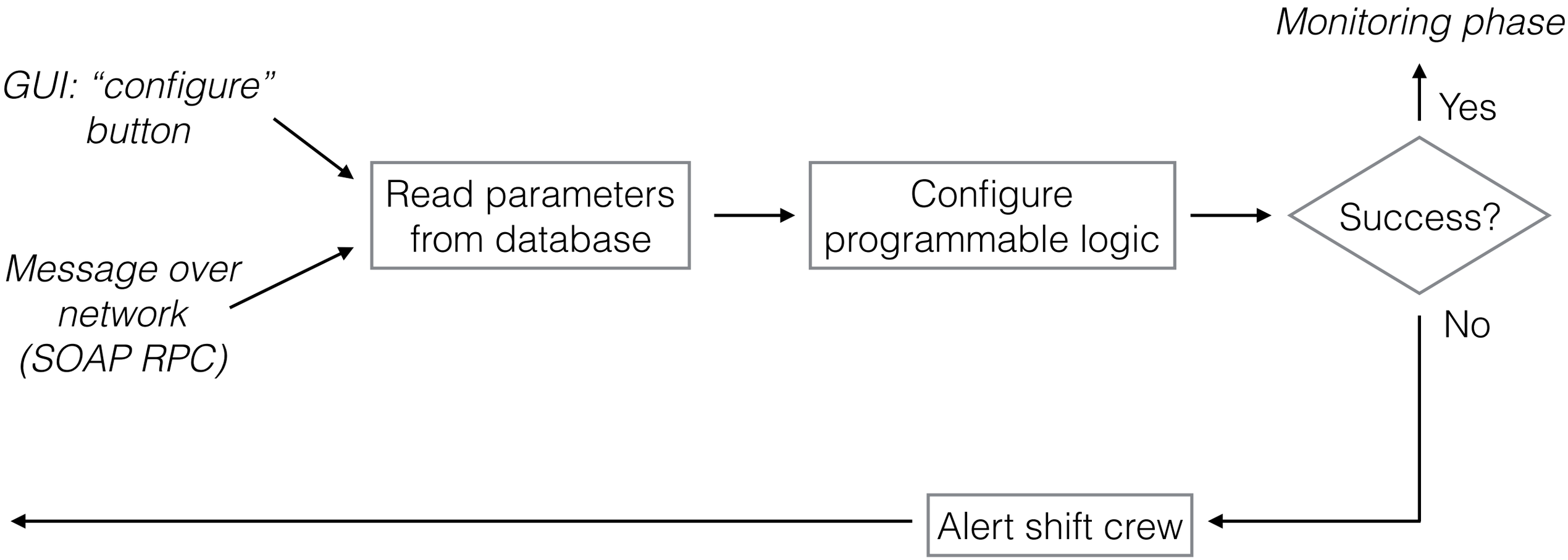


- 8 subsystems — different algorithms & scales
- O(100) boards, O(3000) links
- 5 different designs of processor boards

Online software:
Generic
Consistent

Online software requirements

- *Aim:* ***<u>Control</u> & monitor*** *electronics, with* ***high operational efficiency***

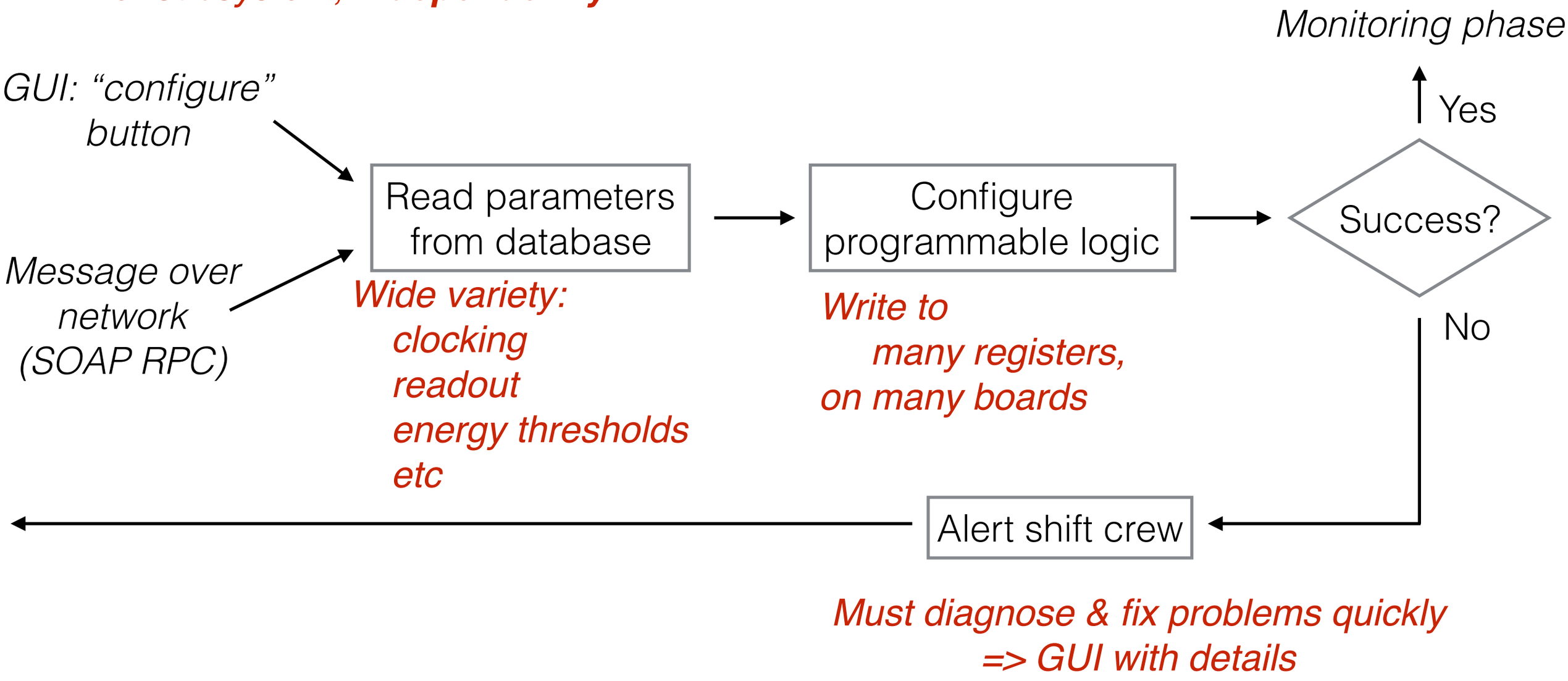


Online software requirements

- *Aim:* ***<u>Control</u> & monitor*** *electronics, with* ***high operational efficiency***

Reliable & predictable, ALWAYS!

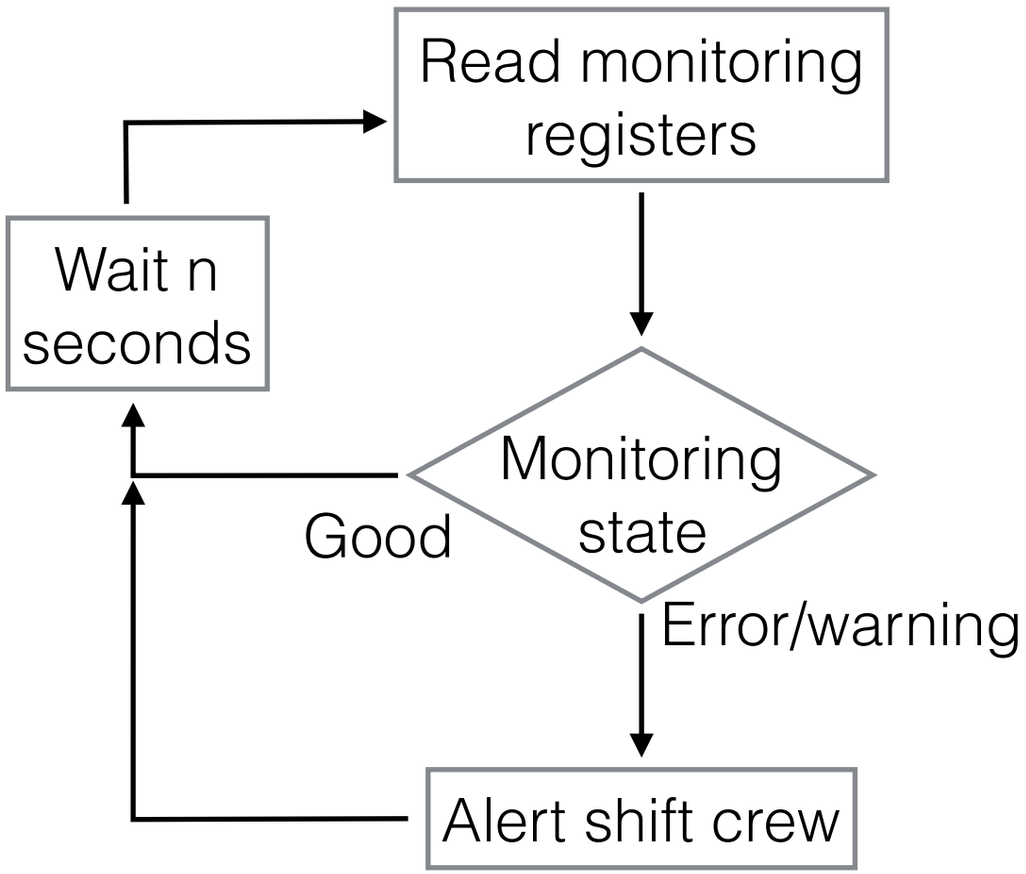
Two operation modes:
- *Entire system*
- *Per subsystem,* ***independently***



Must diagnose & fix problems quickly
=> GUI with details

Online software requirements

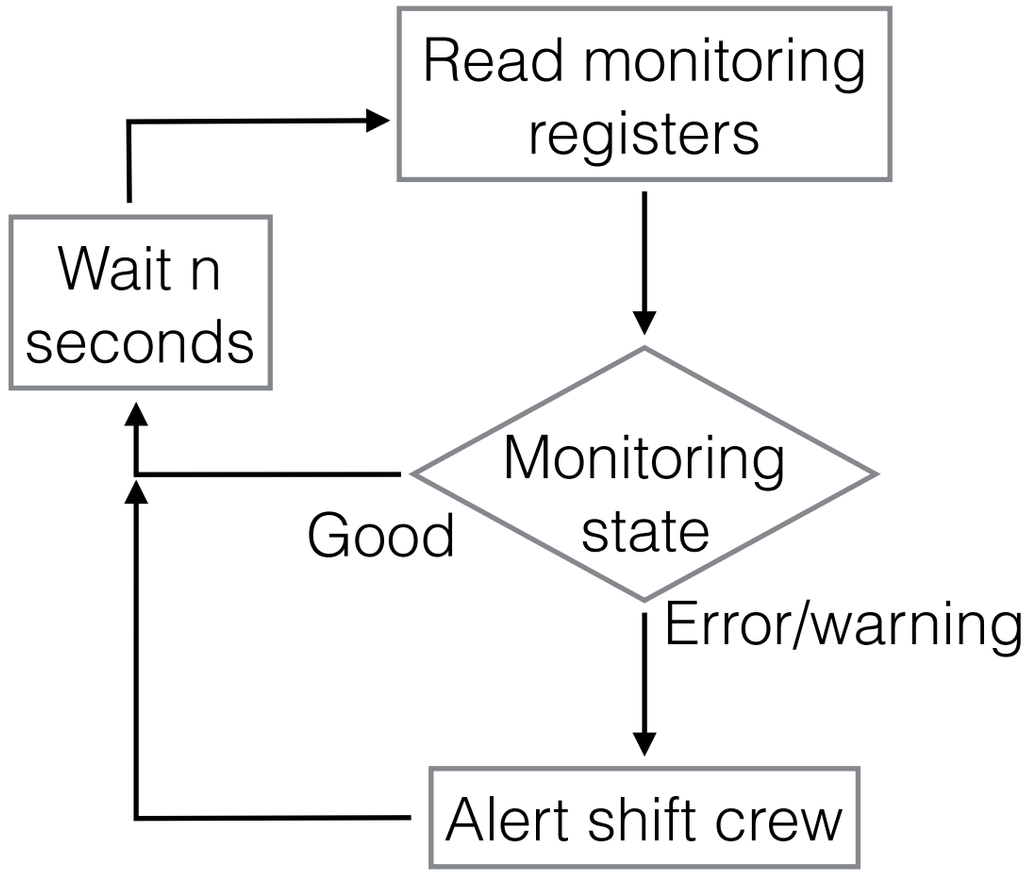
- *Aim:* ***Control & <u>monitor</u>*** *electronics, with* ***high operational efficiency***



Online software requirements

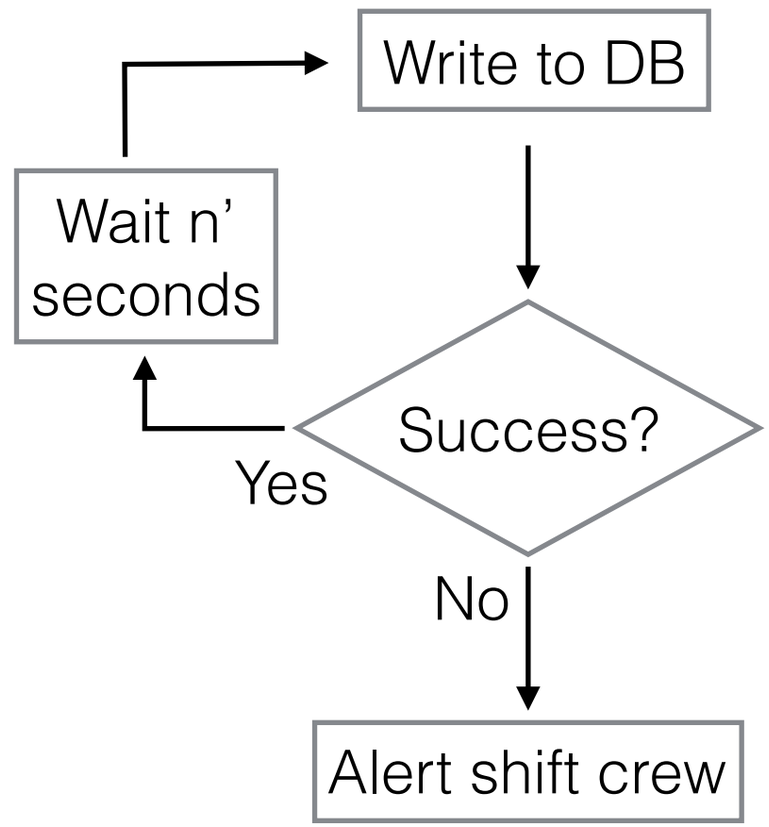
- *Aim:* ***Control & <u>monitor</u>*** *electronics, with* ***high operational efficiency***

Reliable & predictable, ALWAYS!



GUIs: enough info to quickly diagnose & fix

Archiving of monitoring data in database



Critical info
Small volume
Fix problems immediately

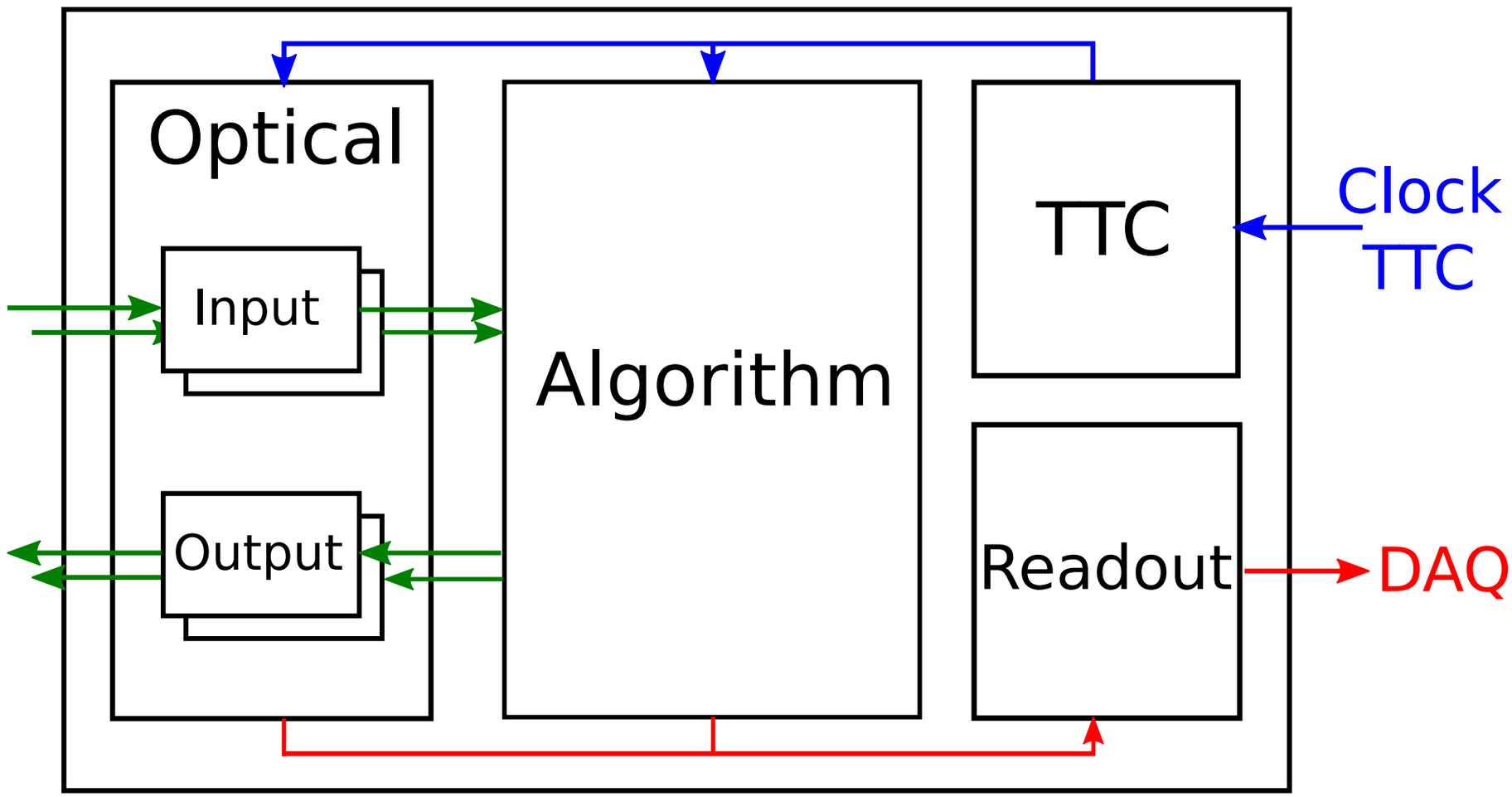
Non-critical info
Larger volume
Must not interfere with critical services

Even more online software requirements

- Commissioning: **Rapidly adapt** software
  - Operational needs & experience: **Continually evolve**
  - Short commissioning time: **No bugs/regressions!**
- Common interfaces: **Test-able without hardware**
- Development & maintenance:
  - **Minimise the person power**
- Maintainable lifetime: **several years**
- **Existing libraries**: Mustn't reinvent the wheel …
  - E.g. XDAQ & Trigger Supervisor — provide building blocks for distributed control & monitoring
- … and try to keep developers sane!

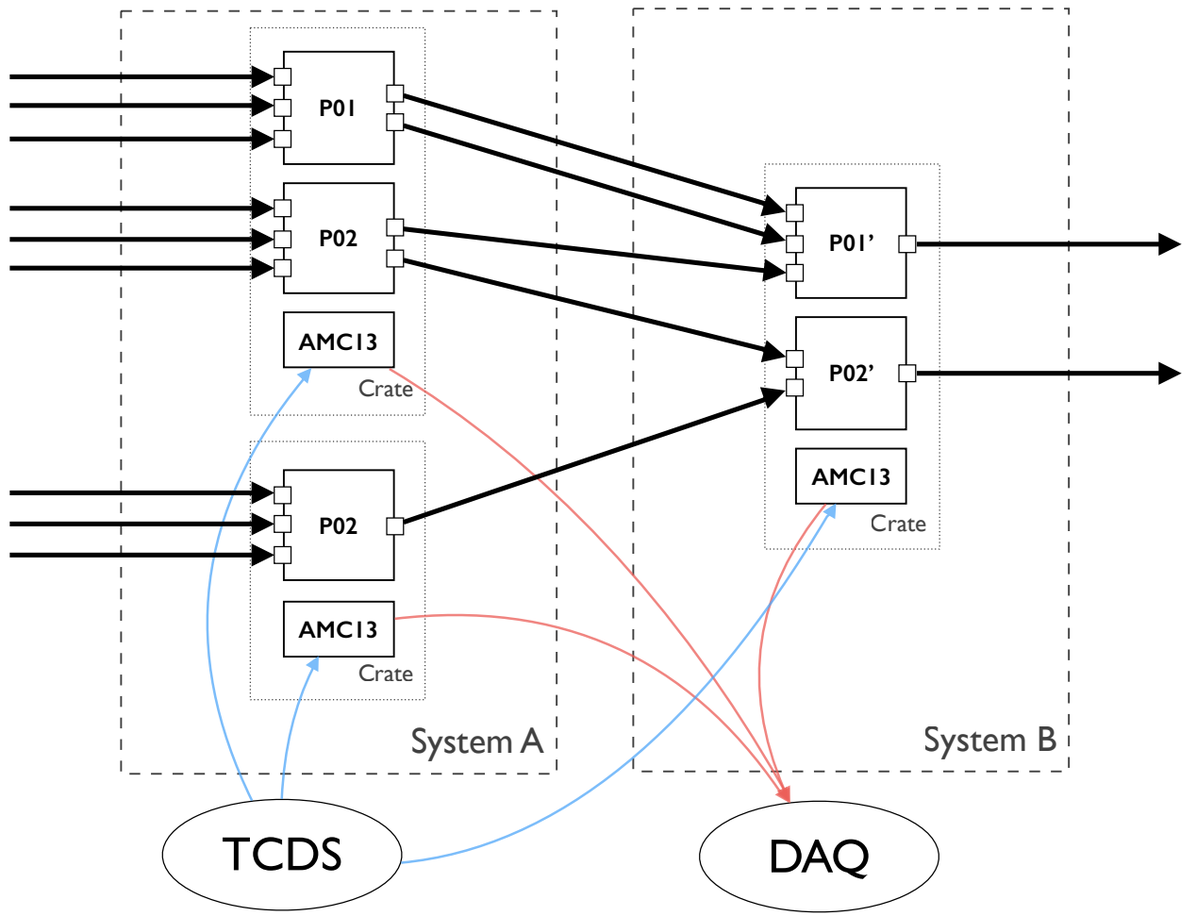
Common processor model

- Each subsystem's data-processing nodes …
  - are **all AMCs** following the MicroTCA specification
  - transmit/receive the trigger data on **high-speed serial optical links**
  - implement the processing logic in an **FPGA** (mainly Virtex 7)



Common system model

- Each subsystem:
  - One or more processor boards, in MicroTCA crates
  - One “AMC13” board in each crate:
    - Common CMS board; provides clock, timing & DAQ services via backplane



Trigger Control & Distribution System
clock & fixed-latency commands

Data AcQuisition network
sink for readout data

SWATCH framework (1)

- SWATCH: SoftWare for Automating the conTrol of Common Hardware
  - Design based on common processor/system models
  - Abstract C++ interfaces for controlling & monitoring hardware
  - Specialisation through inheritance
- Subsystem-agnostic description of hardware
  - E.g. locations & names of processors, optical I/O ports & AMC13s; interconnections
  - Factory pattern for creating instances of subsystem-specific classes
- Tricky tasks — e.g. thread-safety & multi-threading — implemented within generic API

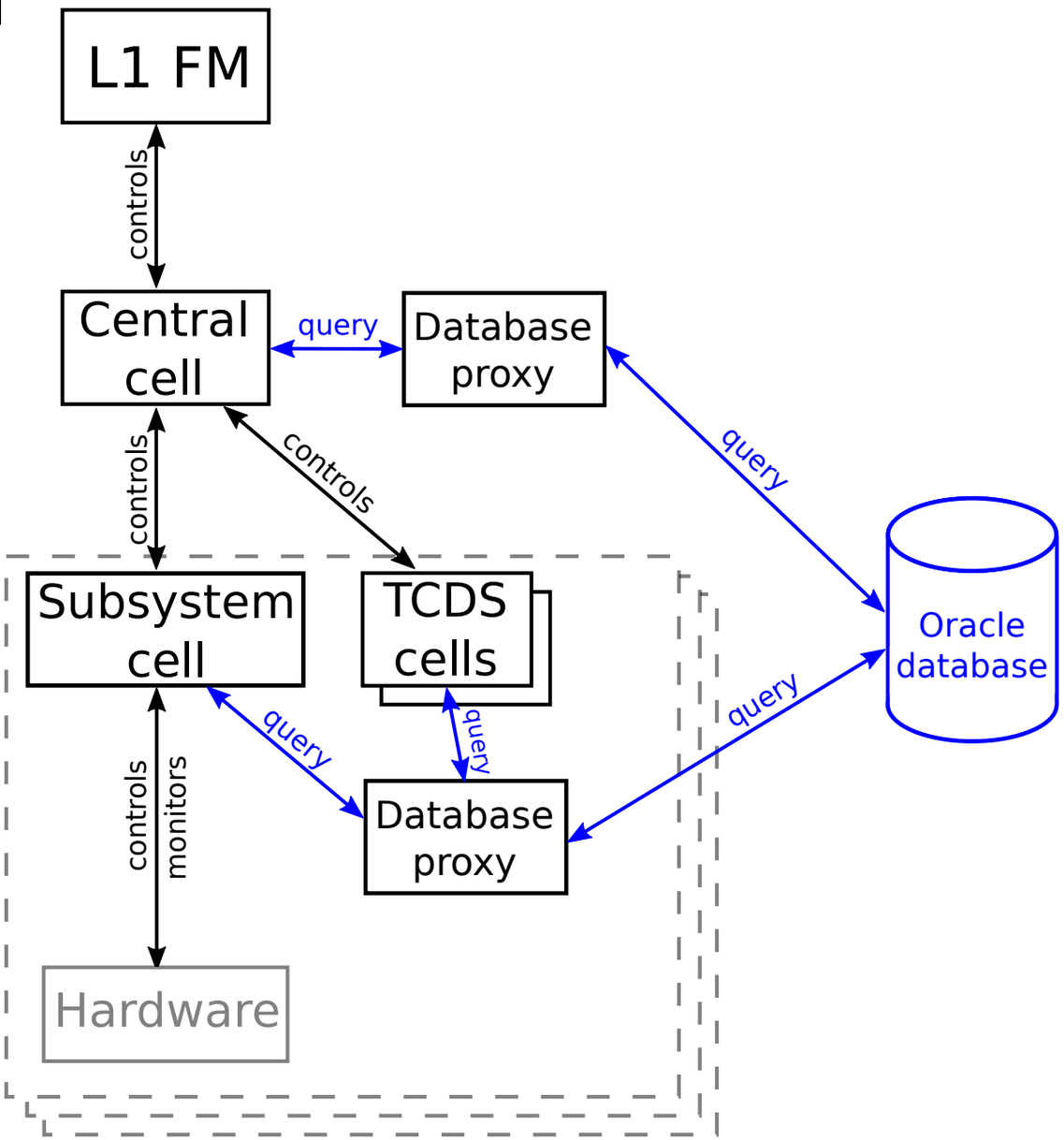
SWATCH framework (2)

- Generic interfaces for controlling individual boards:
  - **Command** — stateless action; basic building block
  - **Sequence of Commands**
  - **Finite State Machine (FSM)** — stateful actions
  - **Gatekeeper** — abstract pool of configuration parameters
    - **Independent of source** of the parameter values (file, database …)

- Generic interfaces for monitoring individual boards:
  - **Metric** — individual piece of monitoring data, read from hardware
    - Error/warning conditions
  - **Monitorable object**
    - Create multi-level trees of monitoring information

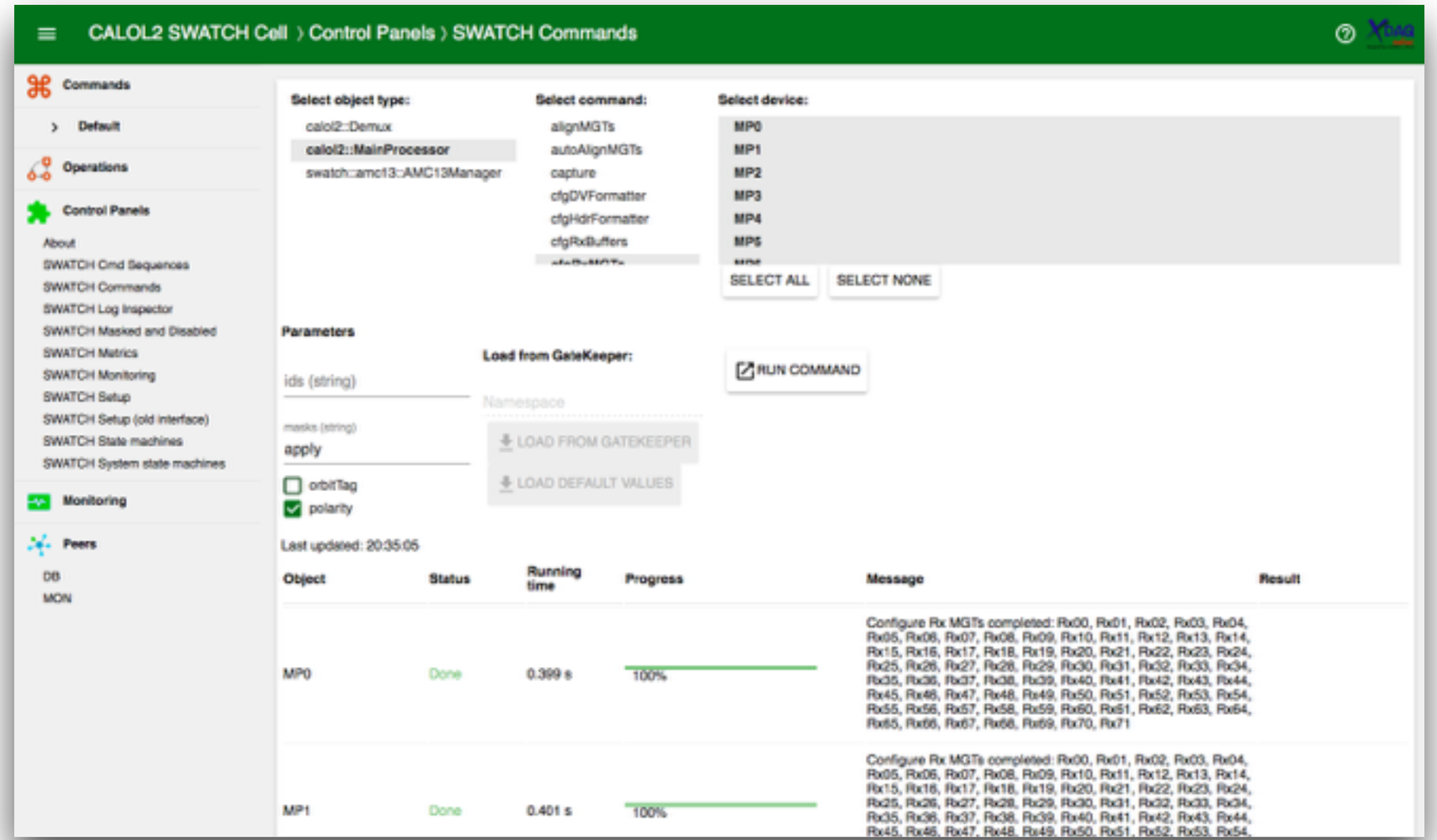
Distributed control & monitoring

- Level-1 Function Manager (L1FM):
  - Gateway from top-level CMS run control
  - java application
- Central cell:
  - Coordinates multiple subsystems
- SWATCH cell:
  - One per trigger subsystem
  - Contains a SWATCH system
  - Generic implementations of services:
    - Network-controllable “run control” FSM
    - Publish monitoring state
    - Retrieve parameters from DB
      - Common DB schema
    - Control/monitoring GUIs



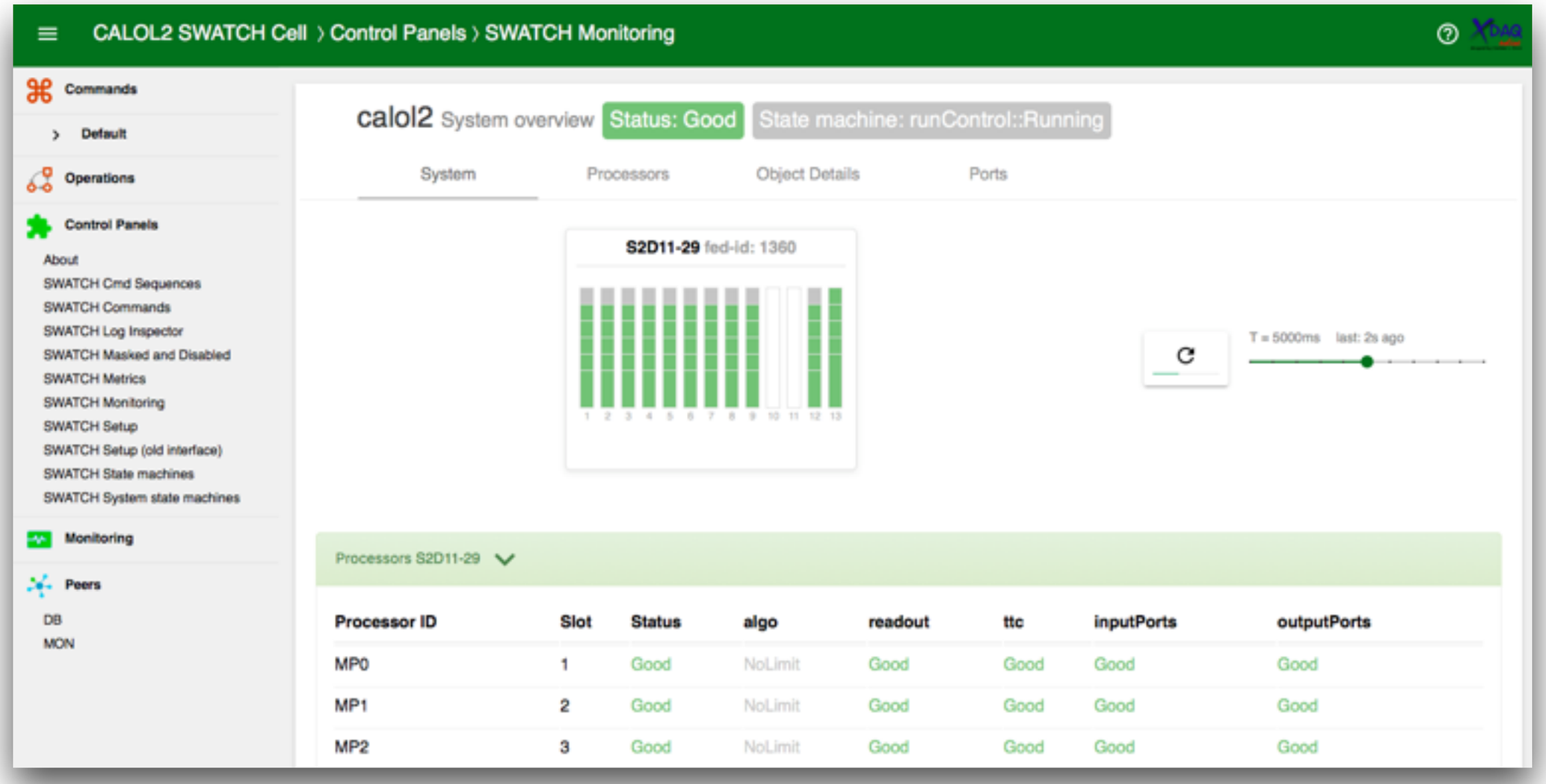
Generic GUIs

- Subsystem-agnostic SWATCH API:
  - Access to common & subsystem-specific control/monitoring primitives
  - Could develop detailed subsystem-agnostic GUIs:
    - Significantly reduced required person power
    - Uniform interface for operation personnel!



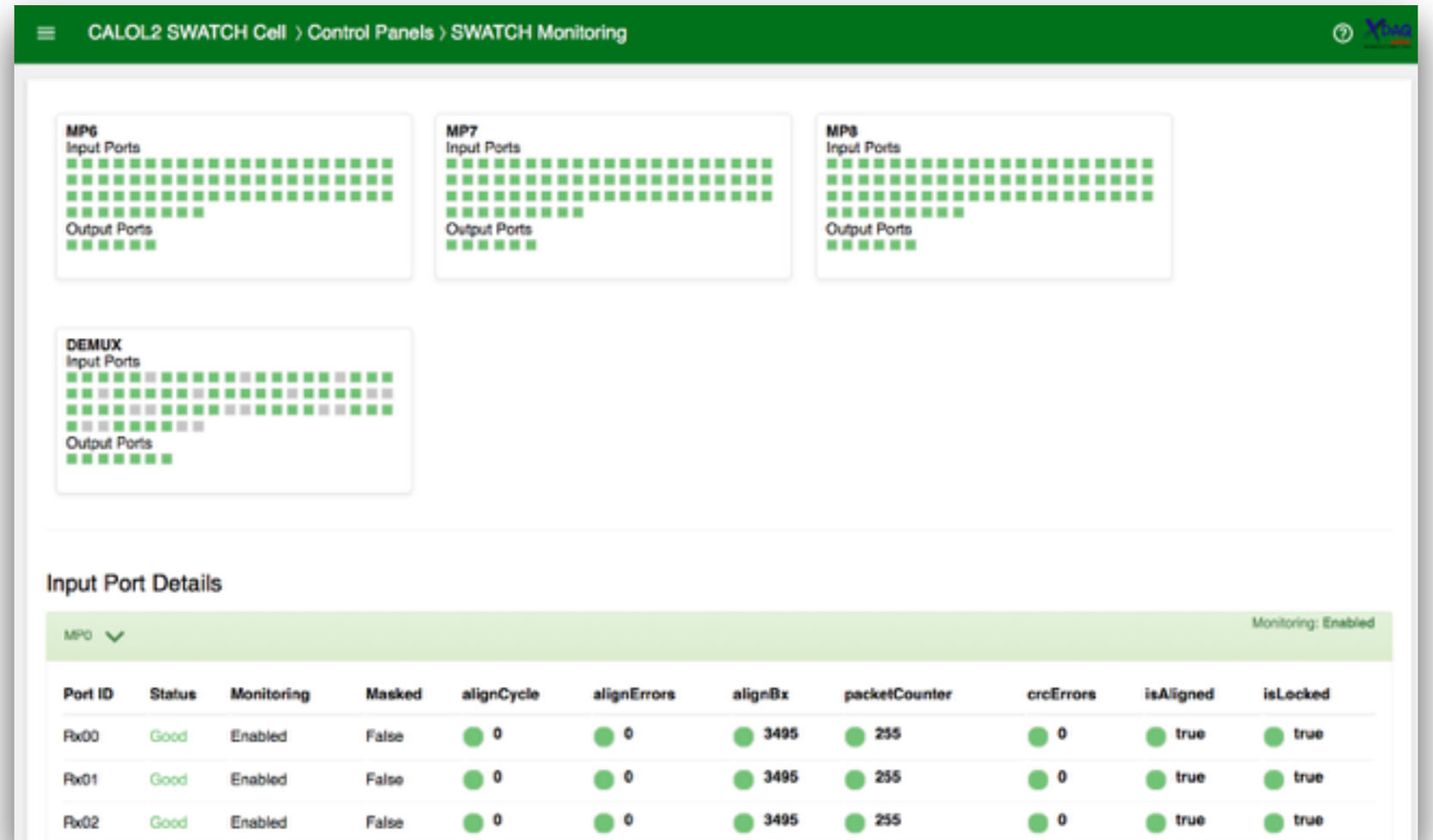
Generic GUIs

- Subsystem-agnostic SWATCH API:
  - Access to common & subsystem-specific control/monitoring primitives
  - Could develop detailed subsystem-agnostic GUIs:
    - Significantly reduced required person power
    - Uniform interface for operation personnel!



Generic GUIs

- Subsystem-agnostic SWATCH API:
  - Access to common & subsystem-specific control/monitoring primitives
  - Could develop detailed subsystem-agnostic GUIs:
    - Significantly reduced required person power
    - Uniform interface for operation personnel!



Commissioning: Did we survive?

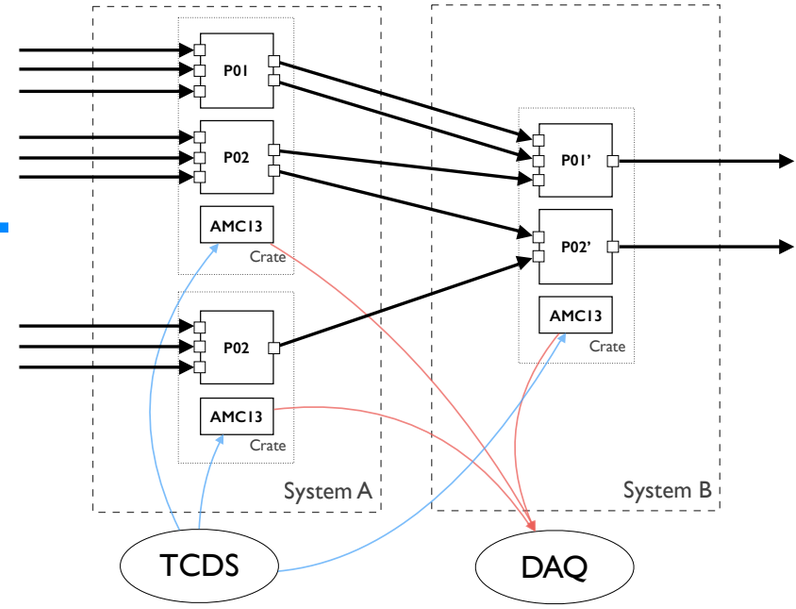
- Short integration phase:
  - **25th February: New system used as CMS trigger** (no beams)
    - 1st time **online software integrated into global "run control" hierarchy**
  - **25th March: Trigger on "splashes"** (beam commissioning)

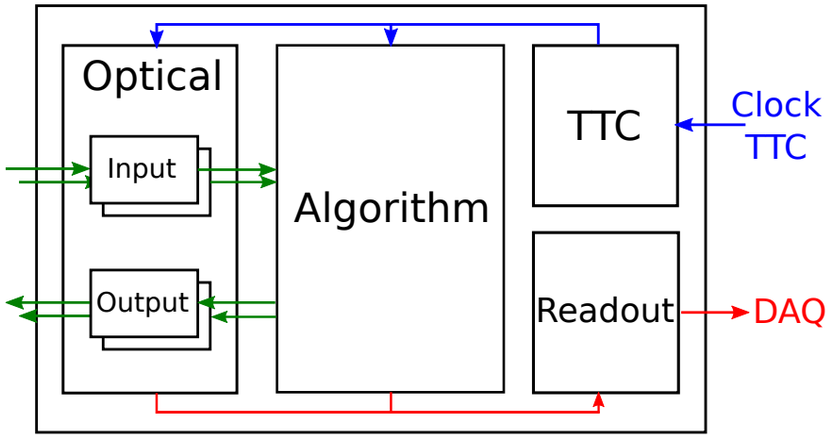
- **High operational efficiency** of software, despite various new features, e.g:
  - **Transition** from file-based configuration **to database**
  - Major **migration of GUIs to HTML5**-based libraries (Polymer)

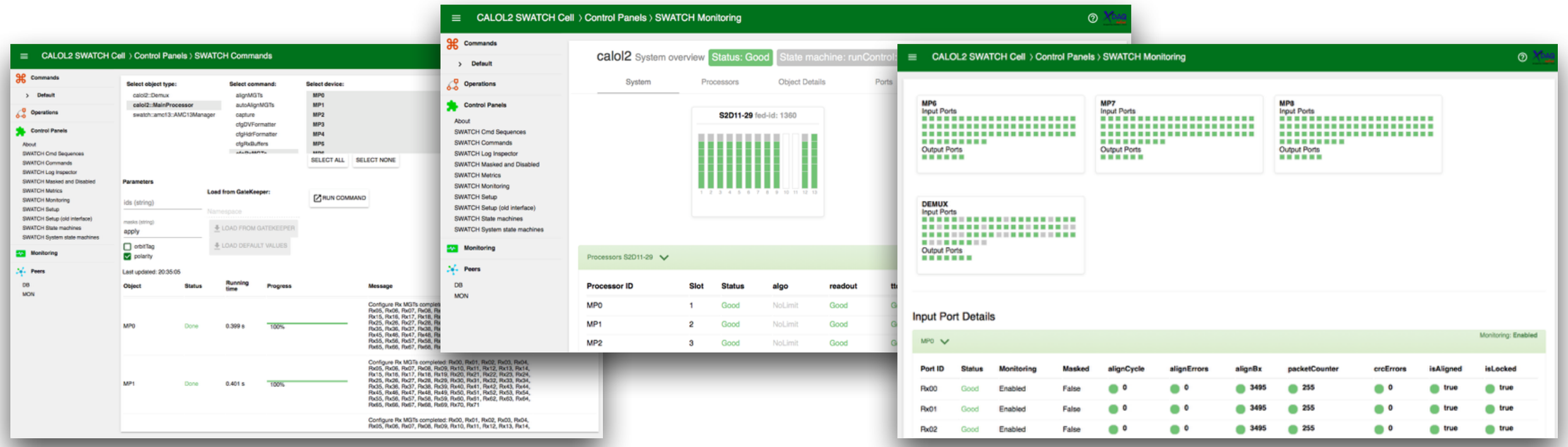
- Success critically depended on design of SWATCH:
  - **Large fraction of software is common**
  - **Common SWATCH plugins** used wherever common firmware is used
  - **Extensive hardware-independent testing**

Conclusions

- Successfully controlled & monitored upgrade system since February
- SWATCH API based on detailed common model of hardware & firmware
  - Flexible & reliable approach
- Increase in uniformity & fraction of common software across subsystems

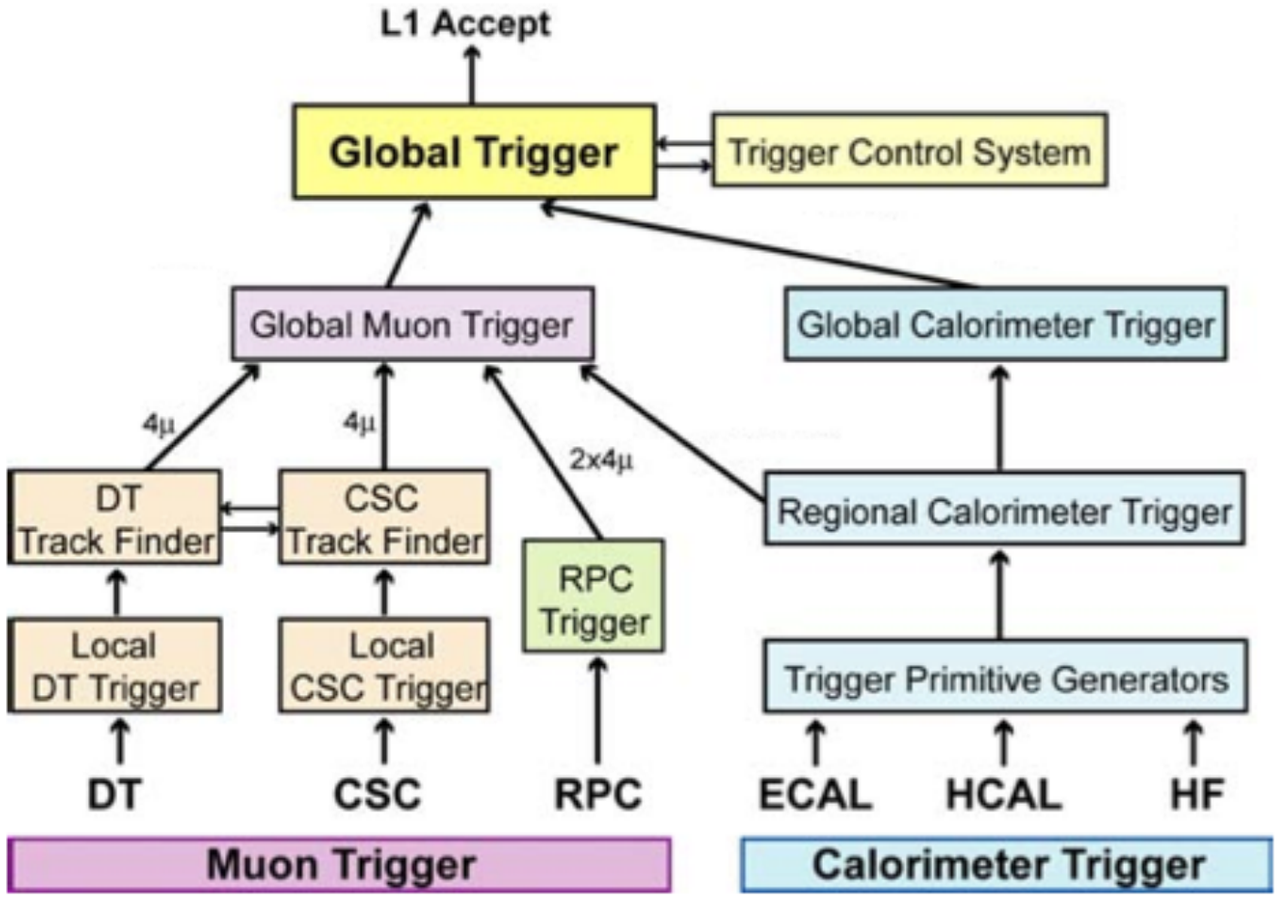






BACKUP: Legacy trigger system

- Level-1 trigger before 2015:
  - Several subsystems
  - O(4000) processor boards custom application-specific designs



- Controlled and monitored by medium-sized distributed system
  - Approx. 40 computers, 200 processes
- Small fraction of online software common between subsystems
- *High long-term maintenance costs & risk of losing critical knowledge through turnover of developers*

BACKUP: Integrating drivers into SWATCH

- Low-level driver software — libraries performing individual register reads/writes
  - Independent of SWATCH for most boards
  - SWATCH “plugin” libraries implement corresponding SWATCH commands & monitorable objects
  - Upgraded trigger uses common hardware & firmware
    - E.g. AMC13, common infrastructural firmware blocks (TTC, DAQ, optical I/O)
    - One common “SWATCH” plugin for each common component
      - Greatly reduced commissioning time