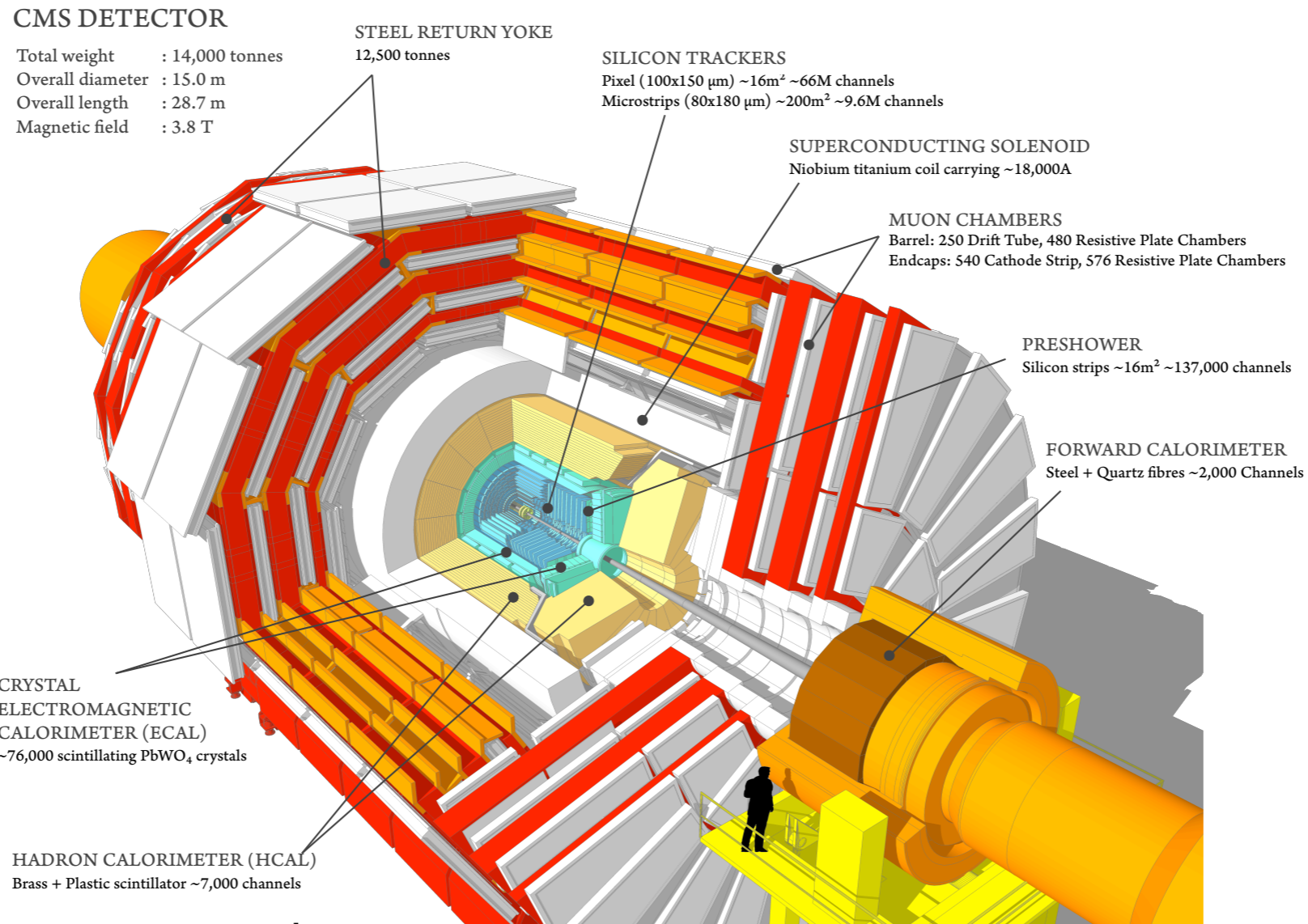


SWATCH: Common software for controlling and monitoring the upgraded Level-1 trigger of the CMS experiment

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

CMS Level-1 trigger system

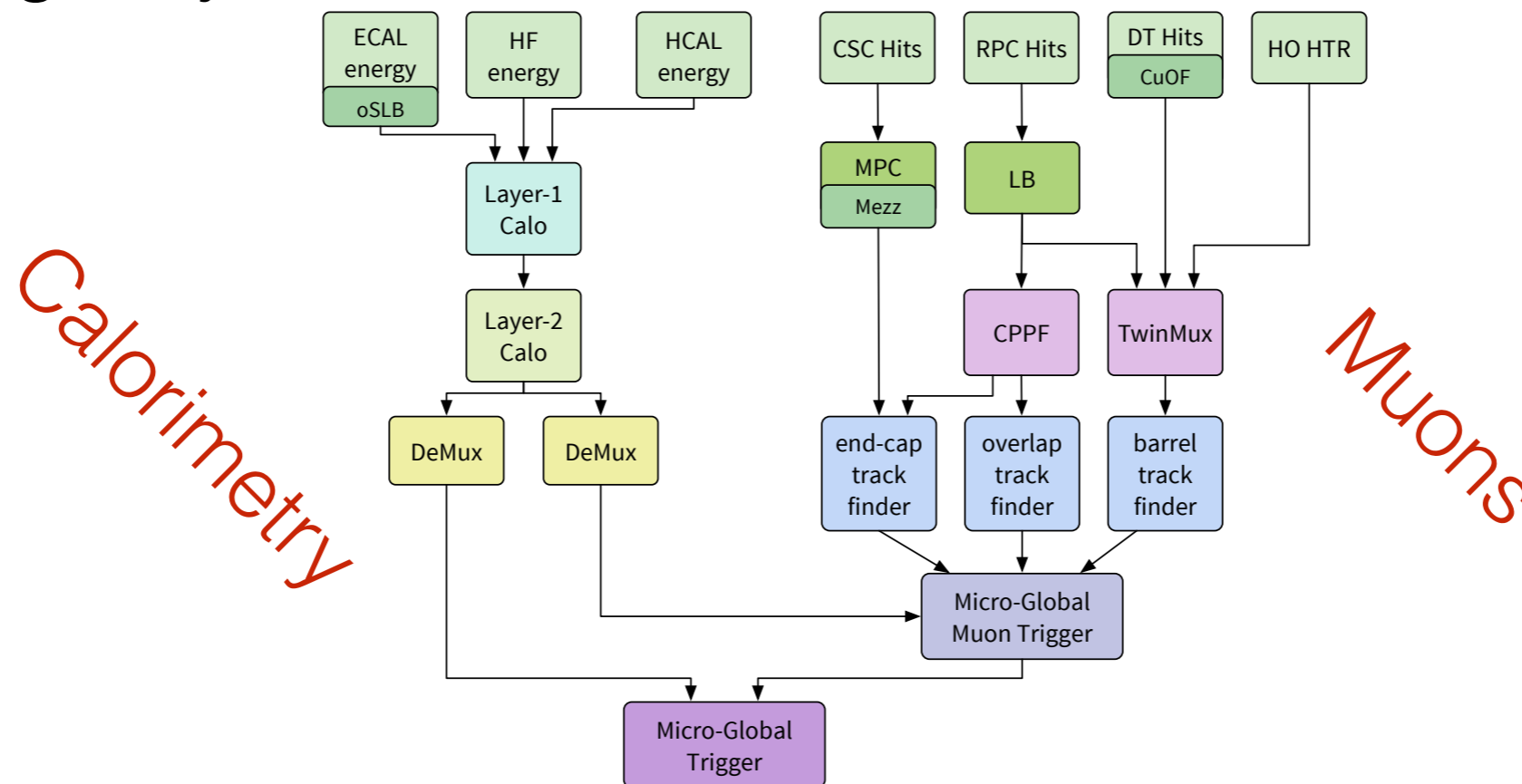


- Level-1 trigger system:

- Selects 100 kHz of interesting events from 40MHz pp collisions
- Fixed latency: Decision within 3.8 μs , 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:

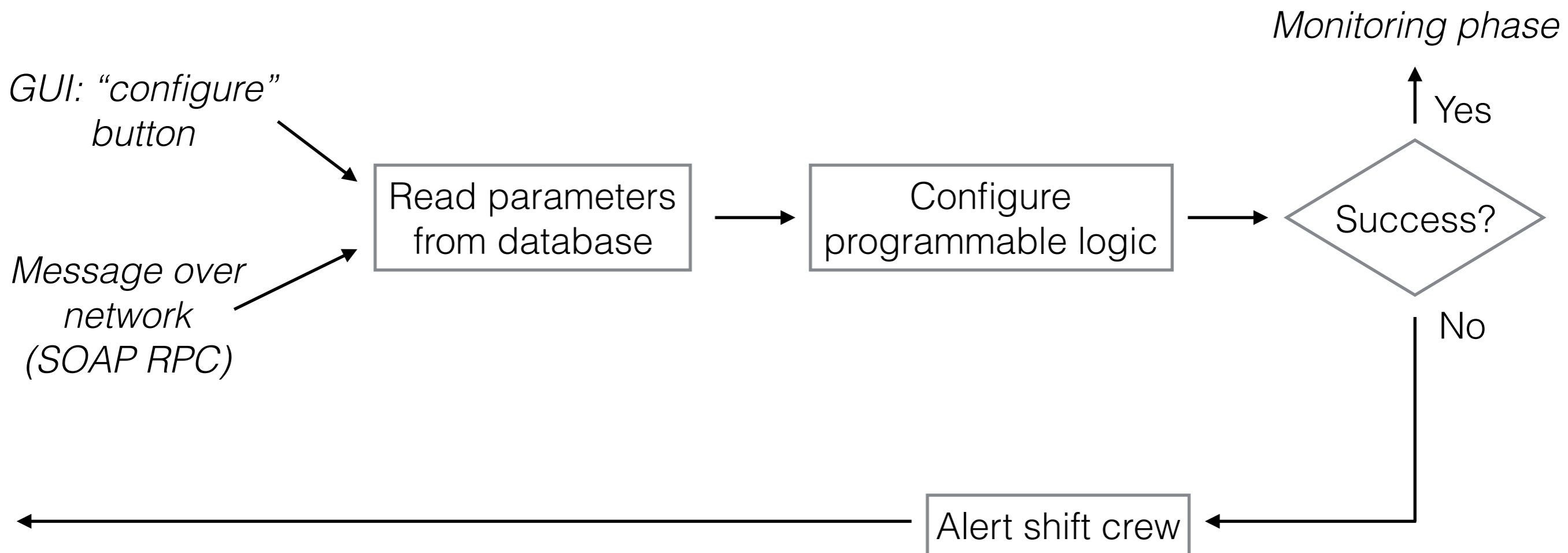


- 8 subsystems — different algorithms & scales
- $O(100)$ boards, $O(3000)$ links
- 5 different designs of processor boards

*Online software:
Generic
Consistent*

Online software requirements

- Aim: ***Control & monitor*** electronics, with ***high operational efficiency***



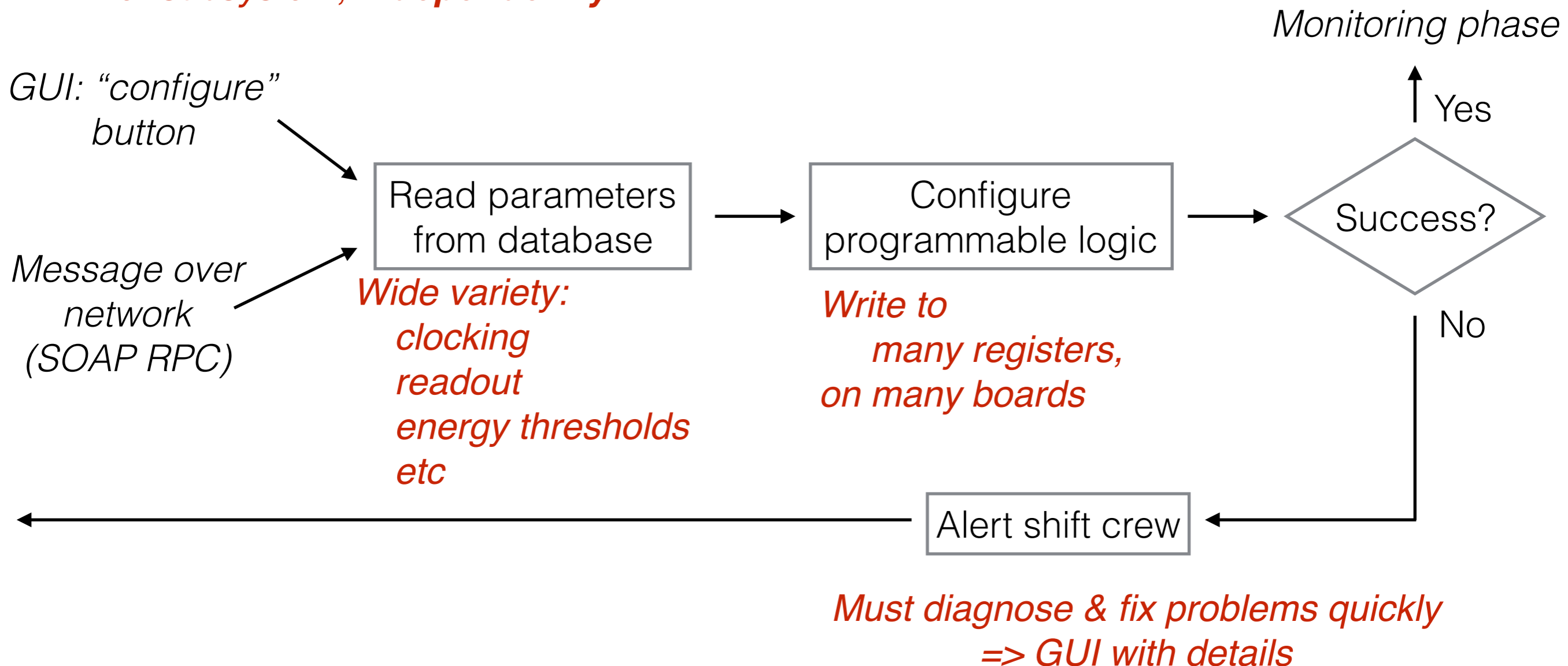
Online software requirements

- Aim: **Control & monitor** electronics, with **high operational efficiency**

Reliable & predictable, ALWAYS!

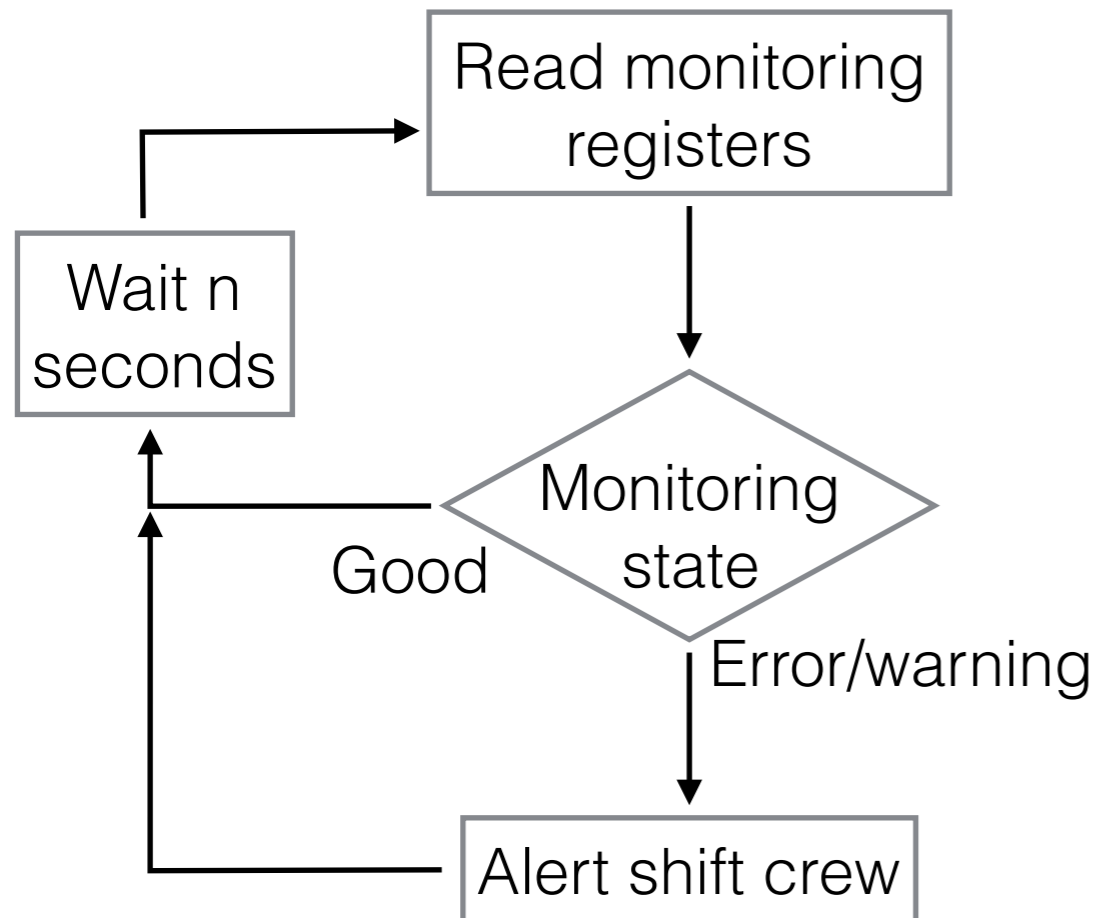
Two operation modes:

- Entire system
- Per subsystem, **independently**



Online software requirements

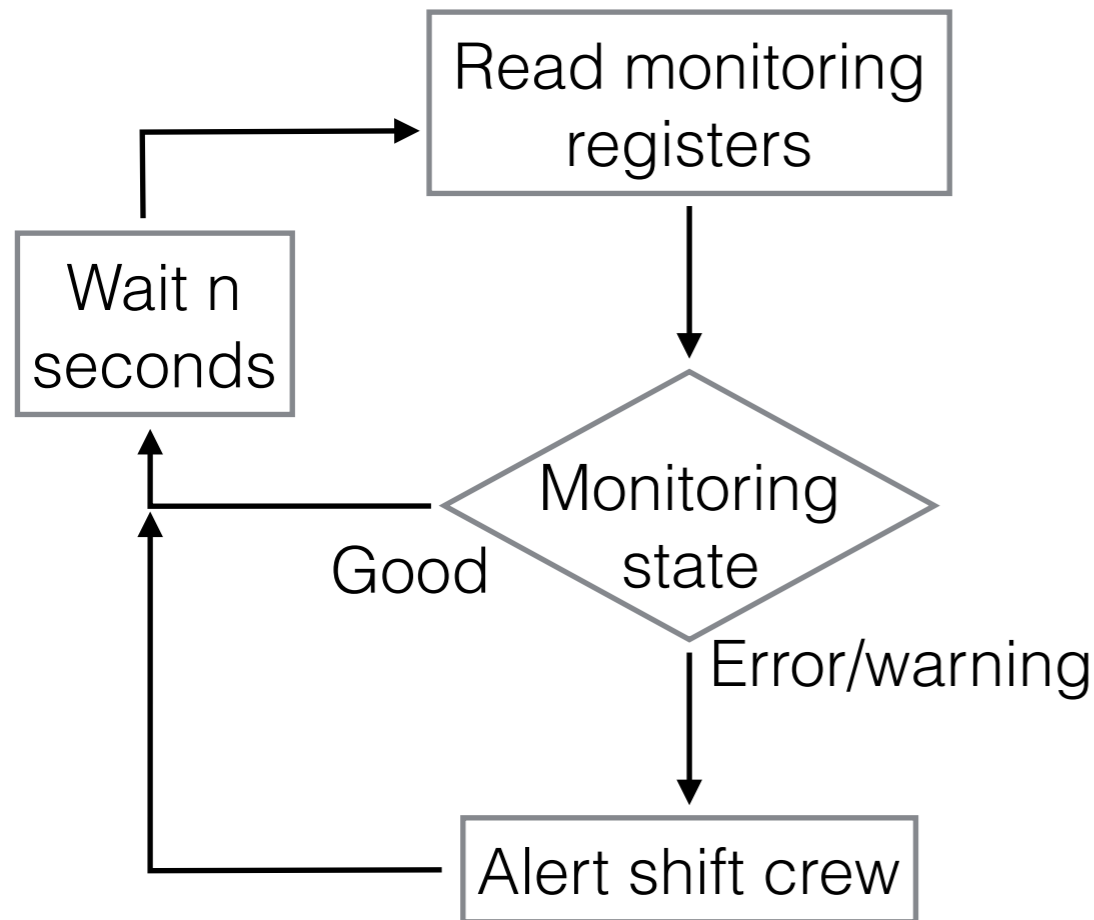
- Aim: **Control & monitor** electronics, with **high operational efficiency**



Online software requirements

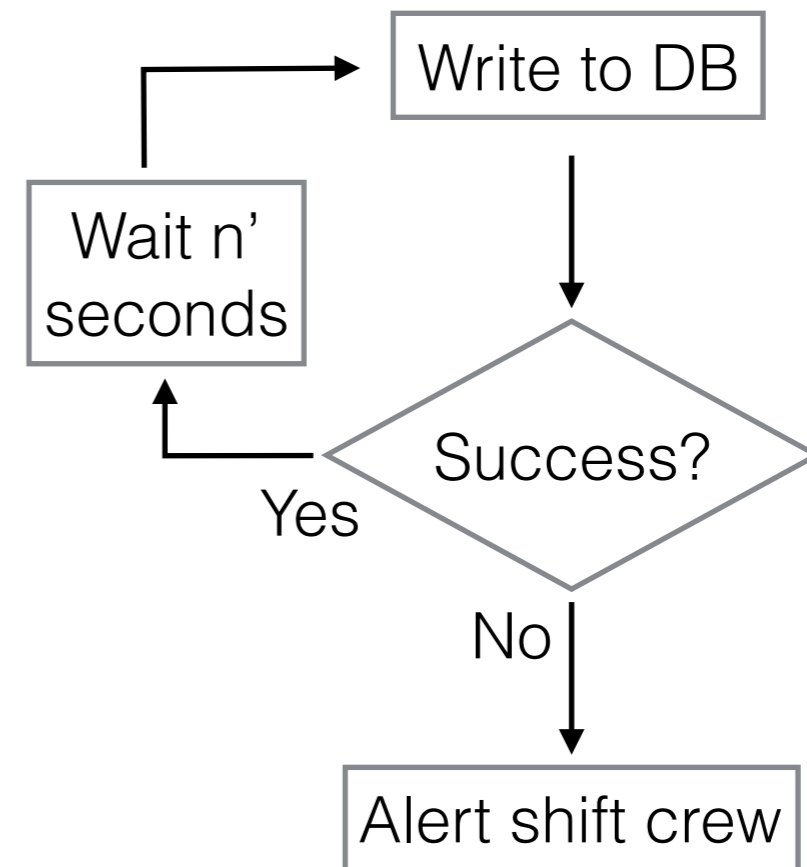
- Aim: **Control & monitor** electronics, with **high operational efficiency**

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

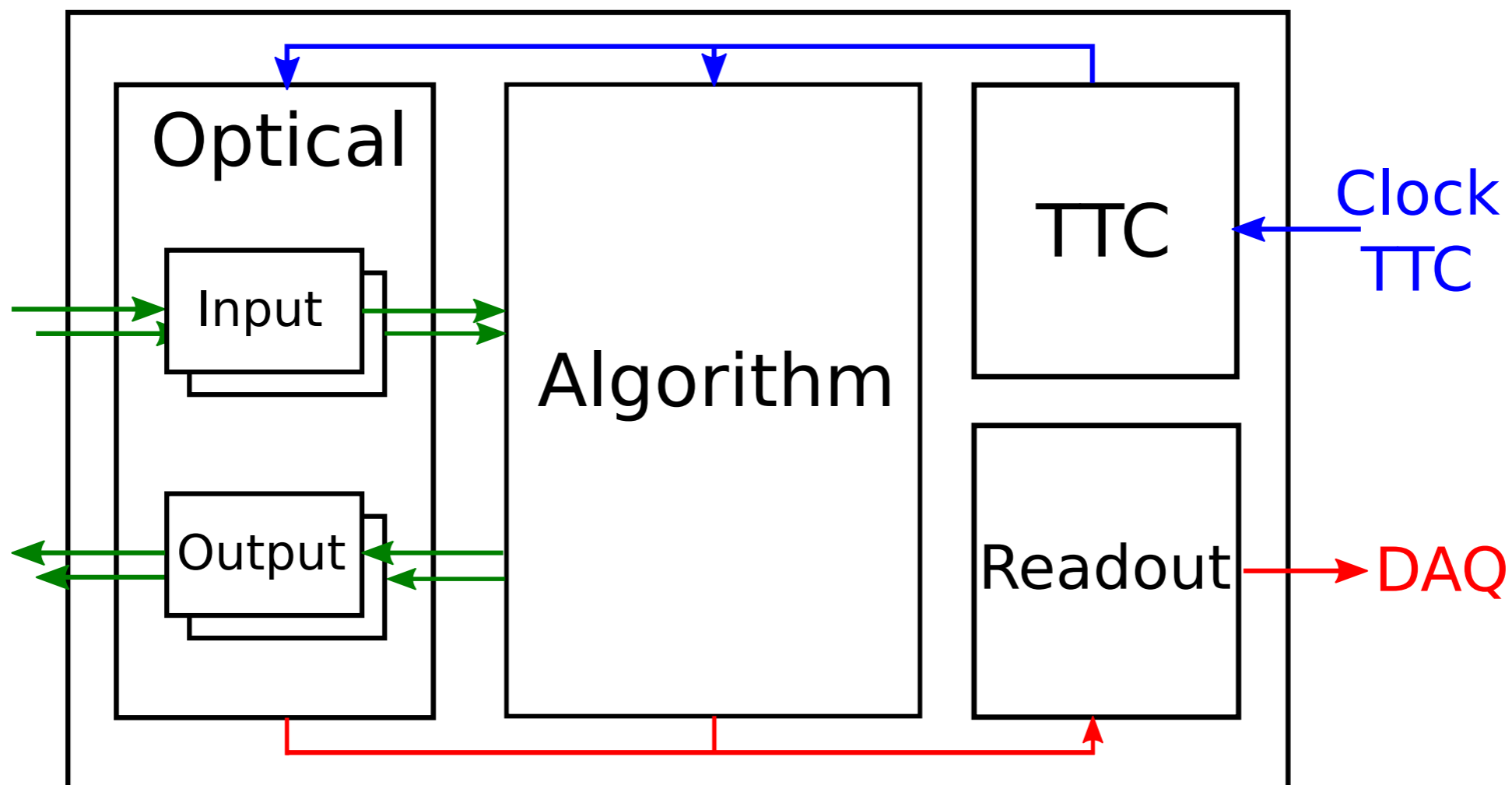
Even more online software requirements

- 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!

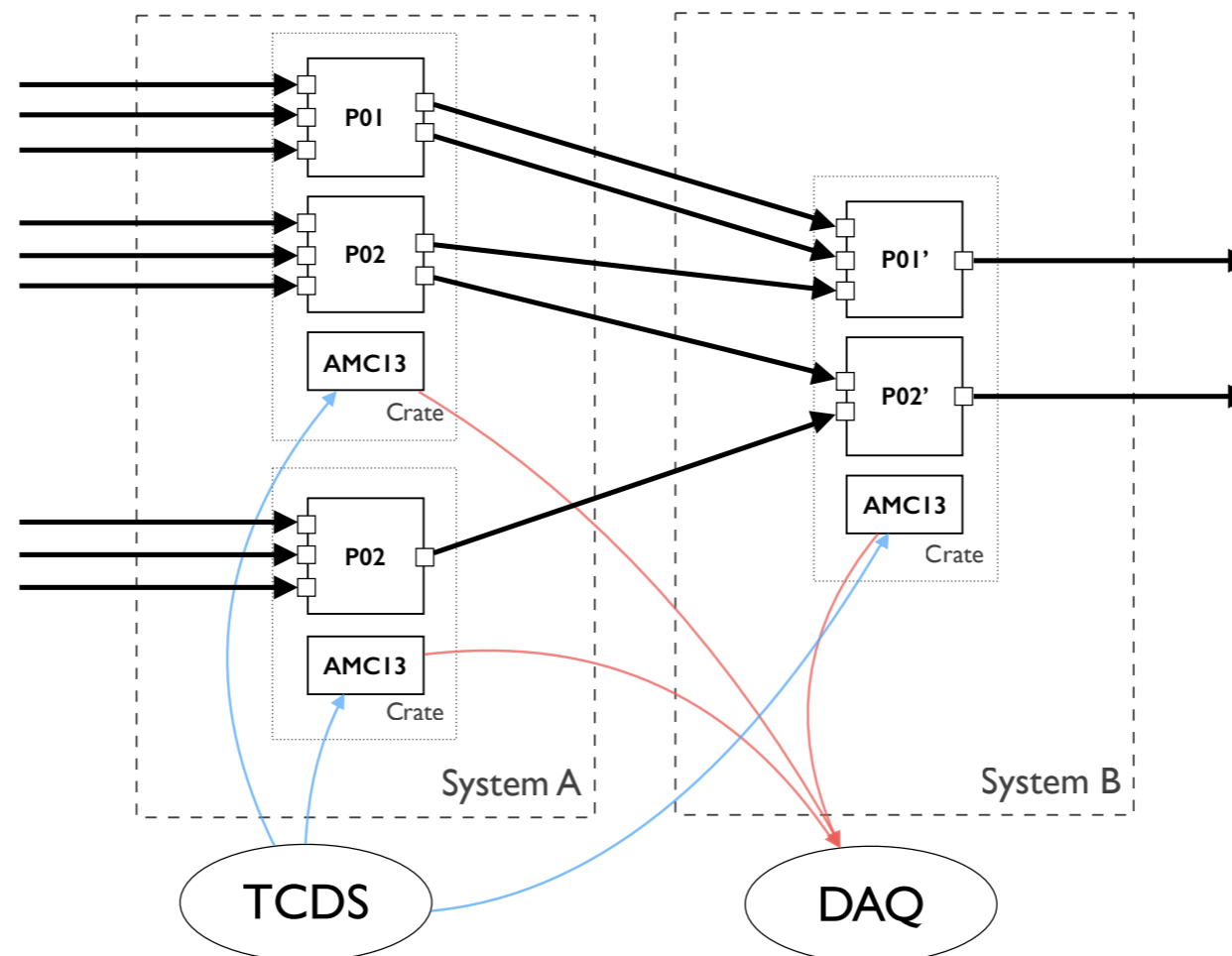
Common processor model

- 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)

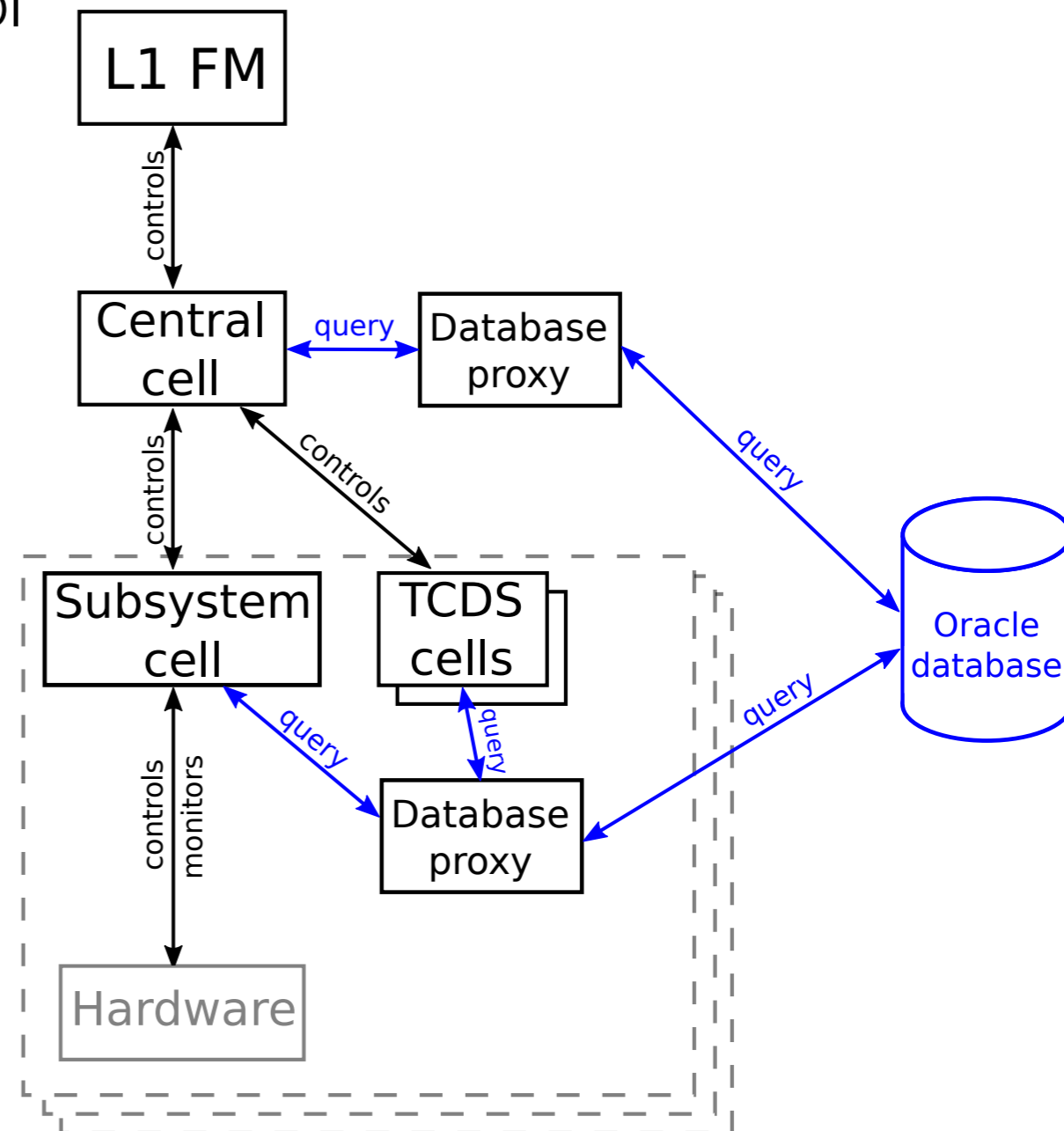
- SWATCH: SoftWare for Automating the conTrol of Common Hardware
 - Design based on common processor/system models
 - Abstract C++ interfaces for controlling & monitoring hardware
 - Specialisation through inheritance
- Subsystem-agnostic description of hardware
 - E.g. locations & names of processors, 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!

The screenshot displays the 'CALOL2 SWATCH Cell' interface, specifically the 'Control Panels > SWATCH Commands' section. The left sidebar contains navigation menus for 'Commands', 'Operations', 'Control Panels', 'Monitoring', and 'Peers'. The main content area is organized into several functional sections:

- Select object type:** A list of objects including 'calol2::Demux', 'calol2::MainProcessor', and 'swatch::amc13:AMC13Manager'.
- Select command:** A list of commands such as 'alignMGTs', 'autoAlignMGTs', 'capture', 'clgDVFormatter', 'clgHdrFormatter', and 'clgRxBuffers'.
- Select device:** A list of devices labeled 'MP0' through 'MP5'.
- Parameters:** Fields for 'ids (string)', 'marks (string)', and 'apply', along with checkboxes for 'orbitTag' and 'polarity'.
- Load from GateKeeper:** A section with 'LOAD FROM GATEKEEPER' and 'LOAD DEFAULT VALUES' buttons.
- Execution Log:** A table showing the results of command execution for devices MP0 and MP1.

Object	Status	Running time	Progress	Message	Result
MP0	Done	0.399 s	100%	Configure Rx MGIs completed: Rx00, Rx01, Rx02, Rx03, Rx04, Rx05, Rx06, Rx07, Rx08, Rx09, Rx10, Rx11, Rx12, Rx13, Rx14, Rx15, Rx16, Rx17, Rx18, Rx19, Rx20, Rx21, Rx22, Rx23, Rx24, Rx25, Rx26, Rx27, Rx28, Rx29, Rx30, Rx31, Rx32, Rx33, Rx34, Rx35, Rx36, Rx37, Rx38, Rx39, Rx40, Rx41, Rx42, Rx43, Rx44, Rx45, Rx46, Rx47, Rx48, Rx49, Rx50, Rx51, Rx52, Rx53, Rx54, Rx55, Rx56, Rx57, Rx58, Rx59, Rx60, Rx61, Rx62, Rx63, Rx64, Rx65, Rx66, Rx67, Rx68, Rx69, Rx70, Rx71	
MP1	Done	0.401 s	100%	Configure Rx MGIs completed: Rx00, Rx01, Rx02, Rx03, Rx04, Rx05, Rx06, Rx07, Rx08, Rx09, Rx10, Rx11, Rx12, Rx13, Rx14, Rx15, Rx16, Rx17, Rx18, Rx19, Rx20, Rx21, Rx22, Rx23, Rx24, Rx25, Rx26, Rx27, Rx28, Rx29, Rx30, Rx31, Rx32, Rx33, Rx34, Rx35, Rx36, Rx37, Rx38, Rx39, Rx40, Rx41, Rx42, Rx43, Rx44, Rx45, Rx46, Rx47, Rx48, Rx49, Rx50, Rx51, Rx52, Rx53, Rx54	

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!

The screenshot displays the 'CALOL2 SWATCH Cell > Control Panels > SWATCH Monitoring' interface. The main content area shows a 'calol2 System overview' with a 'Status: Good' indicator and a 'State machine: runControl::Running' label. Below this, there are tabs for 'System', 'Processors', 'Object Details', and 'Ports'. The 'Processors' tab is active, showing a bar chart for 'S2D11-29 fed-id: 1360' with 13 slots. Slots 1-12 are green, and slot 13 is white. A refresh button and a timer 'T = 5000ms last: 2s ago' are also visible. Below the chart is a table titled 'Processors S2D11-29' with the following data:

Processor ID	Slot	Status	algo	readout	ttc	inputPorts	outputPorts
MP0	1	Good	NoLimit	Good	Good	Good	Good
MP1	2	Good	NoLimit	Good	Good	Good	Good
MP2	3	Good	NoLimit	Good	Good	Good	Good

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!

The screenshot displays the 'CALOL2 SWATCH Cell > Control Panels > SWATCH Monitoring' interface. It features four monitoring panels: MP6, MP7, MP8, and DEMUX. Each panel shows a grid of input and output ports with green indicators representing their status. Below these panels is an 'Input Port Details' section for MP0, which includes a table of port data.

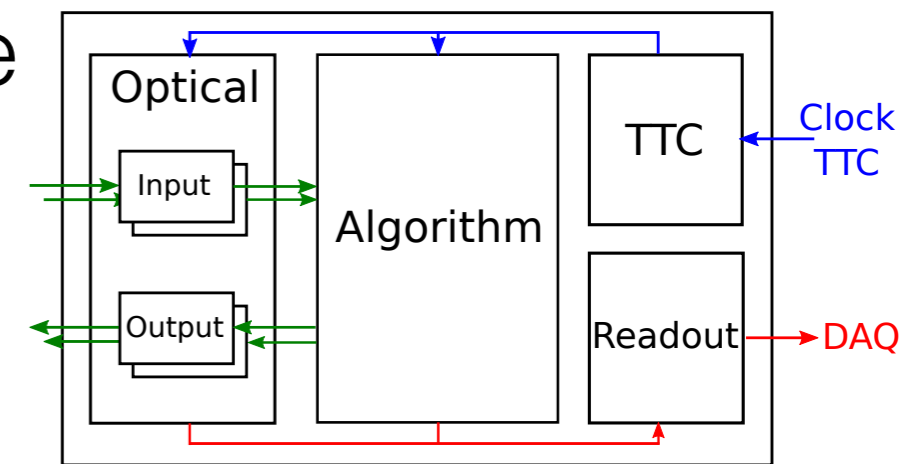
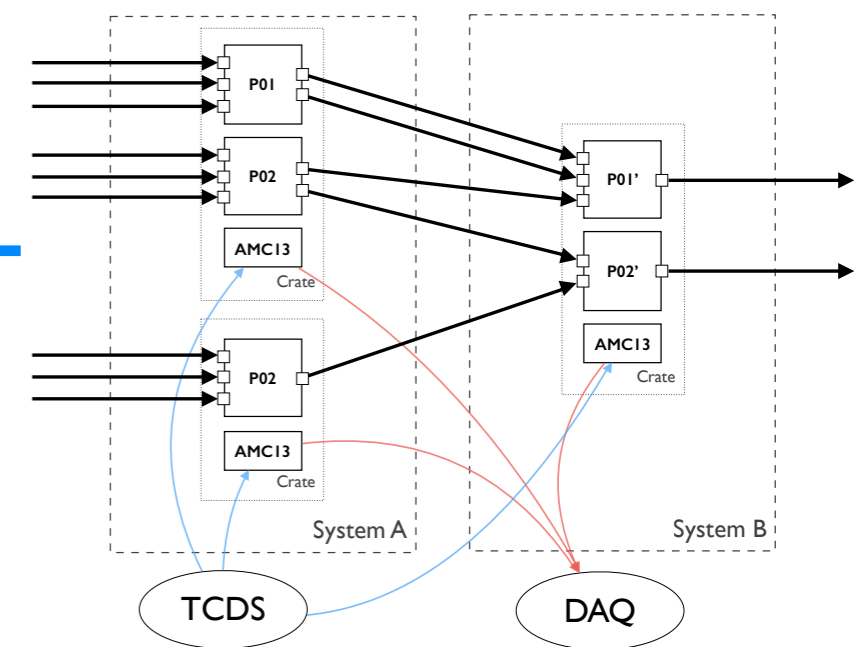
Port ID	Status	Monitoring	Masked	alignCycle	alignErrors	alignBx	packetCounter	crcErrors	isAligned	isLocked
Rx00	Good	Enabled	False	0	0	3495	255	0	true	true
Rx01	Good	Enabled	False	0	0	3495	255	0	true	true
Rx02	Good	Enabled	False	0	0	3495	255	0	true	true

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 SWATCCH:
 - **Large fraction of software is common**
 - **Common SWATCCH 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



The screenshots show the SWATCH Monitoring interface. The top panel displays the system overview for S2D11-29, showing a status of 'Good' and a state machine of 'runControl'. Below this, a table lists the processor status:

Processor ID	Slot	Status	algo	readout
MP0	1	Good	NoLimit	Good
MP1	2	Good	NoLimit	Good
MP2	3	Good	NoLimit	Good

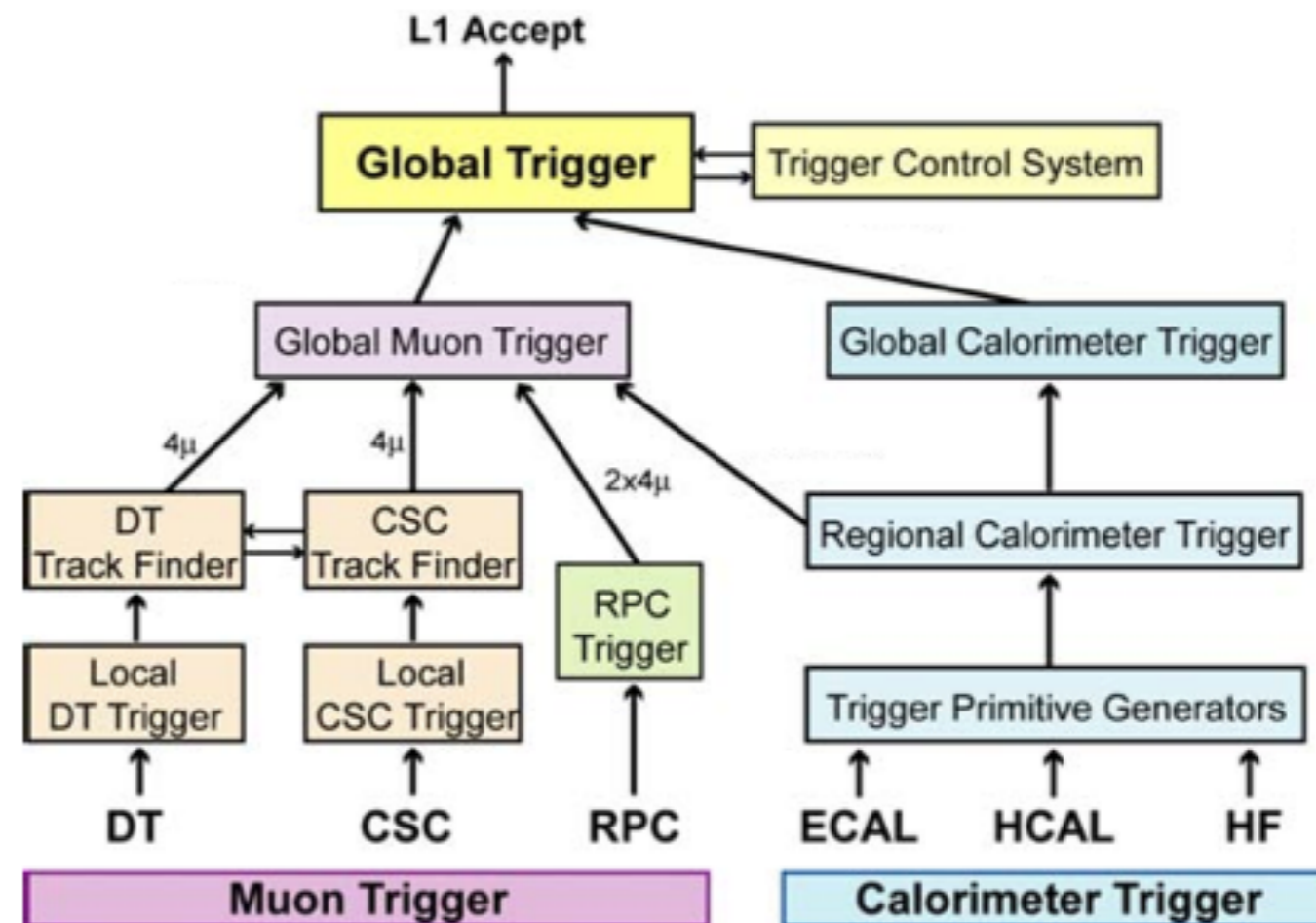
The bottom panel shows the input port details for MP0, with a table listing the status of each port:

Port ID	Status	Monitoring	Masked	alignCycle	alignErrors	alignBx	packetCounter	crcErrors	isAligned	isLocked
Rx00	Good	Enabled	False	0	0	3495	255	0	true	true
Rx01	Good	Enabled	False	0	0	3495	255	0	true	true
Rx02	Good	Enabled	False	0	0	3495	255	0	true	true

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