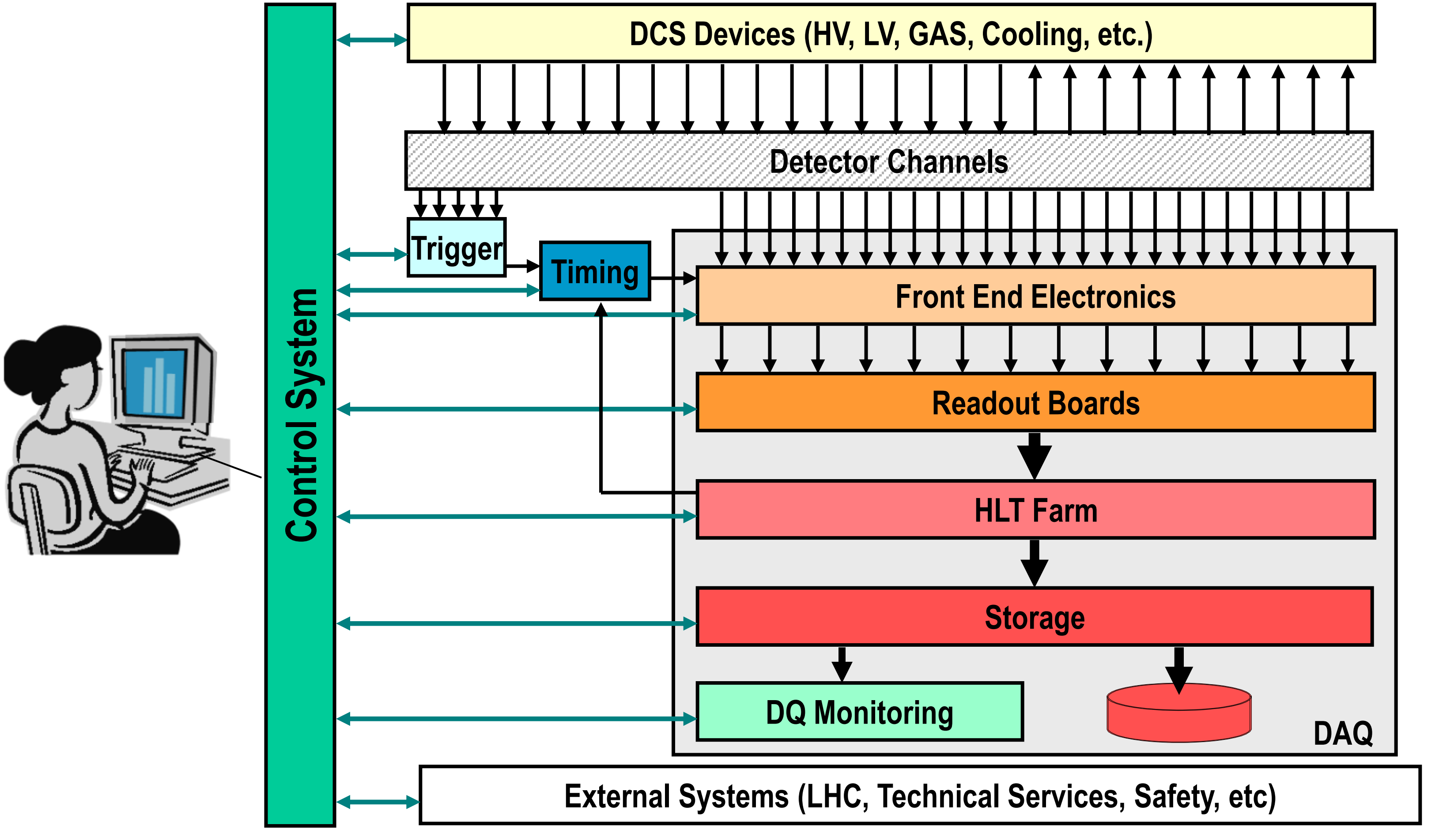
DAQ@LHC

Control Systems

Clara Gaspar, March 2013

Thanks to: Franco Carena, Vasco Chibante Barroso, Giovanna Lehmann Miotto, Hannes Sakulin and Andrea Petrucci for their input

Control System Scope



Control System Tasks

- **Configuration**
  - Selecting which components take part in a certain “Activity”
  - Loading of parameters (according to the “Activity”)
- **Control core**
  - Sequencing and Synchronization of operations across the various components
- **Monitoring, Error Reporting & Recovery**
  - Detect and recover problems as fast as possible
    - Monitor operations in general
    - Monitor Data Quality
- **User Interfacing**
  - Allow the operator to visualize and interact with the system

Some Requirements

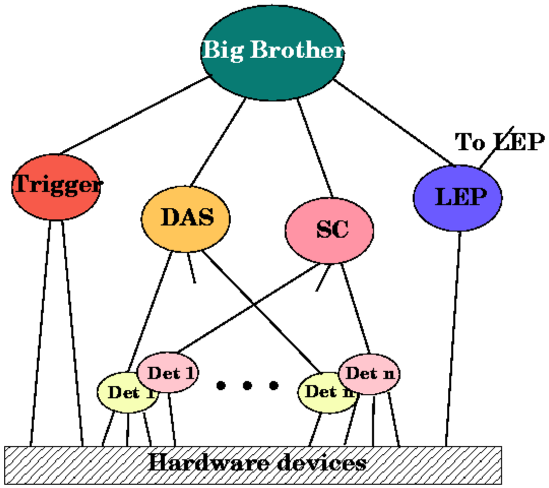
- **Large number of devices/IO channels**
  - Need for Distributed Hierarchical Control
    - De-composition in Systems, sub-systems, … , Devices
    - Maybe: Local decision capabilities in sub-systems
- **Large number of independent teams and very different operation modes**
  - Need for Partitioning Capabilities (concurrent usage)
- **High Complexity & (few) non-expert Operators**
  - Need for good Diagnostics tools and if possible Automation of:
    - Standard Procedures
    - Error Recovery Procedures
  - And for Intuitive User Interfaces

History

- **None of this is new…**
  - Ex.: Both Aleph and Delphi Control Systems:
    - Were Distributed & Hierarchical Systems
    - Implemented Partitioning
    - Were highly Automated
    - Were operated by few shifters:
      - ALEPH: 2 (Shift Leader, Data Quality)
      - DELPHI: 3 (Run Control, Slow Control, Data Quality)

History

- LEP Experiments



The DELPHI experiment control system, 1995

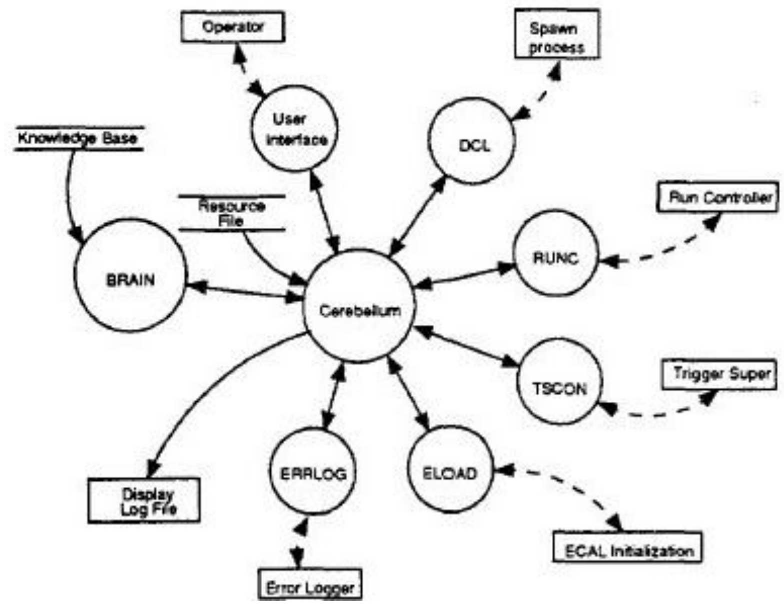


Figure 2 DEXPERT overall design

Applying object oriented, real time, and expert system techniques to an automatic read-out error recovery in the Aleph data acquisition system, 1992

6 Conclusions

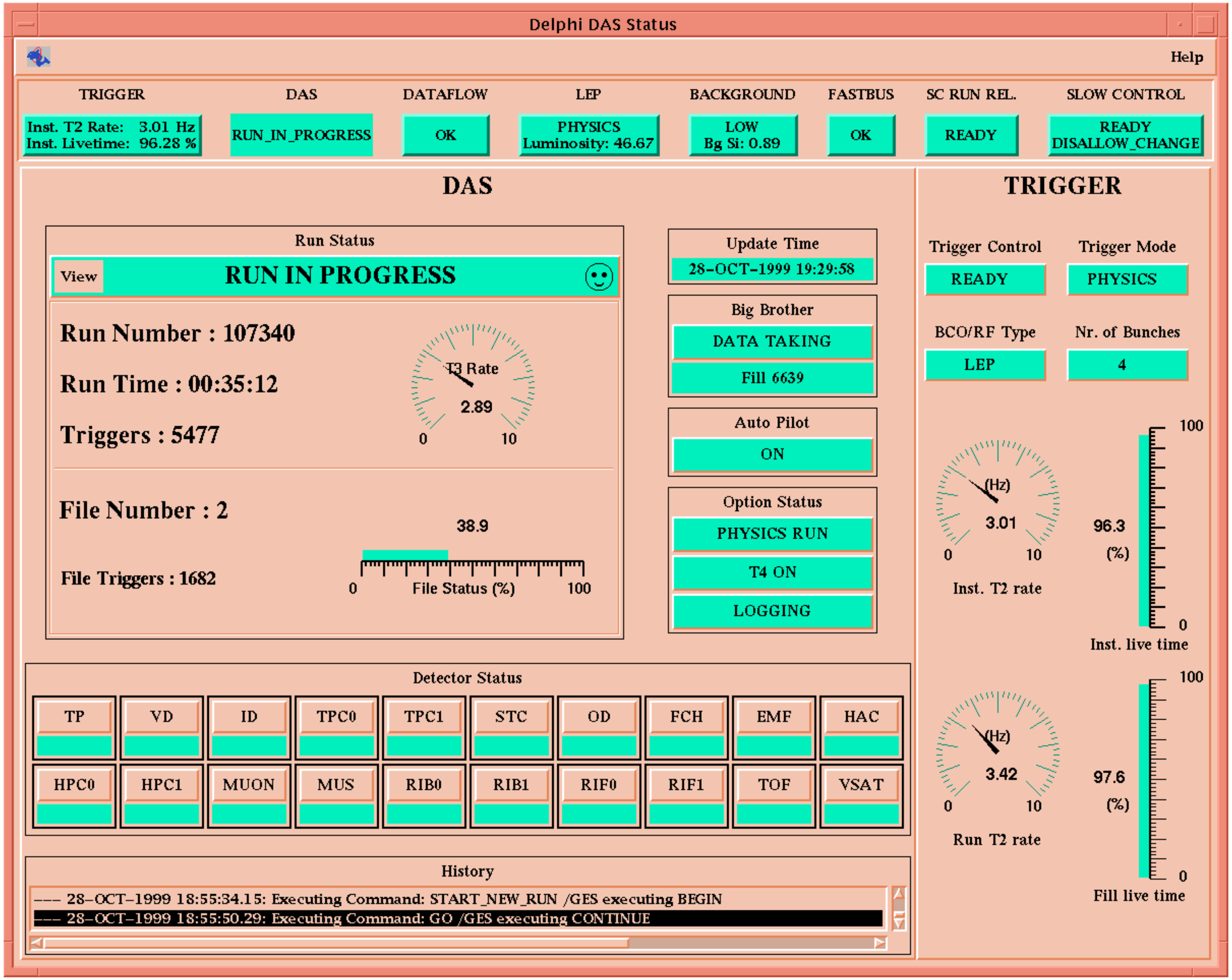
The Control and monitoring System of a large physics experiment involves constraints of operating efficiency, automation and reliability.

The possibility of describing the experiment in terms of objects, using SMI, makes it possible to automate DELPHI operations to a maximum.

3 Conclusions

The DEXPERT program has been in routine use since the beginning of the 1991 data taking period. It handles ~95 % of the possible errors during data-taking very efficiently (i.e. correctly and considerably faster than a human expert). This has permitted us to run the data acquisition system without requiring significant expertise on shift.

History



- Delphi
  Run Control
- Voice Messages
  (recorded)
  most annoying:
  "DAS is not running even though LEP is in Physics"

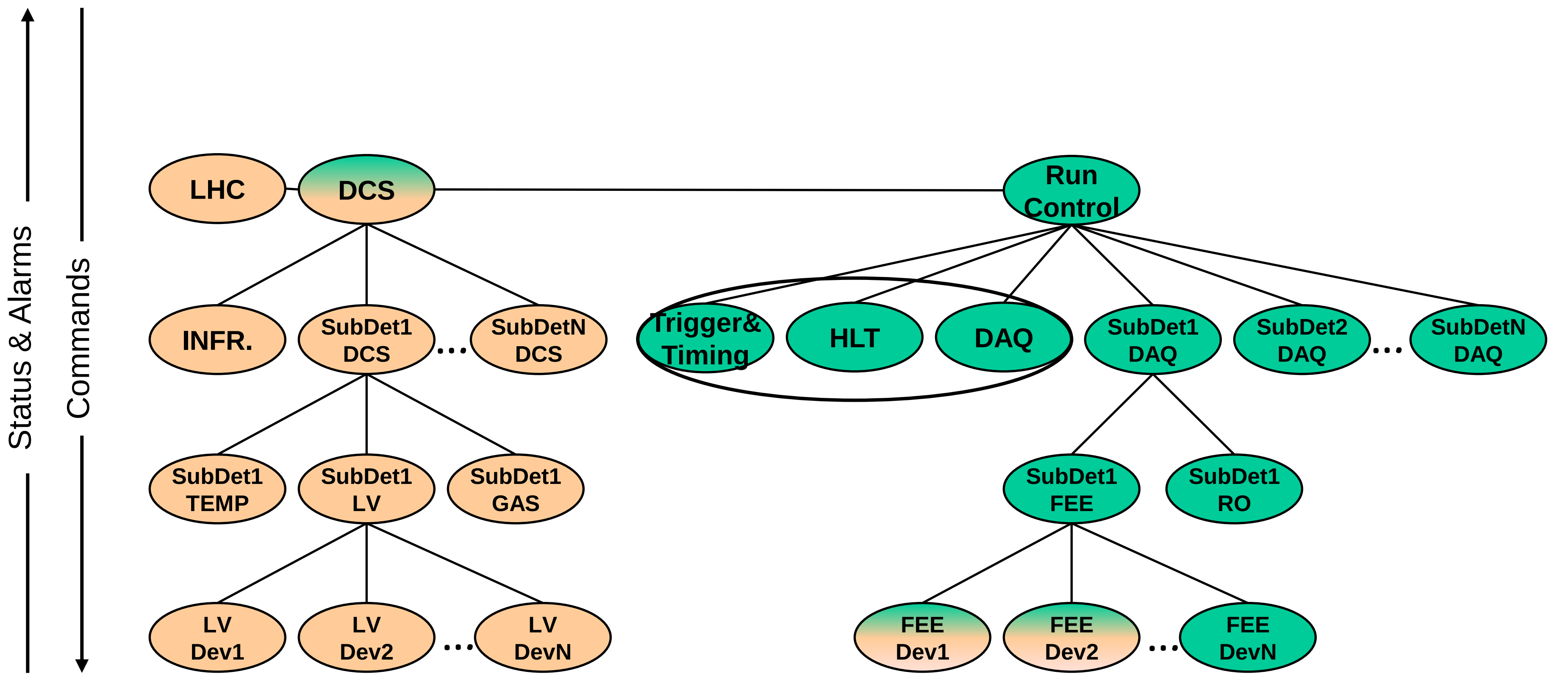
Design Principles

- **Design emphasis per experiment**

Keywords			
ATLAS	Hierarchical	Abstraction	Scalable
CMS	Web Based	Scalable	State Machine Driven
ALICE	Partitioning	Customization	Flexibility
LHCb	Integration	Homogeneity	Automation

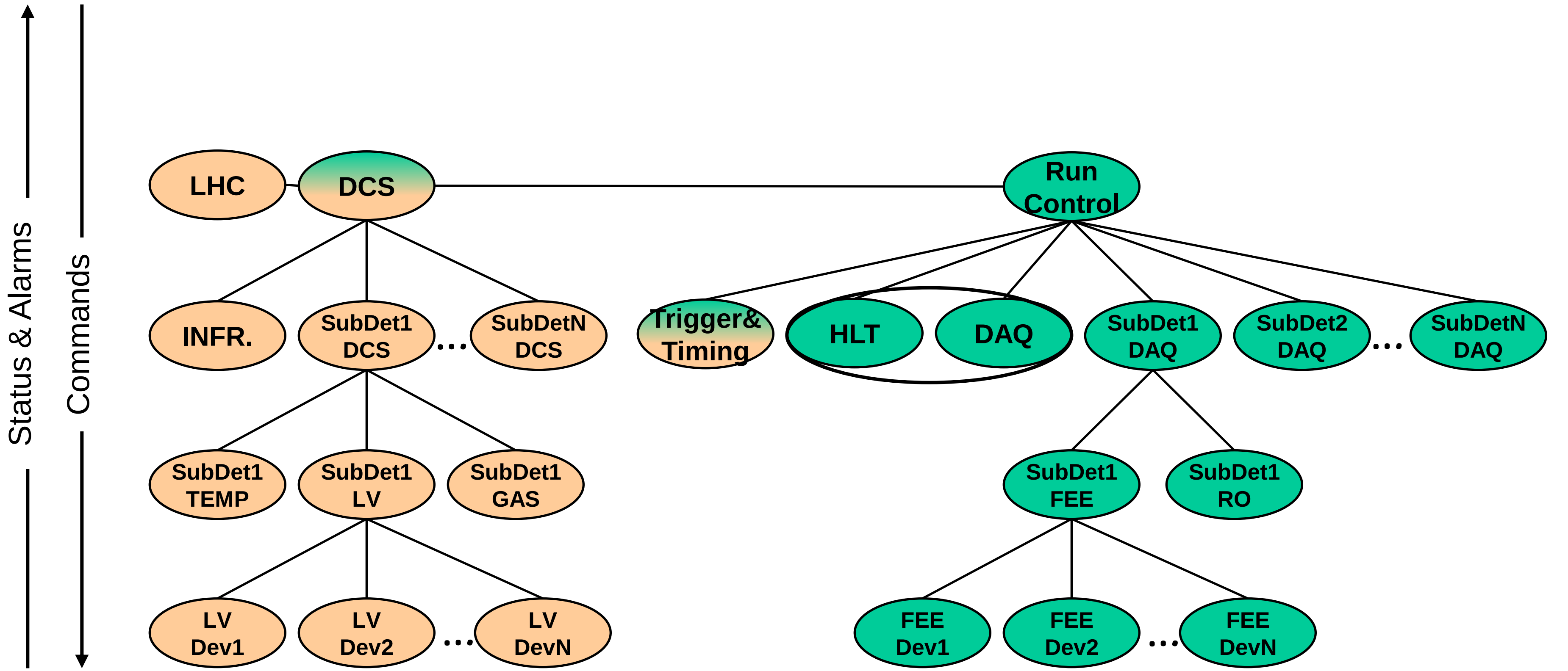
Scope & Architecture

ATLAS



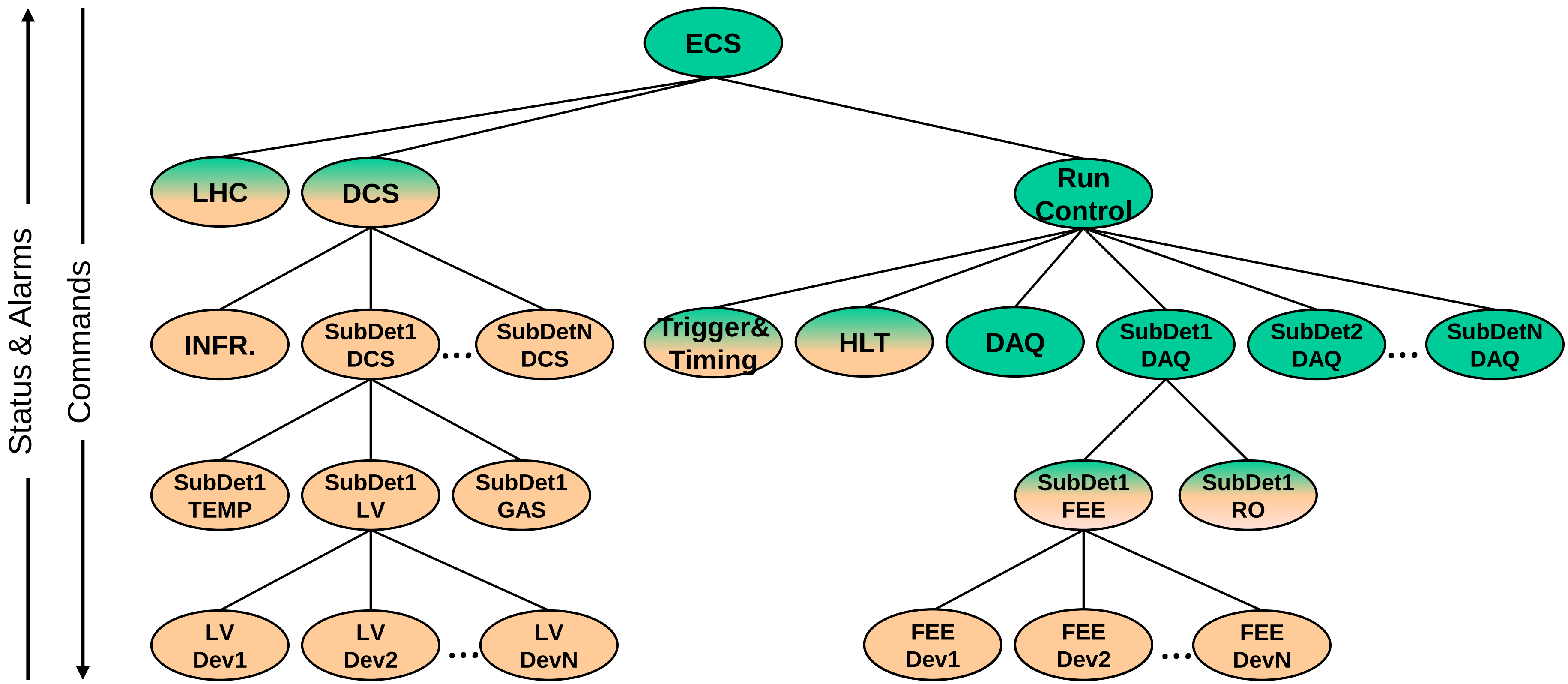
Scope & Architecture

- CMS



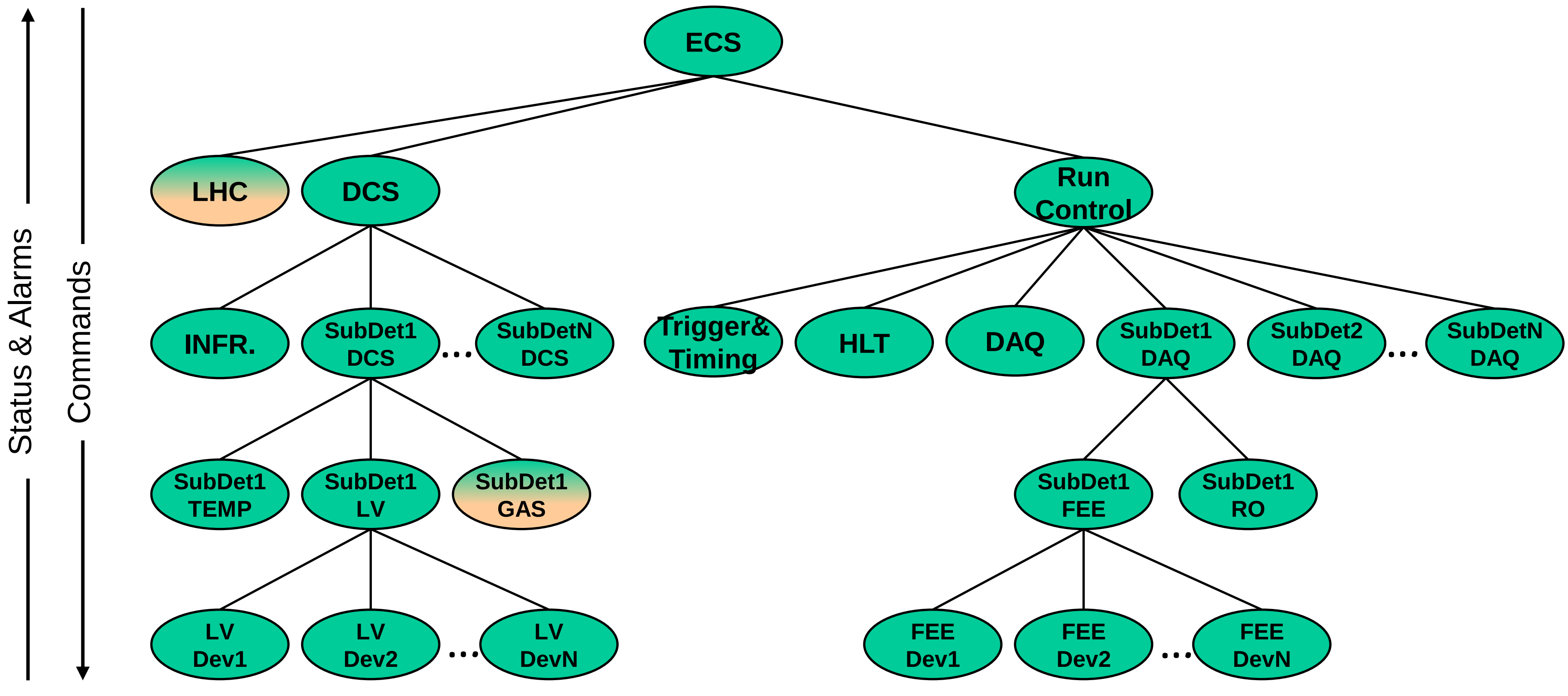
Scope & Architecture

- ALICE



Scope & Architecture

- LHCb



Tools & Components

- **Main Control System Components:**
  - Communications
    - Message Exchange between processes
  - Finite State Machines
    - System Description, Synchronization and Sequencing
  - Expert System Functionality
    - Error Recovery, Assistance and Automation
  - Databases
    - Configuration, Archive, Conditions, etc.
  - User Interfaces
    - Visualization and Operation
  - Other Services:
    - Process Management (start/stop processes across machines)
    - Resource Management (allocate/de-allocate common resources)
    - Logging, etc.

Frameworks

- ALICE
  - DAQ: DATE (Data Acquisition and Test Environment)
    - Comms, FSM, UI, Logging, etc.
- ATLAS
  - Sub-Detectors: RodCrateDAQ;
    - Comms, FSM, UI, Configuration, Monitoring, HW Access libraries
- CMS
  - Control: RCMS (Run Control and Monitoring System)
    - Comms, FSM, UI, Configuration
  - DAQ: XDAQ (DAQ Software Framework)
    - Comms, FSM, UI, Hw Access, Archive
- LHCb
  - Control: JCOP(Joint COntrols Project)/LHCb FW (Dataflow: Gaudi “Online”)
    - Comms, FSM, UI, Configuration, Archive, HW Access, UI builder

Sub-System Integration

- **Sub-Systems use common Framework and tools**
  - ALICE
    - No interface needed: all done centrally
    - Configuration via DCS for most Sub-Detectors
  - ATLAS
    - Interface: FSM + tools & services
    - Configuration via DCS for some Sub-Detectors
  - CMS
    - Interface: FSM in RCMS + XDAQ FW
  - LHCb
    - Interface: FSM + JCOP FW + guidelines (color codes, etc.)

Communications

- **All experiments chose one**
  - ALICE: DIM (mostly within the FSM toolkit)
    - Mostly for Control, some Configuration and Monitoring
  - ATLAS: CORBA (under IPC and IS packages)
    - IPC (Inter Process Comm.) for Control and Configuration
    - IS (Information Service) for Monitoring
  - CMS: Web Services (used by RCMS, XDAQ)
    - RCMS for Control
    - XDAQ for Configuration
    - XMAS (XDAQ Monitoring and Alarm System) for Monitoring
  - LHCb: DIM (and PVSSII, within the JCOP FW)
    - DIM & PVSSII for Control, Configuration and Monitoring

Communications

- **All Client/Server mostly Publish/Subscribe**
  - Difficult to compare (different "paradigms")
    - DIM is a thin layer on top of TCP/IP
    - ATLAS IPC is a thin layer on top of CORBA
      - Both provide a simple API, a Naming Service and error detection & recovery
    - CMS RCMS & XDAQ use WebServices (XML/Soap)
      - Remote Procedure Call (RPC) like, also used as Pub./Sub.
    - ATLAS IS, CMS XMAS and LHCb PVSSII
      - work as data repositories (transient and/or permanent) to be used by clients (UIs, etc.)

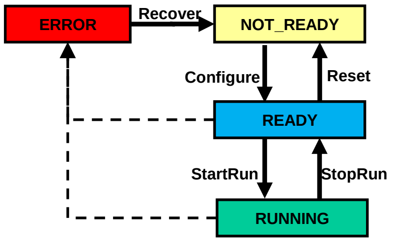
Communications

- **Advantages and drawbacks**
  - DIM
    - ✓ Efficient, Easy-to-use
    - ✗ Home made, old…
  - CORBA
    - ✓ Efficient, Easy-to-use (via the ATLAS API)
    - ✗ Not so popular anymore…
  - WEB Services
    - ✓ Standard, modern protocol
    - ✗ Performance: XML overhead

Finite State Machines

- **All experiments use FSMs**
  - In order to model the system behaviour:
    - For Synchronization, Sequencing, in some cases also for Error Recovery and Automation of procedures
  - ALICE: SMI++
    - FSM for all sub-systems provided centrally (can be different)
  - ATLAS: CHSM -> CLIPS -> C++
    - FSM for all sub-systems provided centrally (all the same)
  - CMS: Java for RCMS, C++ for XDAQ
    - Each sub-system provided their own code (Java/C++)
  - LHCb: SMI++ (integrated in PVSS II)
    - FSM provided centrally, sub-systems modify template graphically

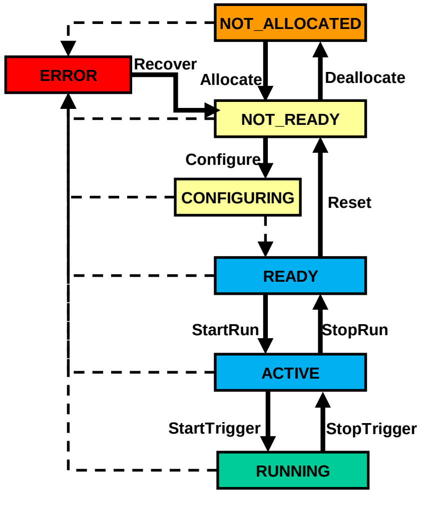
FSM Model Design



- **Two Approaches:**
  - Few, coarse-grained States:
    - Generic actions are sent from the top
      - Each sub-system synchronizes it's own operations to go to the required state
    - The top-level needs very little knowledge of the sub-systems
    - Assumes most things can be done in parallel
  - Many, fine-grained States
    - Every detailed transition is sequenced from the top
    - The top-level needs to know the details of the sub-systems

FSM Definitions

- **Top-level FSM from "ground" to Running**
  - ATLAS
    - None -> Booted -> Initial -> Configured -> Connected -> Running
  - CMS
    - Initial -> Halted -> Configured -> Running (+intermediary states)
  - LHCb
    - Not Allocated -> Not Ready -> Ready -> Active -> Running
  - ALICE
    - Many: 20 to 25 states, 15 to get to Running



Expert System Functionality

- **Several experiments saw the need…**
  - Approach:
    - "We are in the mess how do we get out of it?"
    - No Learning…
- **Used for:**
  - Advising the Shifter
    - ➡ ATLAS, CMS
  - Automated Error Recovery
    - ➡ ATLAS, LHCb, ALICE (modestly)
  - Completely Automate Standard Operations
    - ➡ LHCb

Expert System Functionality

- **ATLAS**
  - Uses CLIPS for Error Recovery
    - Common and distributed, domain specific, rules
    - Used by experts only, sub-system rules on request
  - Uses Esper for “Shifter Assistant”
    - Centralised, global “Complex Event Processing”
    - Moving more towards this approach…
- **CMS**
  - Uses Perl for “DAQ Doctor”
    - “Rules” are hardcoded by experts

Expert System Functionality

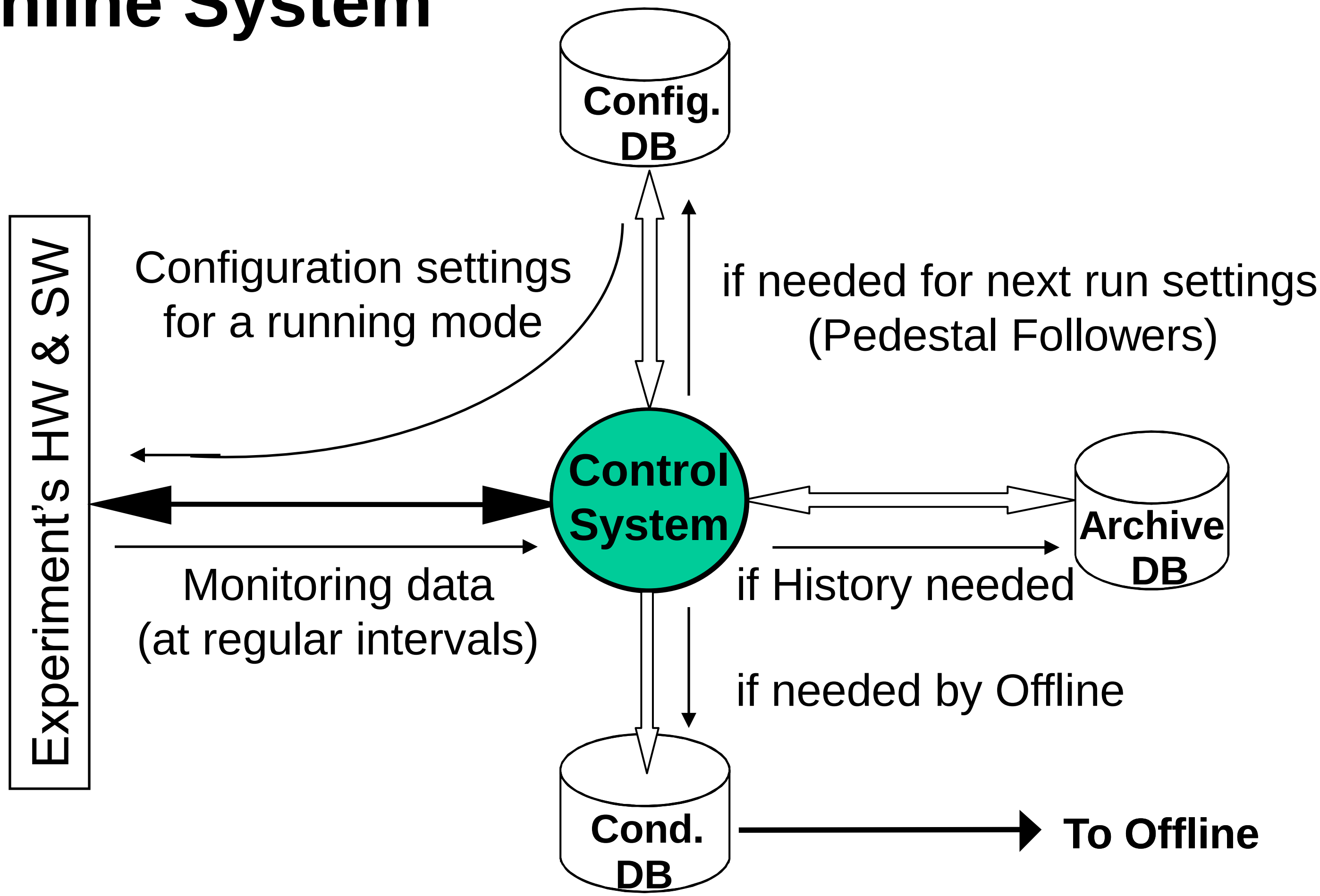
- **LHCb**
  - Uses SMI++ for everything
    - Distributed FSM and Rule based system
    - Used by sub-systems for local error recovery and automation
    - Used by central team for top-level rules integrating various sub-systems
- **ALICE**
  - Uses SMI++ too
    - Some error recovery (only few specific cases)

Expert System Functionality

- **Decision Making, Reasoning, Approaches**
  - Decentralized
    - Bottom-up: Sub-systems react only to their "children"
      - In an event-driven, asynchronous, fashion
    - Distributed: Each Sub-System can recover its errors
      - Normally each team knows better how to handle local errors
    - Hierarchical/Parallel recovery
    - Scalable
  - Centralized
    - All "rules" in the same repository

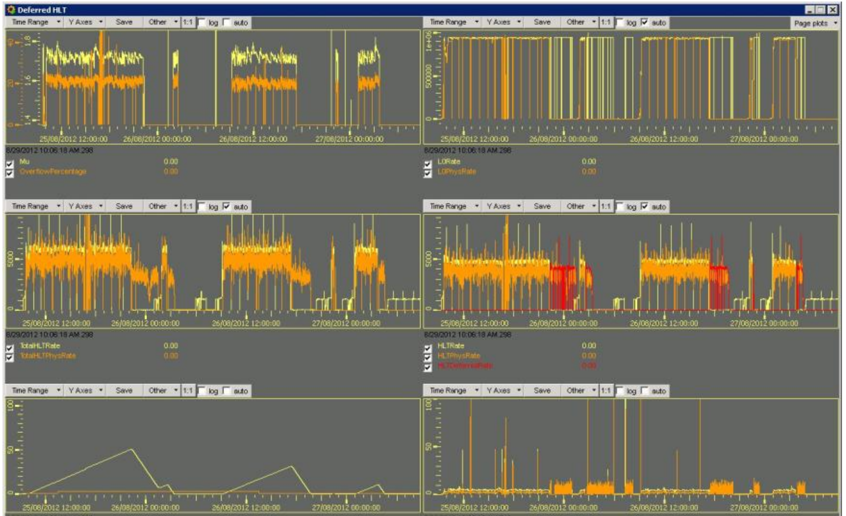
Online Databases

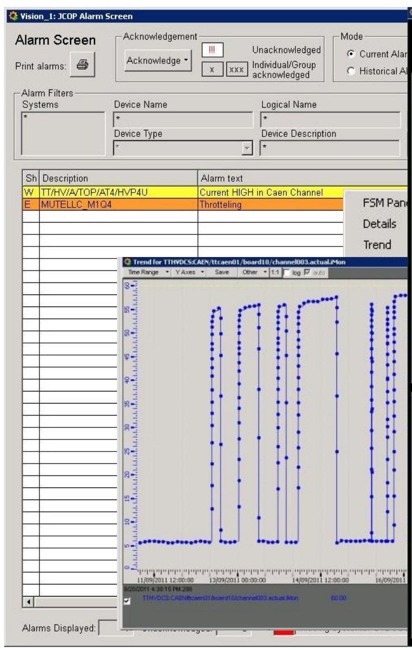
- Three main logical Database concepts in the Online System

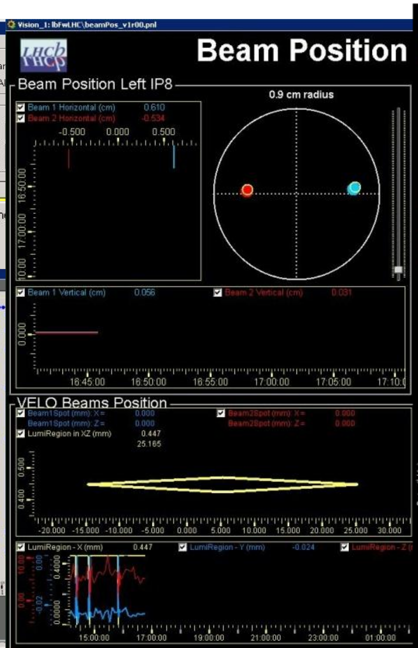


User Interfacing

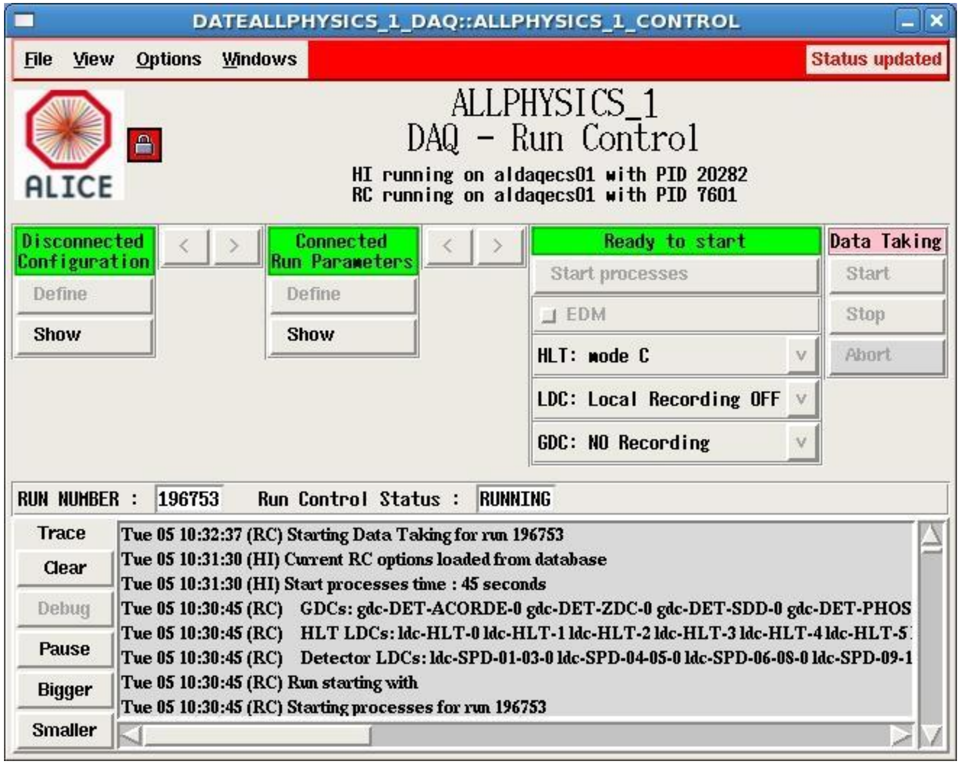
- **Types of User Interfaces**
  - Alarm Screens and/or Message Displays
  - Monitoring Displays
  - Run Control

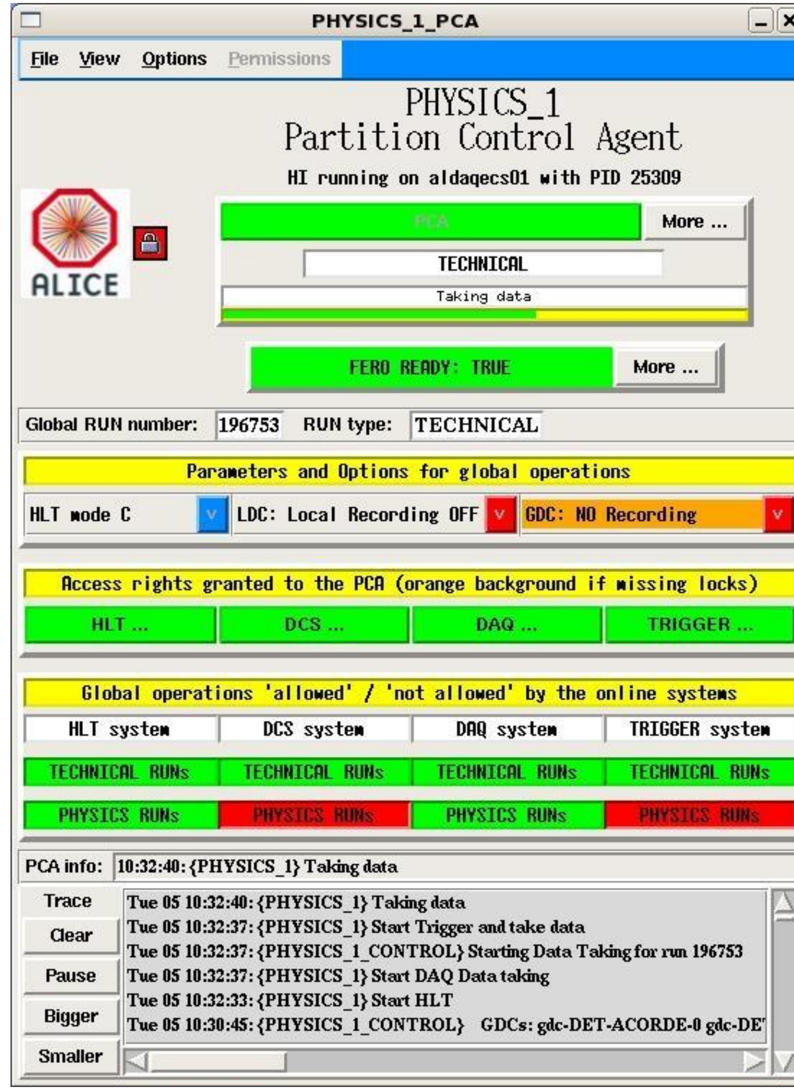






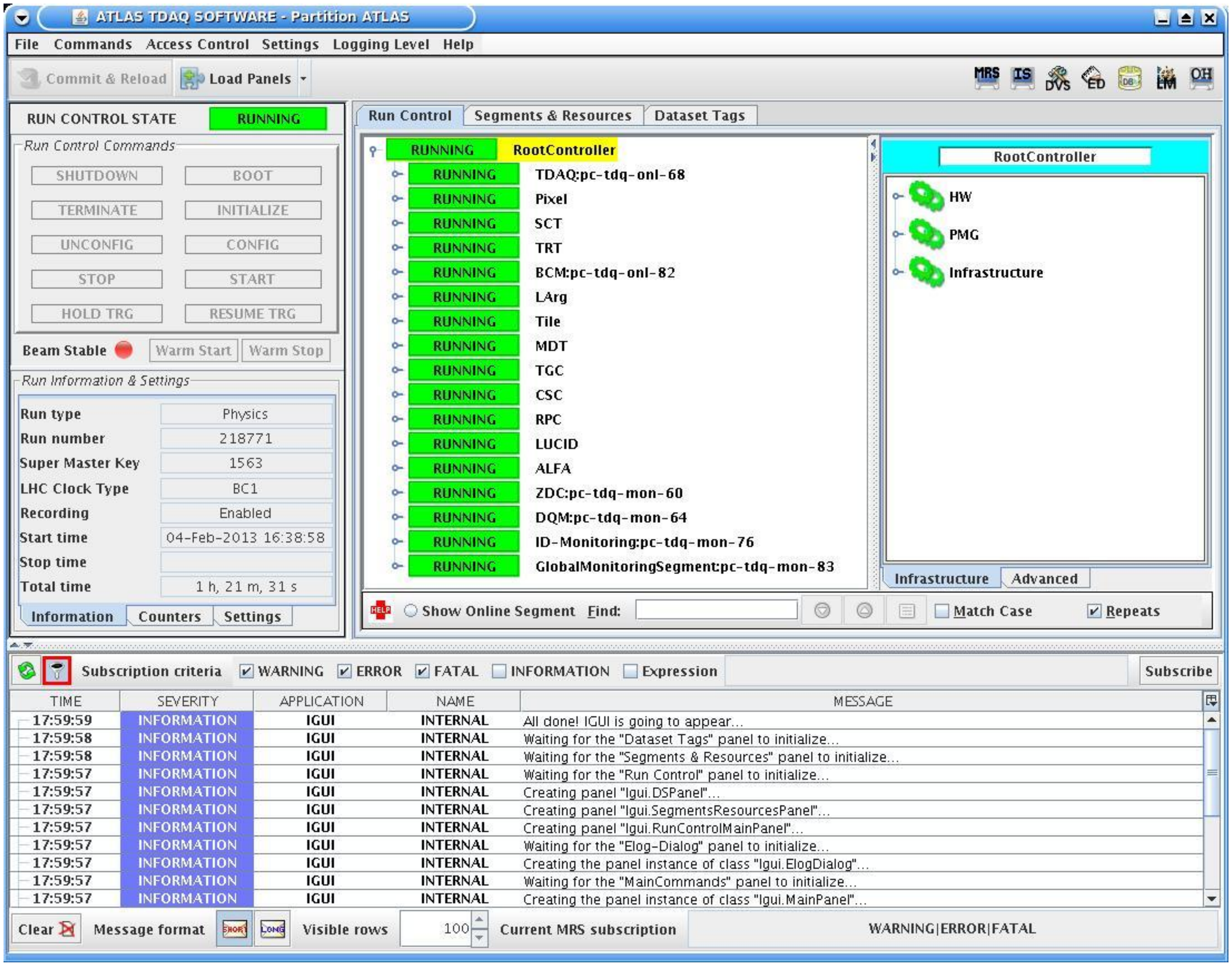
ALICE Run Control+ECS





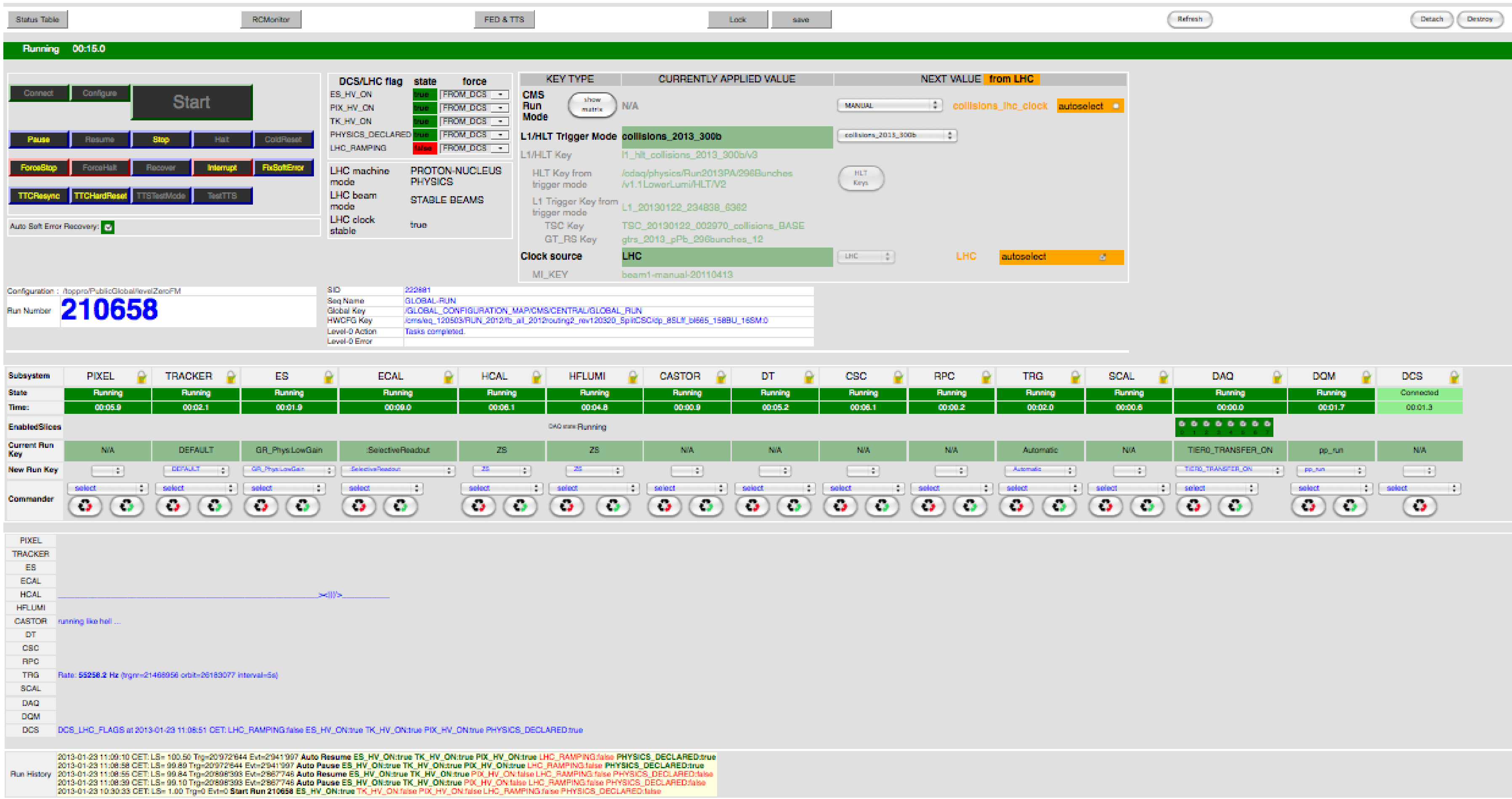
- Implemented in Tcl/Tk

ATLAS Run Control



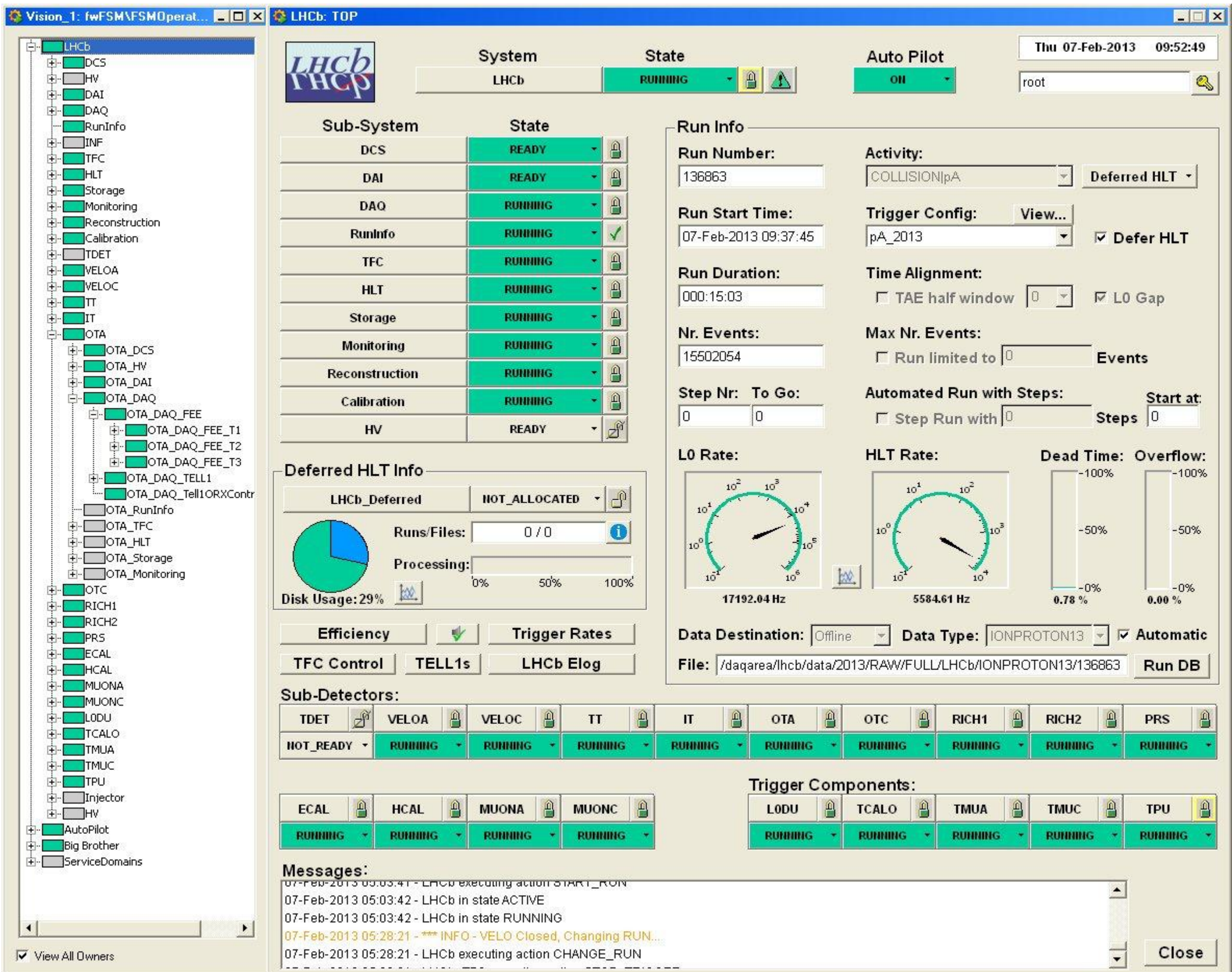
- IGUI
  - Java
  - Modular:
  - Sub-systems can add their panels

CMS Run Control



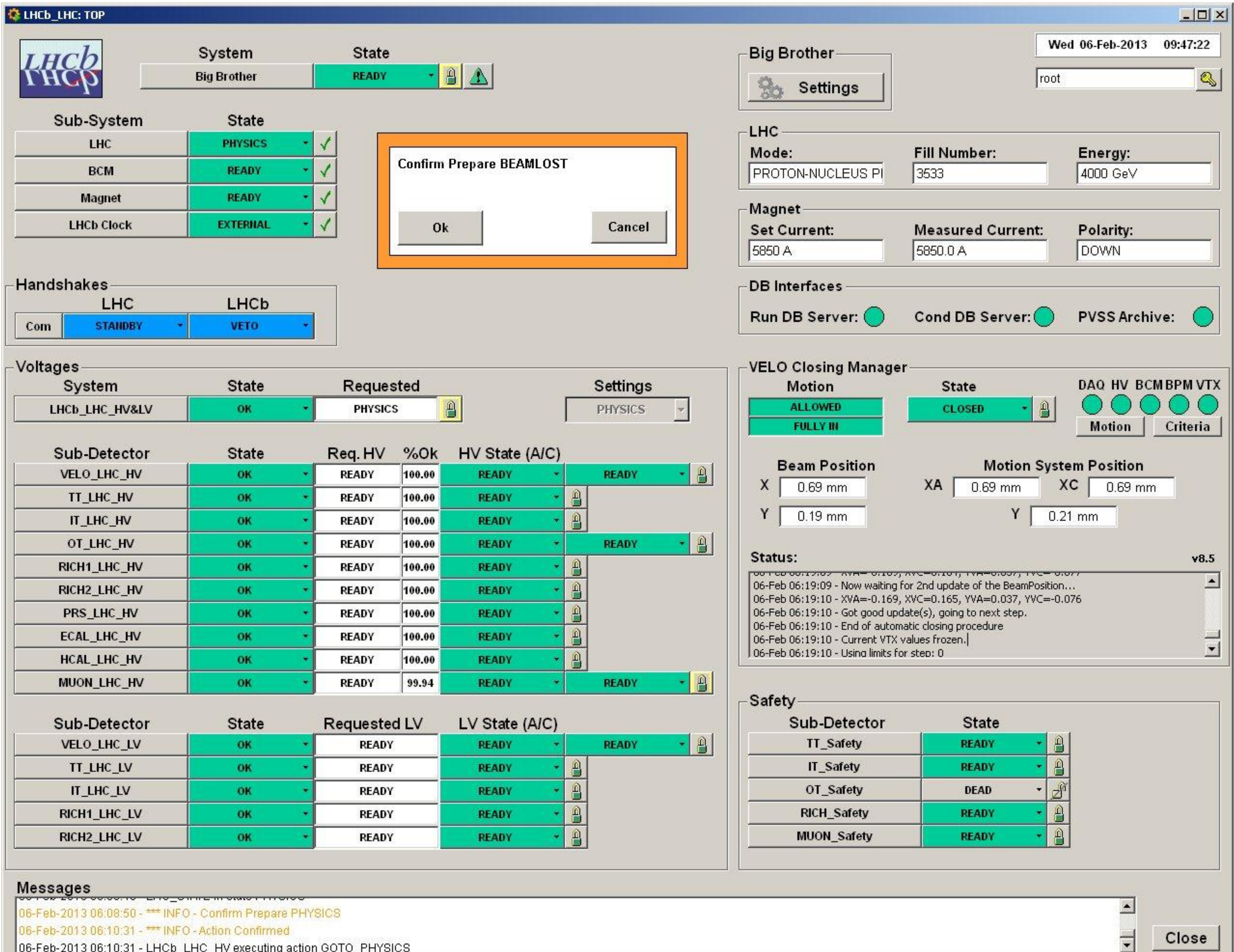
- Web tools: JavaScript+HTML
- Also LabView Monitoring

LHCb Run Control



- JCOP FW
  - PVSS+SMI++
  - Like all UIs at all levels
  - Using a very convenient Graphic Editor
  - Very easy to modify

LHCb Big Brother



- Used by operator to confirm automated actions
- Voice Messages (synthesized)

Access Control

- **What are we trying to achieve?**
  - Protect against "evil" attacks?
    - We are "authenticated" and inside a protected network…
  - Avoid Mistakes?
    - Mistakes are often done by experts…
  - Traceability and Accountability…
- **Types of Protection**
  - At UI Level
    - LHCb, ALICE, CMS: Very basic: ownership (CMS: also Role based views)
  - Everywhere (at message reception)
    - ATLAS: Role Based

Size and Performance

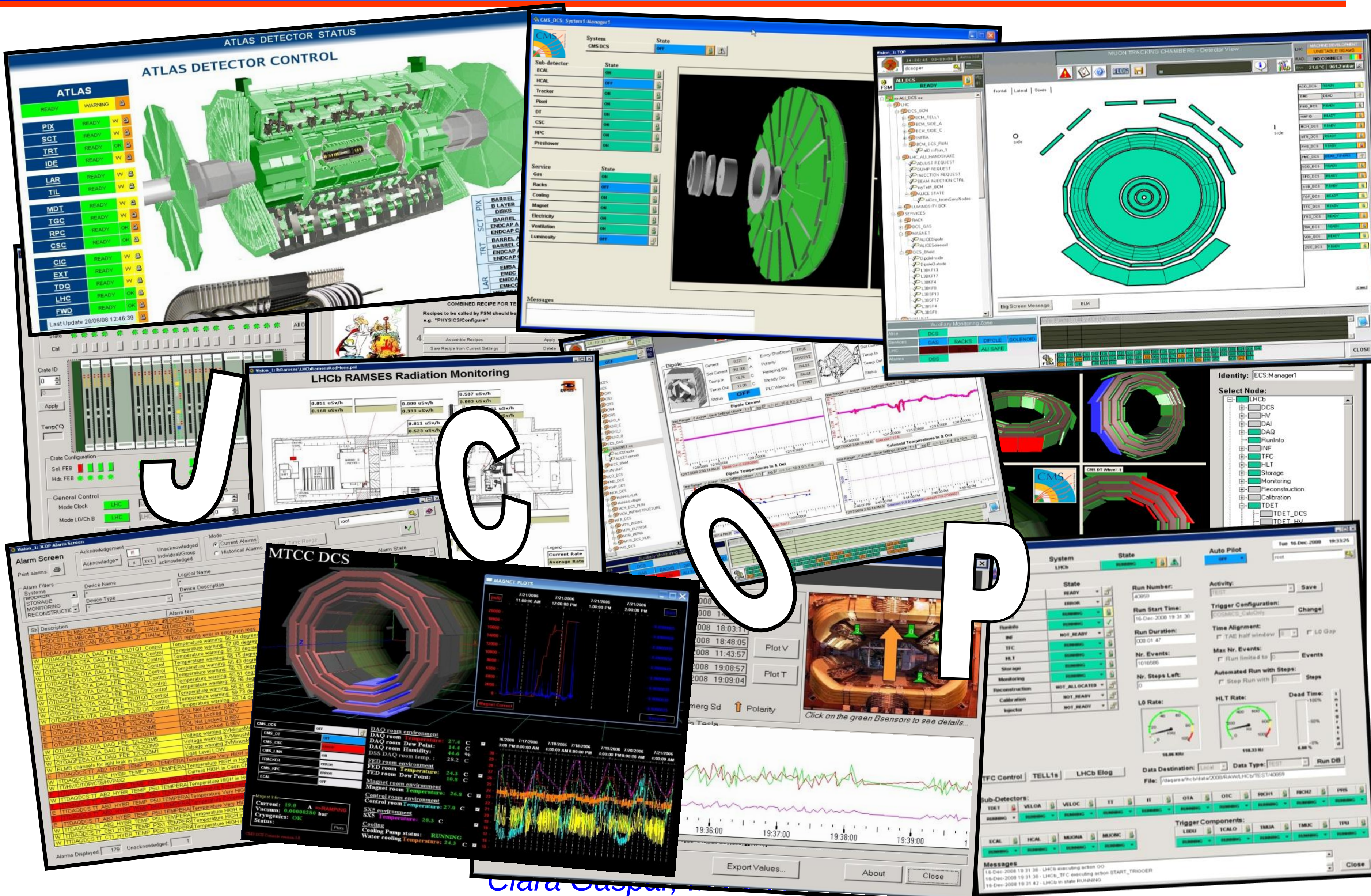
- **Size of the Control Systems (in PCs)**
  - ALICE: 1
  - ATLAS: 32 + some for HLT control
  - CMS: 12
  - LHCb: ~50 DAQ + ~50 HLT + ~50 DCS
- **Some Performance numbers**

	ALICE	ATLAS	CMS	LHCb
Cold Start to Running (min.)	5	5	3	4
Stop/Start Run (min.)	6	2	1	1
Fast Stop/Start (sec.)	-	<10	<10	<10
DAQ Inefficiency (%)	1	<1	<1	<1

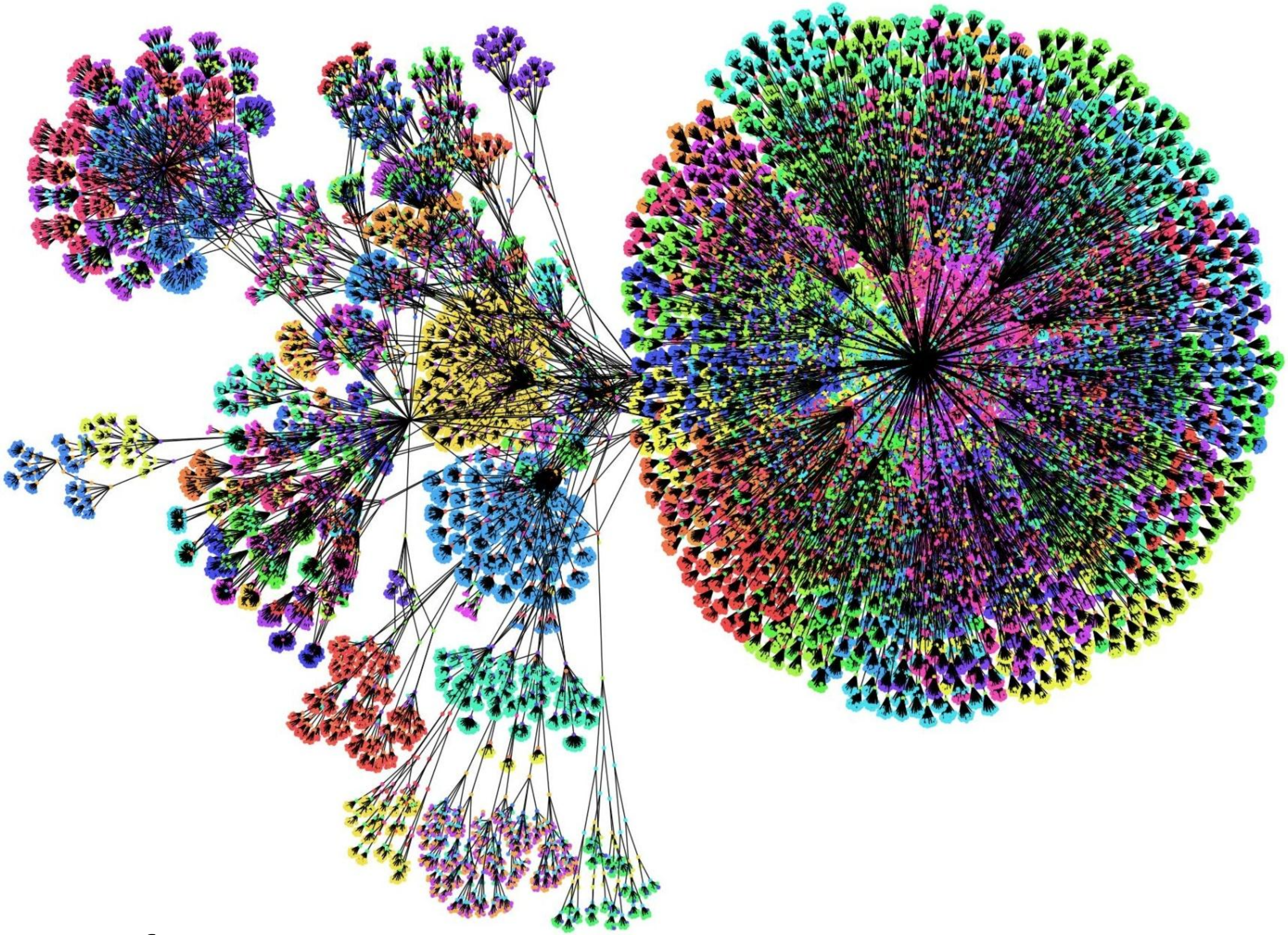
Operations

- **Experiment Operations**
  - Shifters:
    - ALICE: 4 (SL, DCS, RC, DQ+HLT)
    - ATLAS: 8 (SL, DCS, RC, TRG, DQ, ID, Muon, Calo)
    - CMS: 5 (SL, DCS, RC, TRG, DQ)
    - LHCb: 2 (SL, DQ)
  - Ex.: Start of Fill sequence
    - In general HV automatically handled
    - Run Control Shifter manually Configures/Starts the Run (apart from LHCb)

Detector Control Systems



LHCb Control System



- **Courtesy of CMS DCS Team**