Experiment Control@LHC

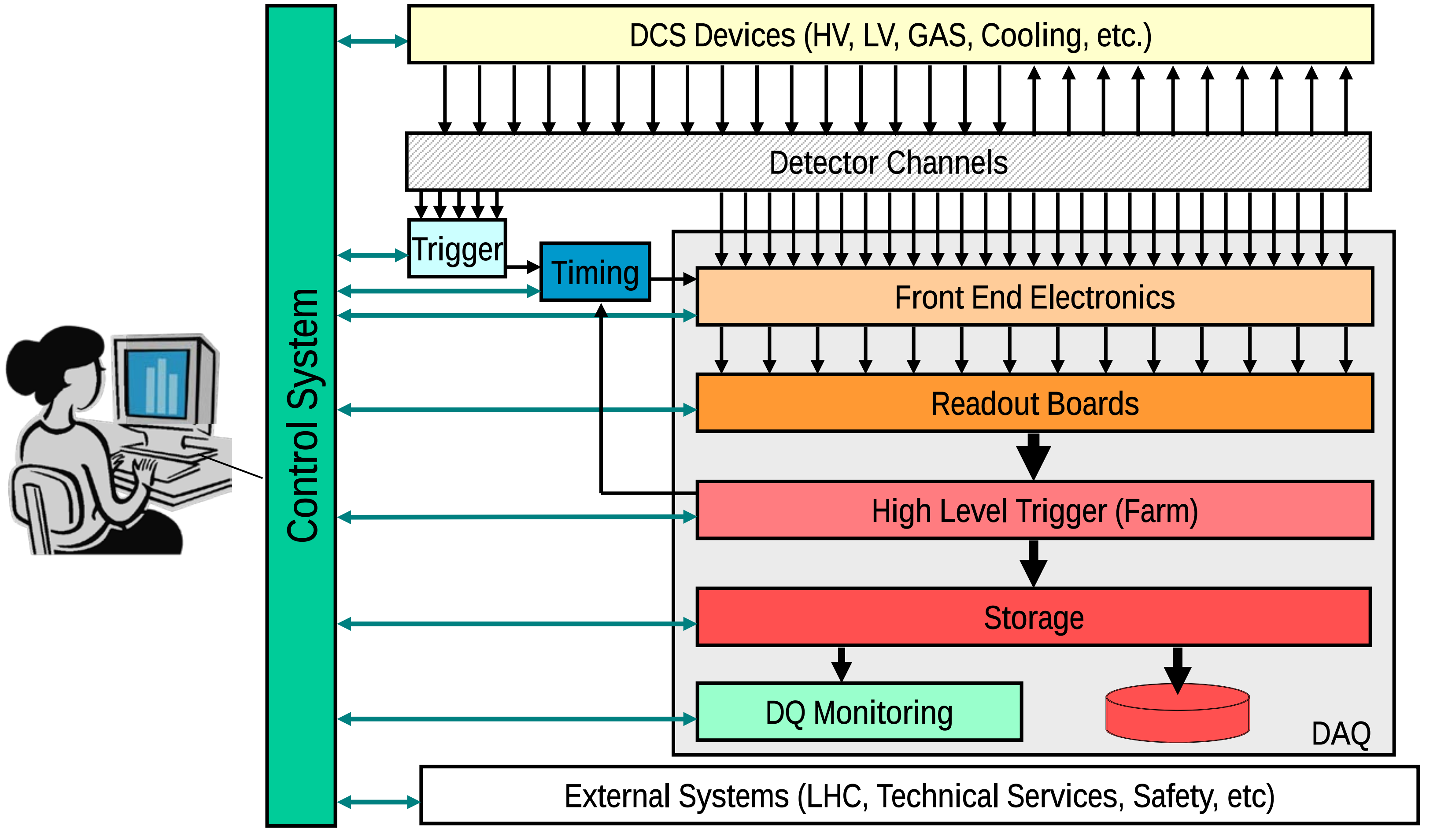
An Overview

Clara Gaspar, October 2013

Many thanks to the colleagues in the four experiments and the EN/ICE group, in particular:
ALICE: Franco Carena, Vasco Chibante Barroso (DAQ), Andre Augustinus (DCS)
ATLAS: Giovanna Lehmann Miotto (DAQ), Stefan Schlenker (DCS)
CMS: Hannes Sakulin, Andrea Petrucci (DAQ), Frank Glege (DCS)
JCOP: Fernando Varela Rodriguez

The LHC Experiments

Control System Scope



Control System Tasks

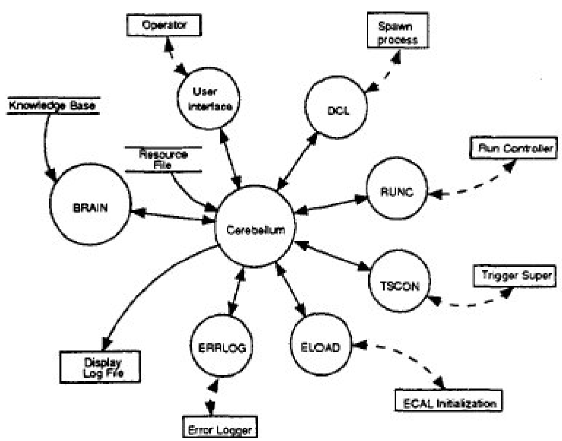
- **Configuration**
  - Selecting which components take part in a certain “Activity”
  - Loading of parameters (according to the “Activity”)
- **Control core**
  - Sequencing and Synchronization of operations across the various components
- **Monitoring, Error Reporting & Recovery**
  - Detect and recover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

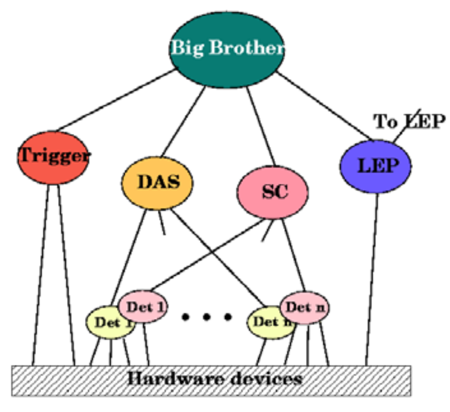
Some Requirements

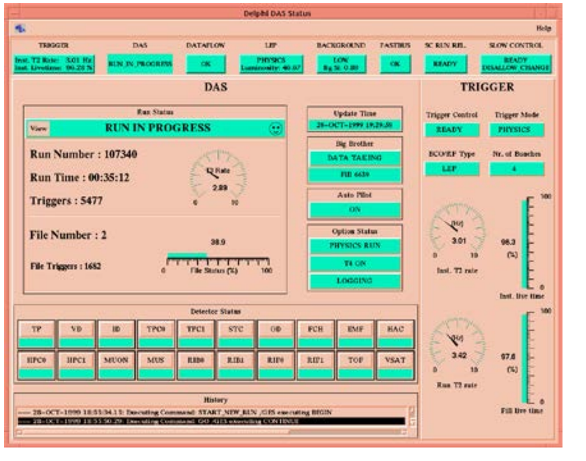
- **Large number of devices/IO channels**
  - Need for Distributed Hierarchical Control
    - De-composition in Systems, sub-systems, … , Devices
    - Maybe: Local decision capabilities in sub-systems
- **Large number of independent teams and very different operation modes**
  - Need for Partitioning 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
- **+ Scalability, reliability, maintainability, etc.**

History

- **None of this is really new…**
  - Ex.: At LEP (in the 80s/90s) both ALEPH and DELPHI Control Systems:
    - Were Distributed & Hierarchical Systems, implemented Partitioning, were highly Automated and were operated by few shifters:
      - ALEPH: 2 (Shift Leader, Data Quality)
      - DELPHI: 3 (Run Control, Slow Control, Data Quality)







- **ALEPH: DEXPERT**
- **DELPHI: Big Brother**

LHC Exp. Commonalities

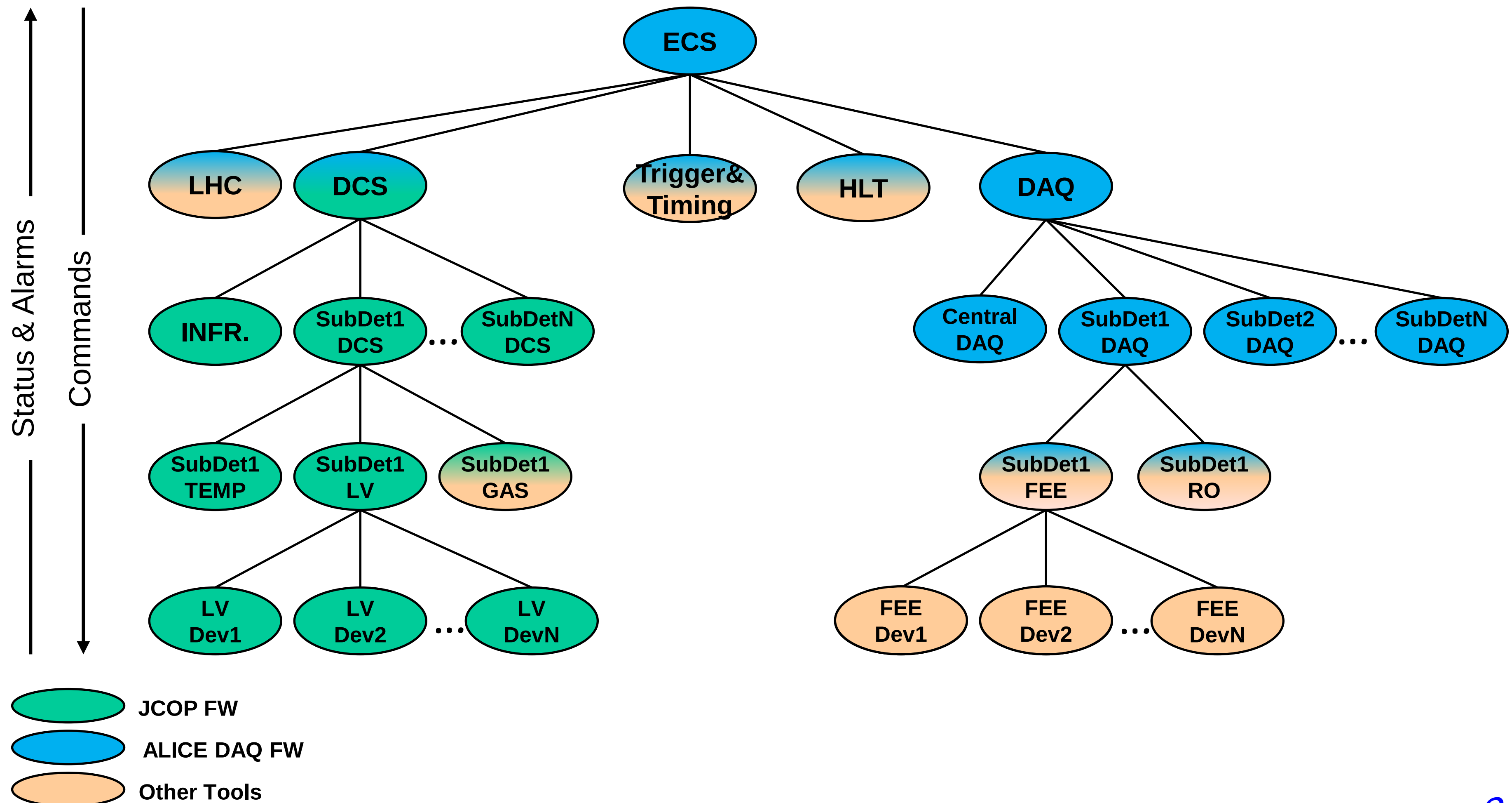
- **Joint COntrols Project (JCOP)**
  - A common project between the four LHC experiments and a CERN Control Group (IT/CO -> EN/ICE)
  - Mandate (1997/1998):
    - "Provide a common DCS for all 4 experiments in a resource effective manner"
    - "Define, select and/or implement as appropriate the architecture, framework and components required to build the control system"
  - Scope:
    - DCS - Detector Control System (at least)
  - Main Deliverable:
    - JCOP Framework (JCOP FW)
  - ➡ Major Success! Still active

LHC Exp. Differences

- **Basically the Control of everything else:**
  - DAQ, Trigger, etc. -> Run Control
- **Design Principles**
  - Similar requirements, different emphasis, for example:
    - ATLAS: Large detector -> Scalability
    - CMS: Many users -> Web Based
    - LHCb: Few shifters -> Integration, homogeneity
    - ALICE: Many sub-detectors -> Customization, Flexibility

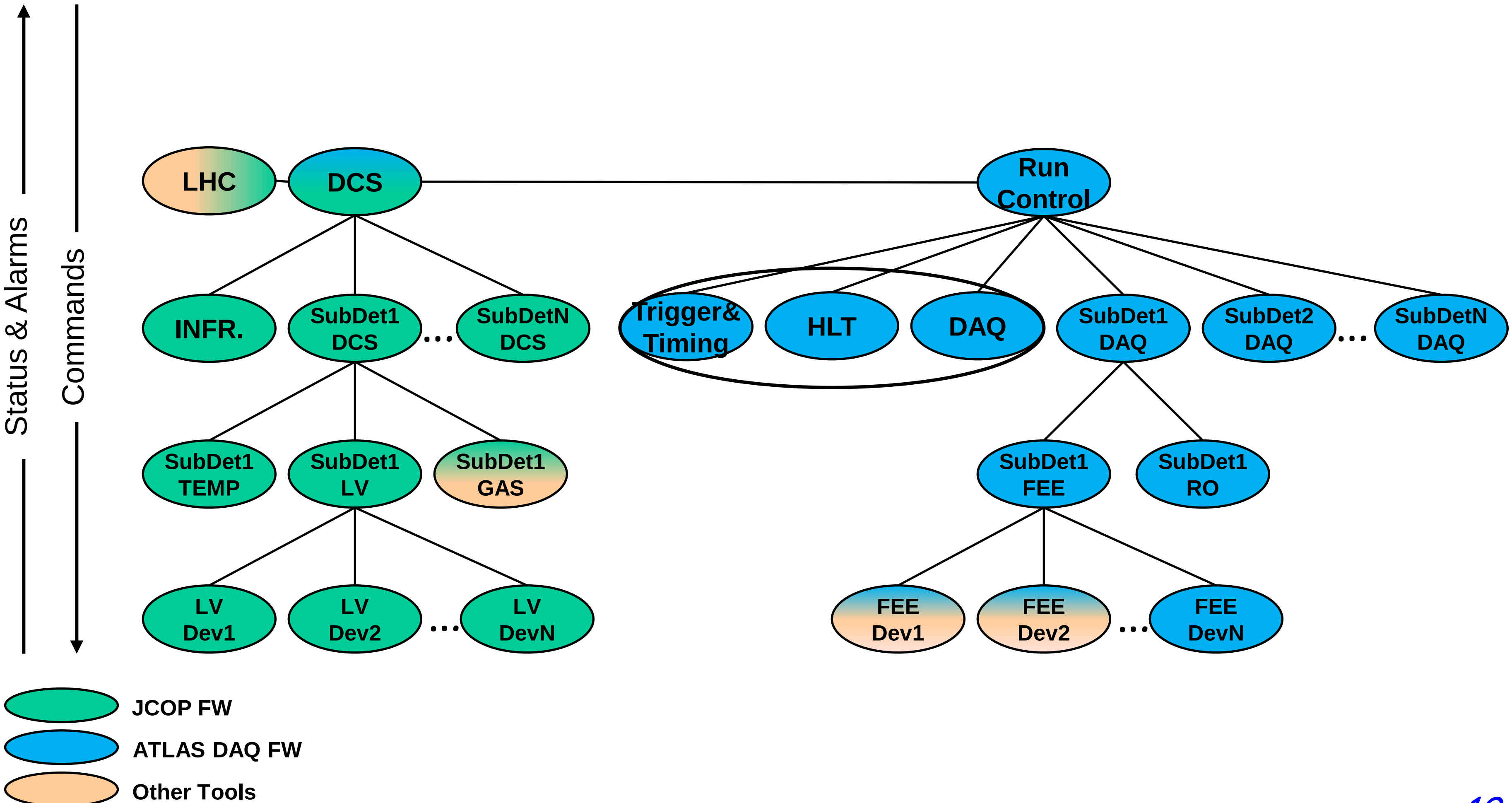
Architecture & Scope

ALICE



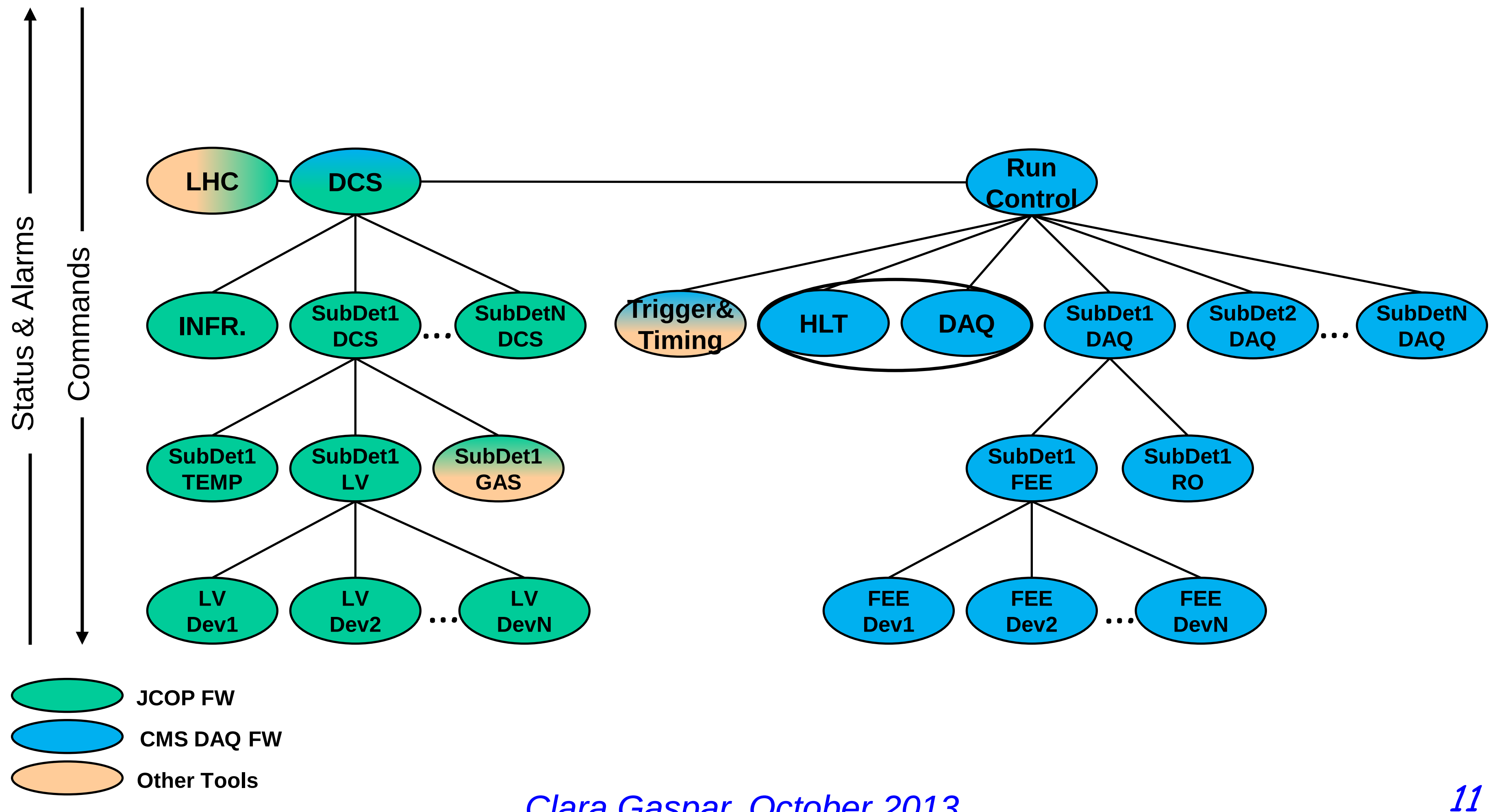
Architecture & Scope

- ATLAS



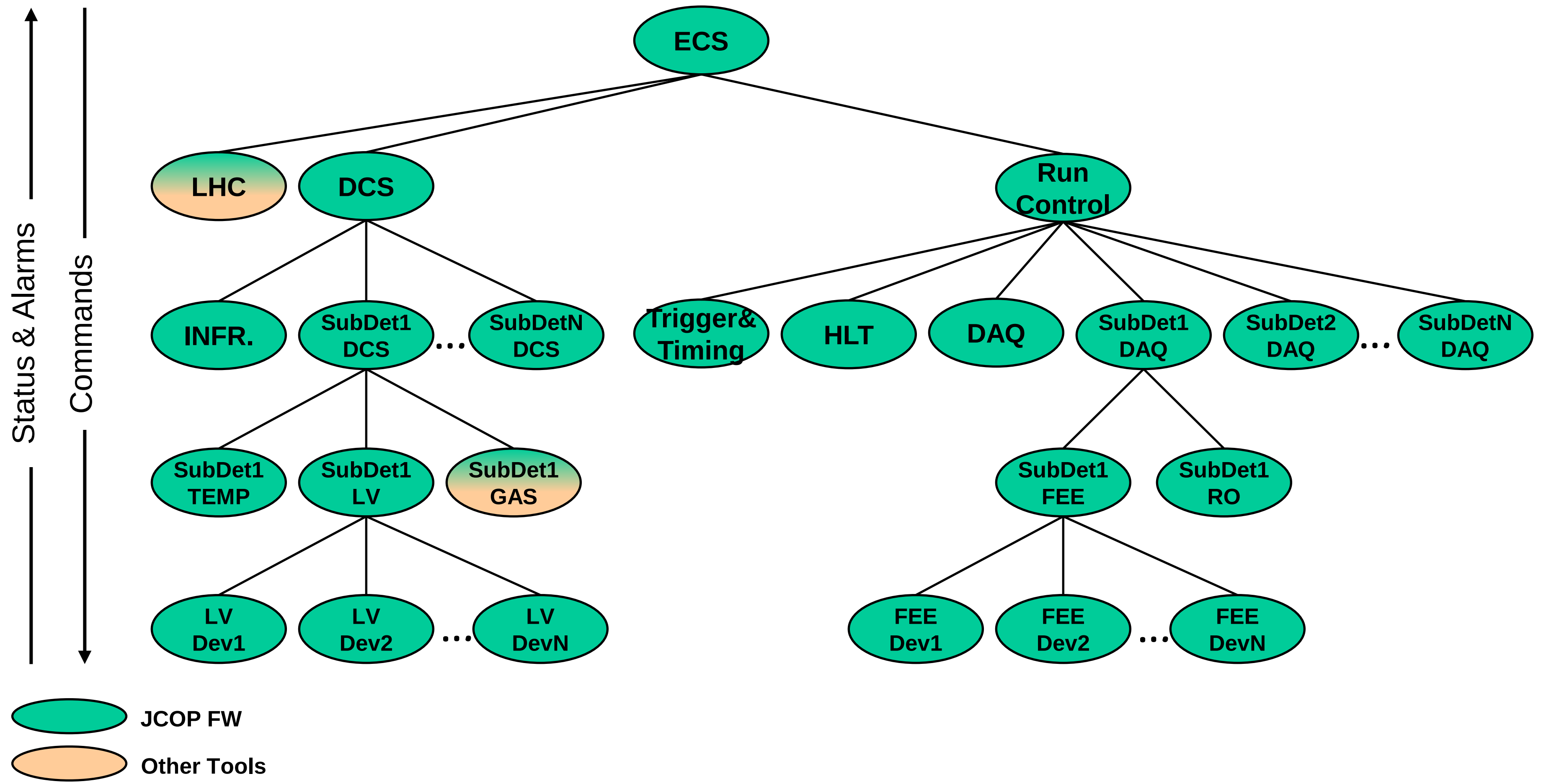
Architecture & Scope

▮ CMS



Architecture & Scope

- **LHCb**



Tools & Components

- **Main Control System Components:**
  - Communications
    - Device Access and 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

- JCOP FW (All Experiments DCSs + LHCb)
  - Based on SCADA System PVSS II (Now Siemens WinCC-OA)
    - Comms, FSM, UI, UI builder, Configuration, Archive, HW Access, Alarms, etc. (also guidelines and ready-made components for many types of equipment)
- ALICE
  - DAQ: DATE (Data Acquisition and Test Environment)
    - Comms, FSM, UI, Logging, etc.
- ATLAS
  - DAQ: Set of high-level Services + Sub-Detector FW: RodCrateDAQ
    - Comms, FSM, UI, Configuration, Monitoring, + HW Access libraries
- CMS
  - Control: RCMS (Run Control and Monitoring System)
    - Comms, FSM, UI, Configuration, Archive
  - DAQ: XDAQ (DAQ Software Framework)
    - Comms, FSM, UI, Hw Access, Archive

Communications

- **Each experiment chose one**
  - ALICE DAQ: DIM (mostly within the FSM toolkit)
    - Mostly for Control, some Configuration and Monitoring
  - ATLAS DAQ: CORBA (under IPC and IS packages)
    - IPC (Inter Process Comm.) for Control and Configuration
    - IS (Information Service) for Monitoring
  - CMS DAQ: Web Services (used by RCMS, XDAQ)
    - RCMS for Control
    - XDAQ for Configuration
    - XMAS (XDAQ Monitoring and Alarm System) for Monitoring
  - LHCb & DCSs: PVSSII+drivers+DIM (within JCOP FW)
    - PVSSII offers many drivers (most used in DCS is OPC)
    - LHCb DAQ: DIM 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 recovery
    - CMS RCMS & XDAQ use WebServices (XML/Soap)
      - Remote Procedure Call (RPC) like, also used as Pub./Sub.
    - OPC is based on Microsoft's OLE, COM and DCOM

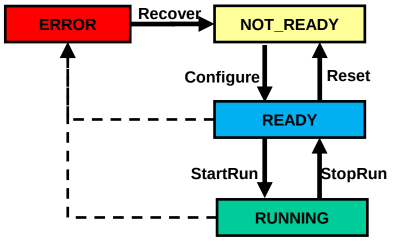
	✓	✗
DIM	**Efficient, Easy to use**	**Home made**
CORBA	**Efficient, Easy to use (via API)**	**Not so popular anymore**
Web Services	**Standard, modern protocol**	**Performance: XML overhead**
OPC	**Industry Standard**	**Only Windows (-> OPC UA)**

  - ATLAS IS, CMS XMAS and PVSS II in the DCSs and LHCb
    - work as data repositories (transient and/or permanent) to be used by clients (UIs, etc.)

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 DAQ: SMI++
    - FSM for all sub-systems provided centrally (can be different)
  - ATLAS DAQ: CHSM -> CLIPS -> C++
    - FSM for all sub-systems provided centrally (all the same)
  - CMS DAQ: Java for RCMS, C++ for XDAQ
    - Each sub-system provided specific transition code (Java/C++)
  - LHCb & DCSs: SMI++ (integrated in the JCOP FW)
    - LHCb: FSM provided centrally, sub-systems can 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
    - ➡ Followed by most experiments (both DAQ & DCS)
      - Ex: CMS States from "ground" to Running:
        Initial -> Halted -> Configured -> Running
  - Many, fine-grained States
    - Every detailed transition is sequenced from the top
    - The top-level knows the details of the sub-systems
    - ➡ Followed by ALICE DAQ (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, CMS, LHCb, ALICE (modestly)
  - Completely Automate Standard Operations
    - ➡ LHCb, and within the DCSs

Expert System Functionality

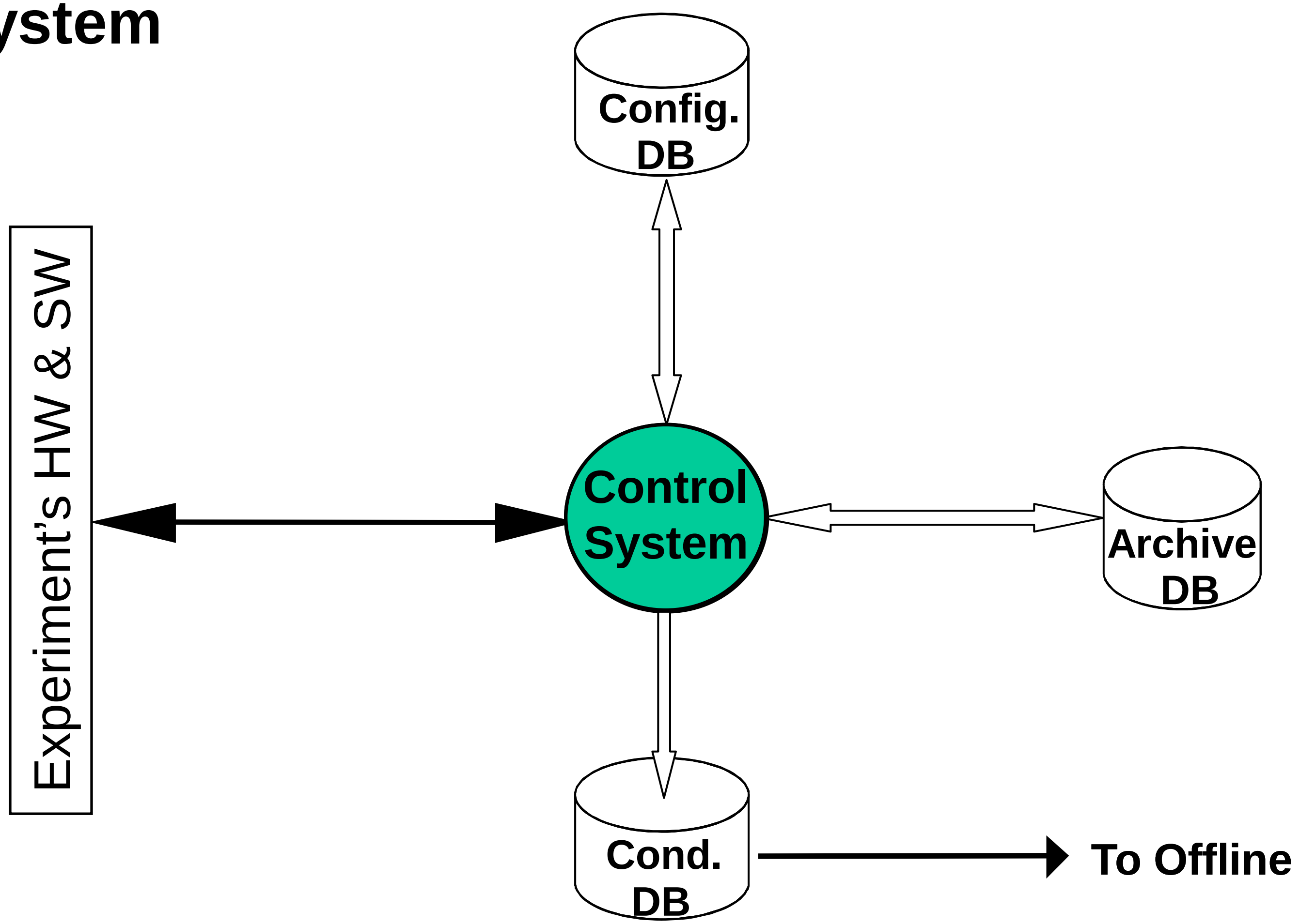
- **ATLAS**
  - CLIPS for Error Recovery
    - Central and distributed, domain specific, rules
    - Used by experts only, sub-system rules on request
  - Esper for “Shifter Assistant”
    - Centralised, global “Complex Event Processing”
      - ➡ Moving more towards this approach…
- **CMS**
  - Java (within RCMS) for Error recovery and Automation
  - Perl for “DAQ Doctor”
    - “Rules” are hardcoded by experts
- **LHCb & DCSs** (within JCOP FW) **+ ALICE** (in standalone)
  - SMI++ for Error Recovery and Automation
    - Distributed FSM and Rule based system
    - Sub-systems use it for local rules, central team for top-level rules

Expert System Functionality

- **Decision Making, Reasoning, Approaches**
  - Decentralized (Ex.: SMI++)
    - 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 how to handle local errors
    - Hierarchical/Parallel recovery
    - Scalable
  - Centralized (Ex.: Esper)
    - All “rules” in the same repository, one central engine

Online Databases

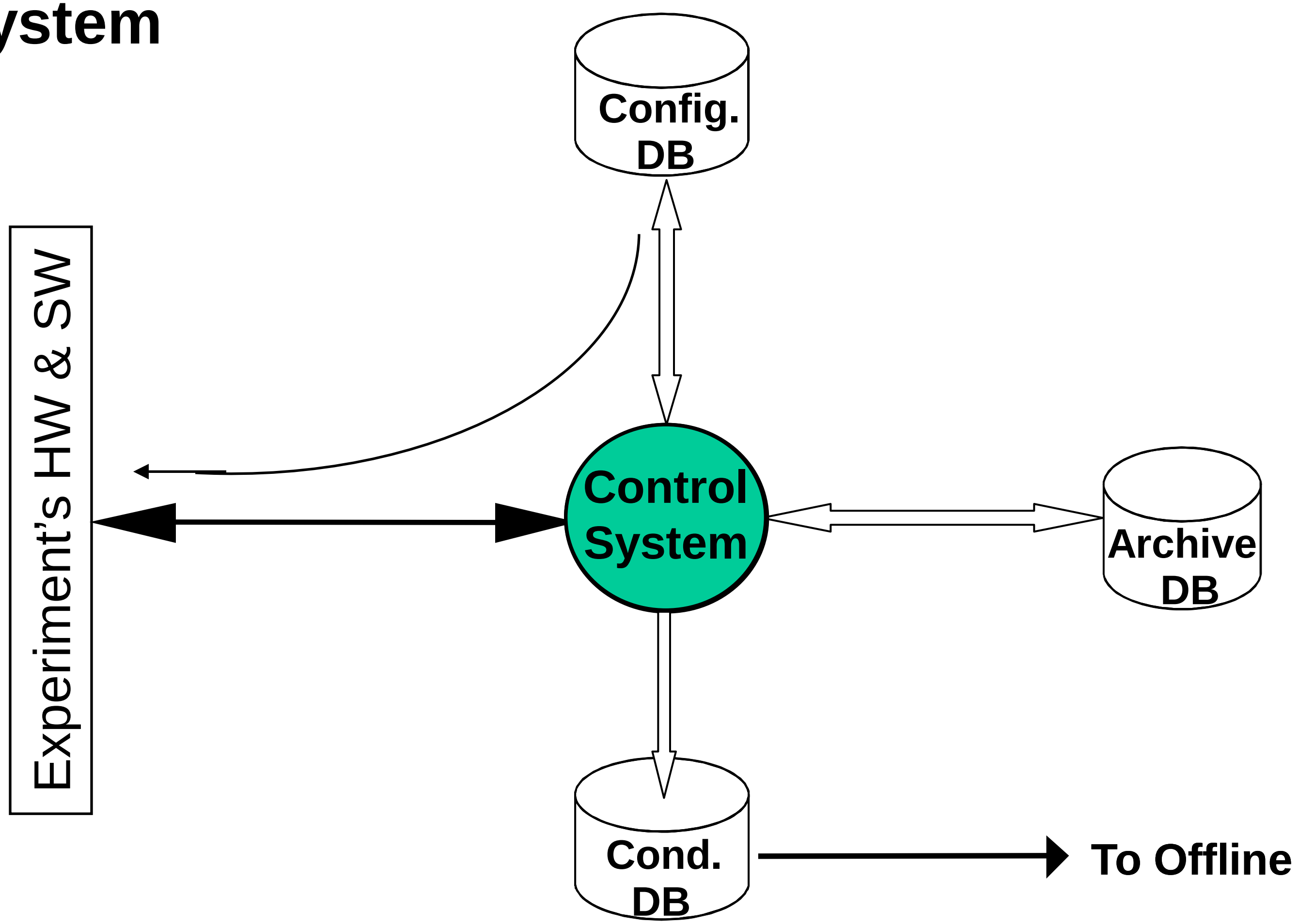
- Three main logical Database concepts in the Online System



➡ But naming, grouping and technology can be different in the different experiments…

Online Databases

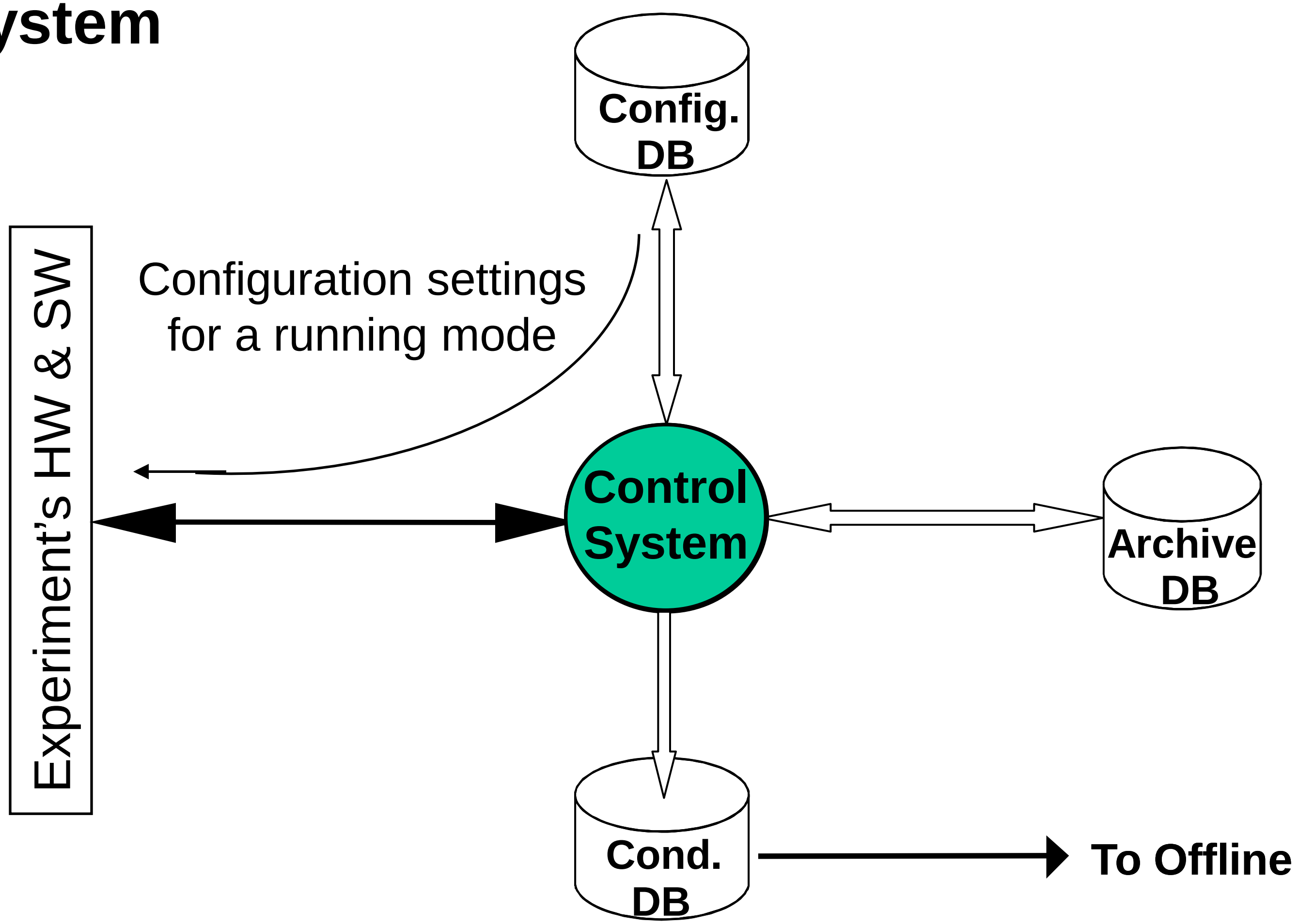
- Three main logical Database concepts in the Online System



- ➡ But naming, grouping and technology can be different in the different experiments…

Online Databases

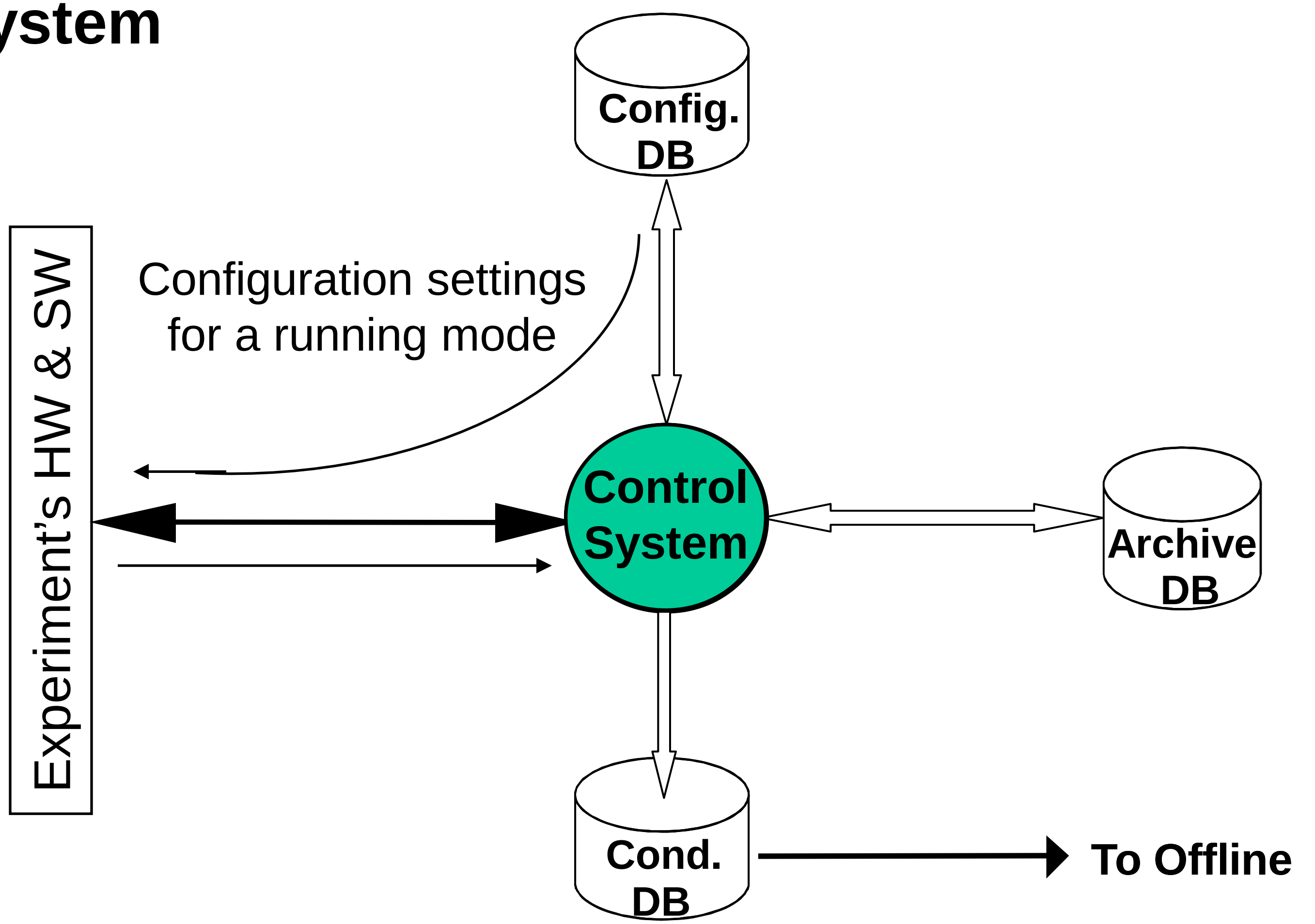
- **Three main logical Database concepts in the Online System**



- **But naming, grouping and technology can be different in the different experiments…**

Online Databases

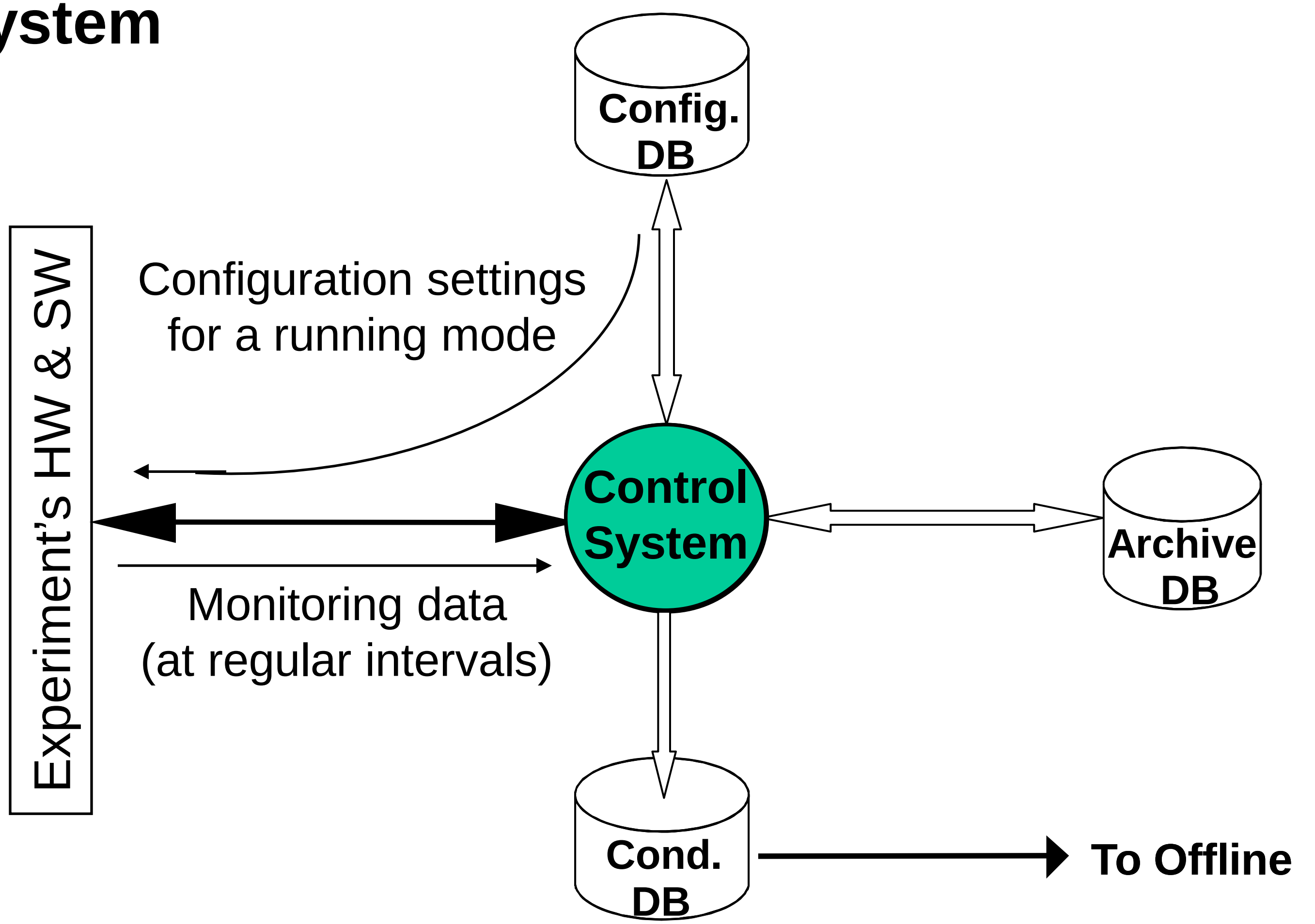
- **Three main logical Database concepts in the Online System**



- **But naming, grouping and technology can be different in the different experiments…**

Online Databases

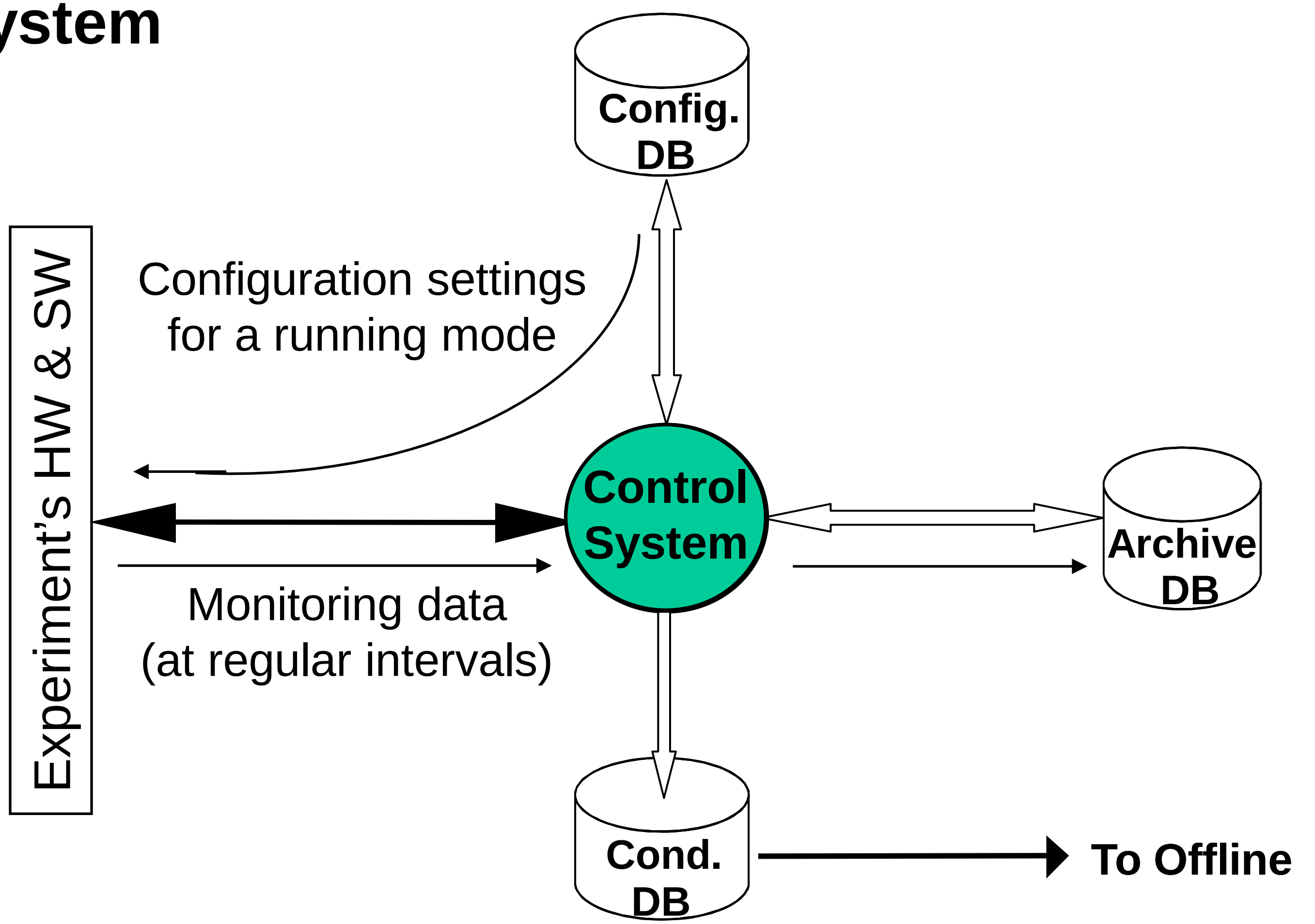
- **Three main logical Database concepts in the Online System**



- **But naming, grouping and technology can be different in the different experiments…**

Online Databases

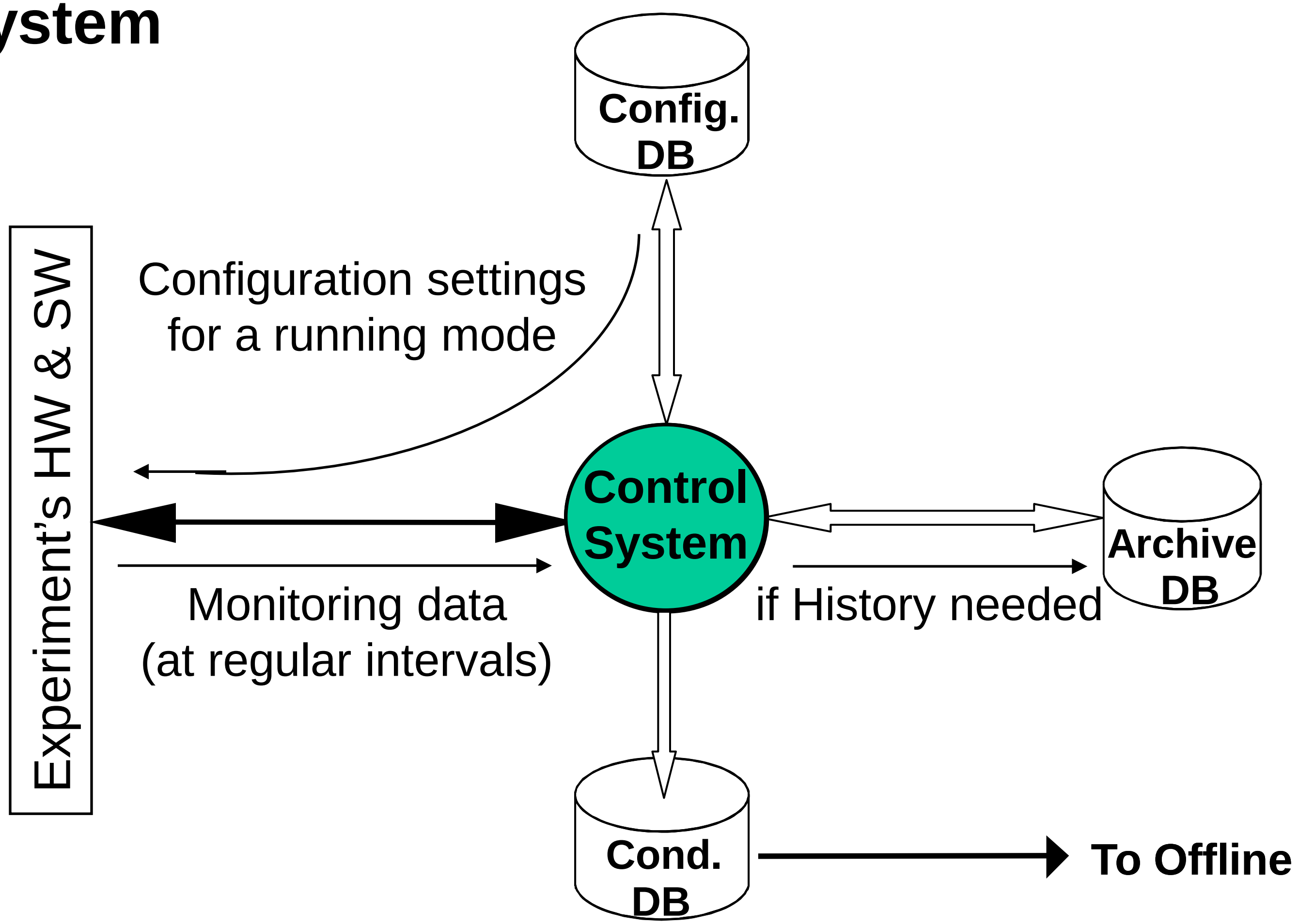
- **Three main logical Database concepts in the Online System**



- **But naming, grouping and technology can be different in the different experiments…**

Online Databases

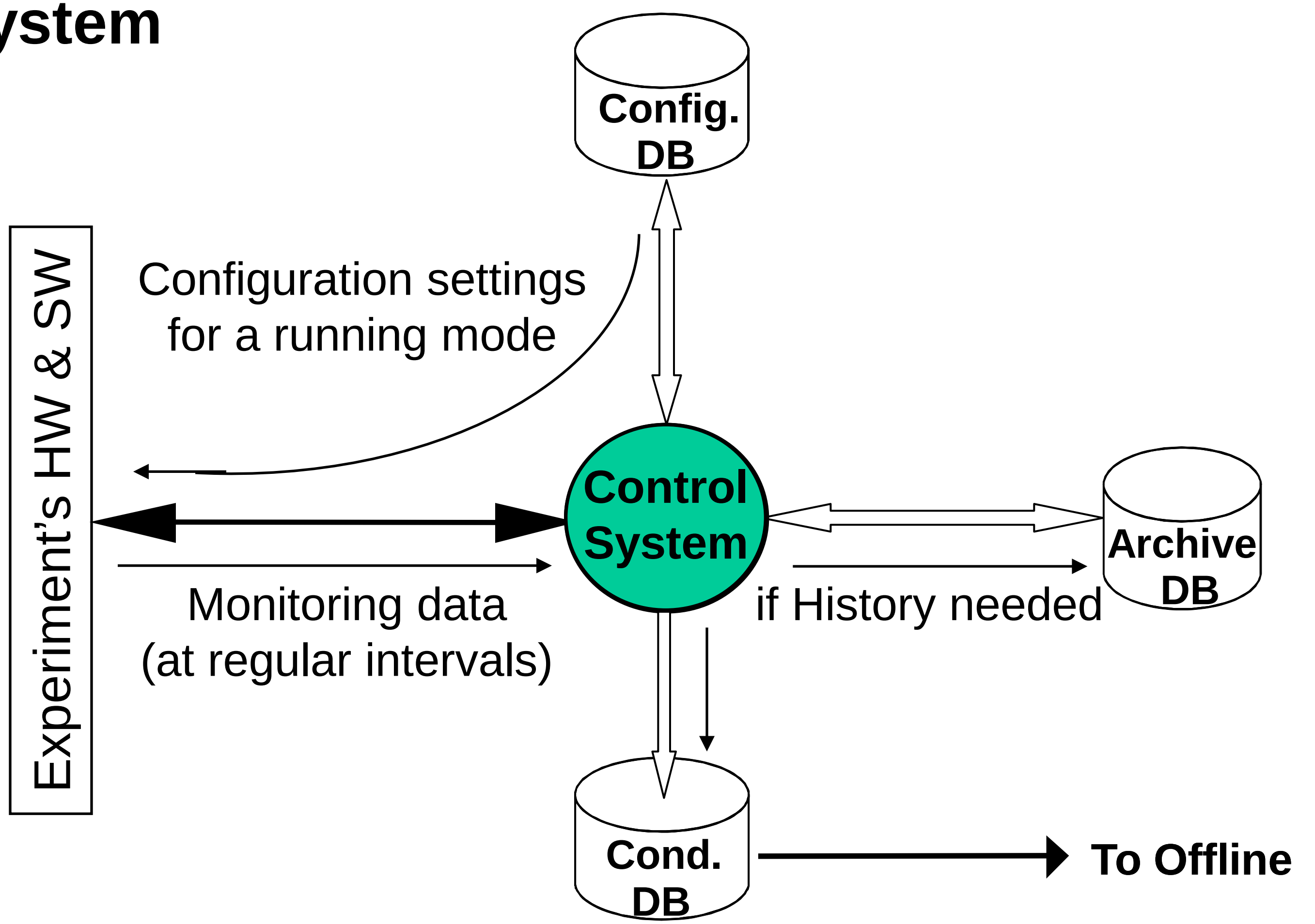
- Three main logical Database concepts in the Online System



➡ **But naming, grouping and technology can be different in the different experiments…**

Online Databases

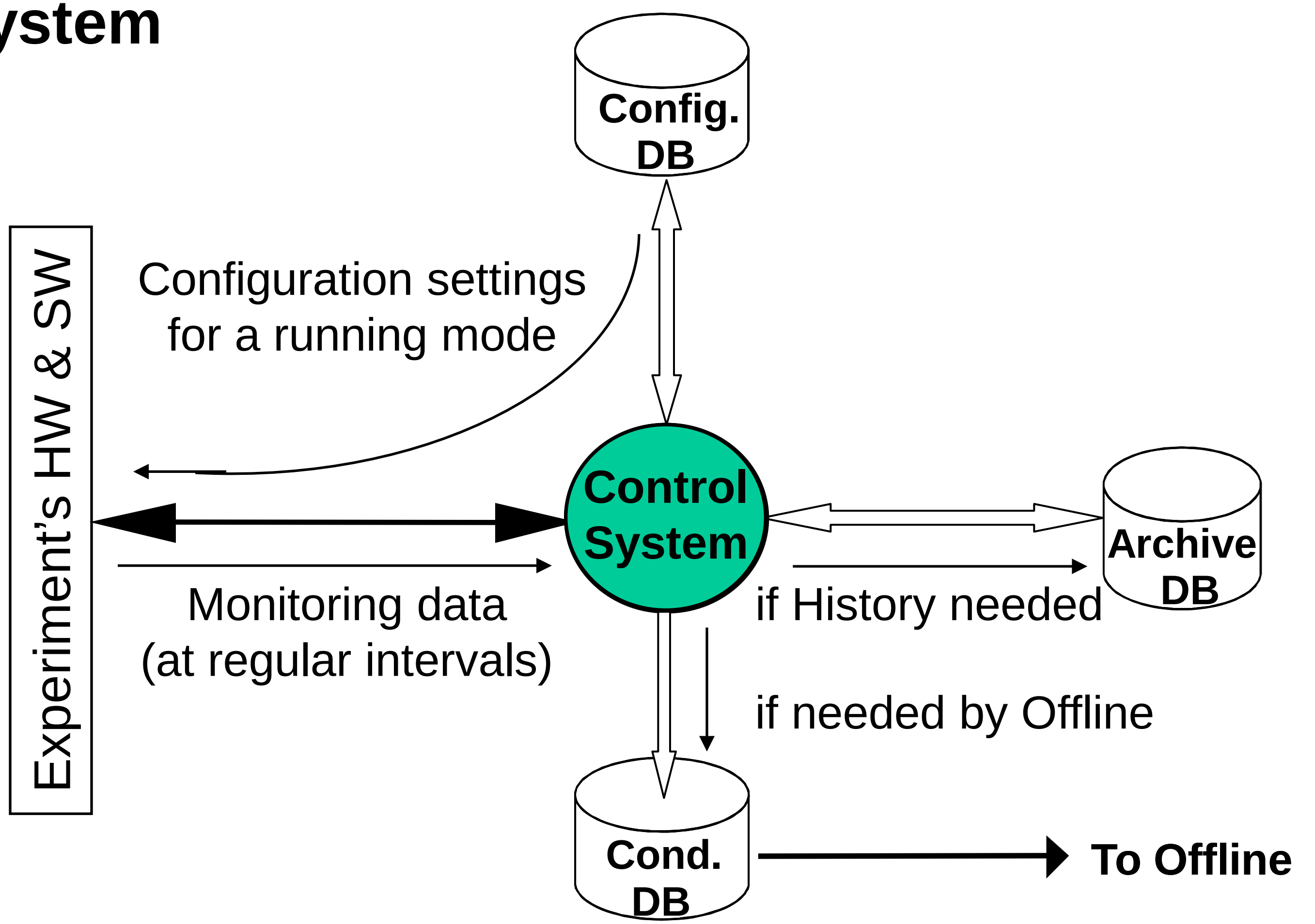
- **Three main logical Database concepts in the Online System**



- **But naming, grouping and technology can be different in the different experiments…**

Online Databases

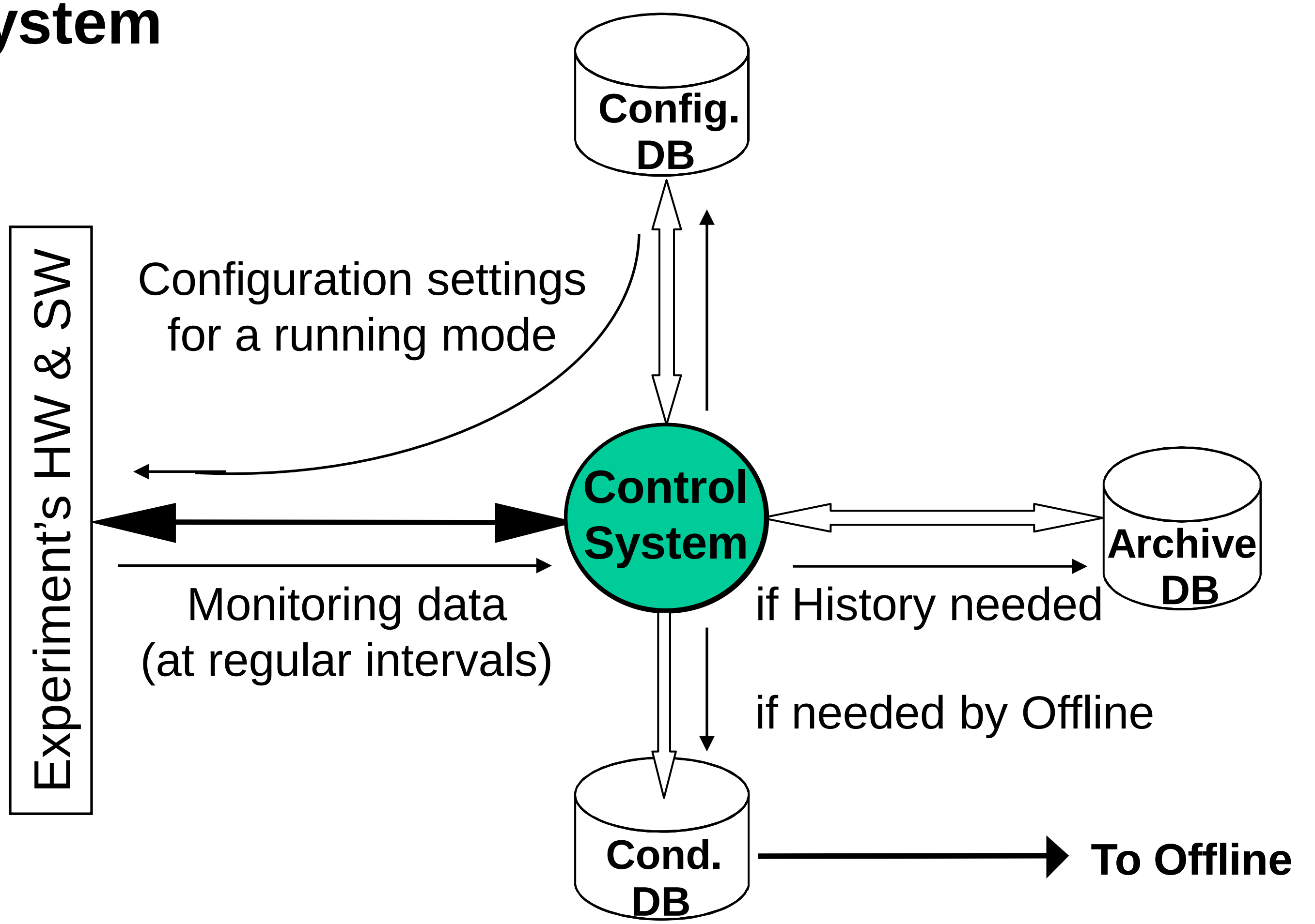
- **Three main logical Database concepts in the Online System**



➡ **But naming, grouping and technology can be different in the different experiments…**

Online Databases

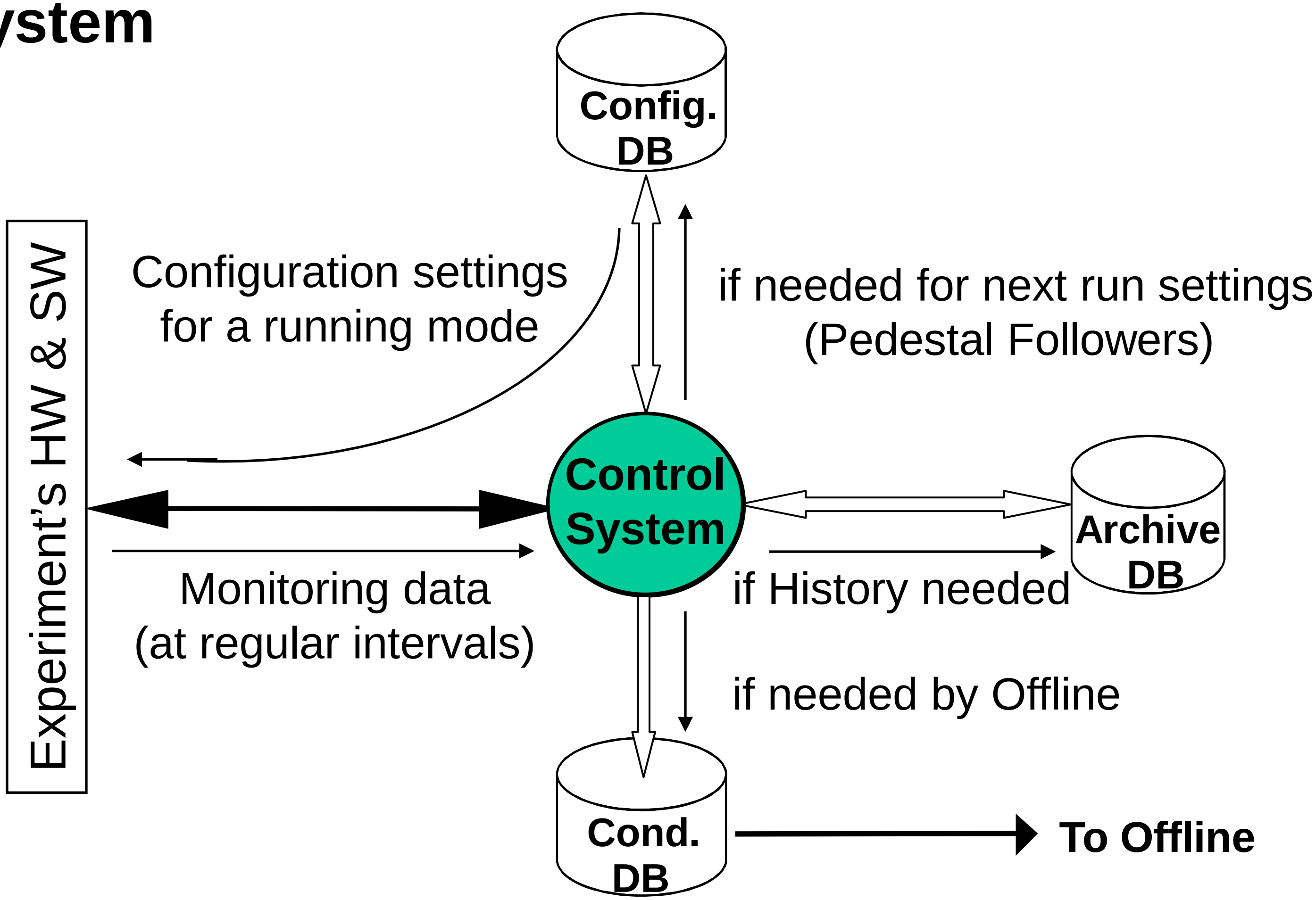
- **Three main logical Database concepts in the Online System**



- **But naming, grouping and technology can be different in the different experiments…**

Online Databases

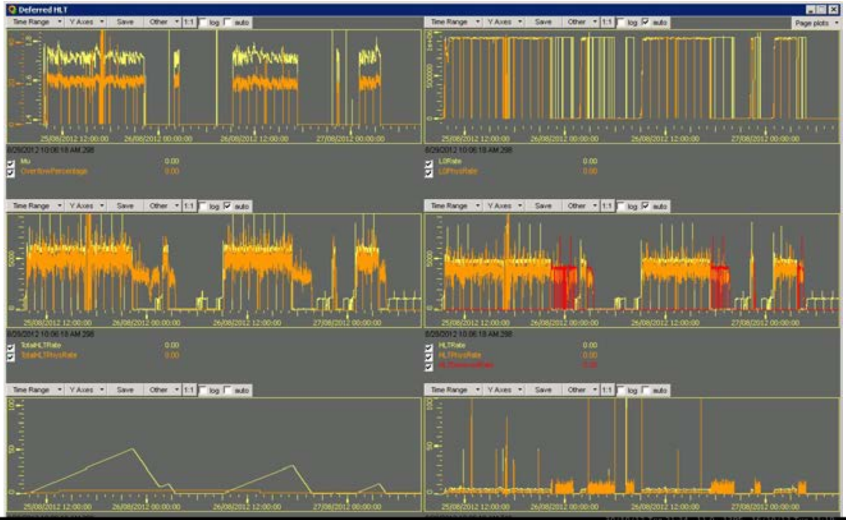
- **Three main logical Database concepts in the Online System**

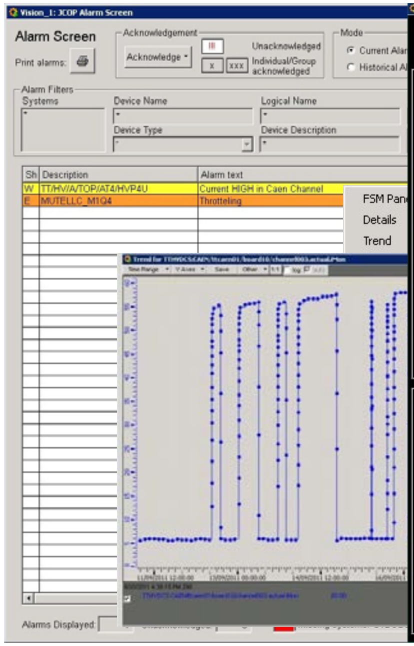


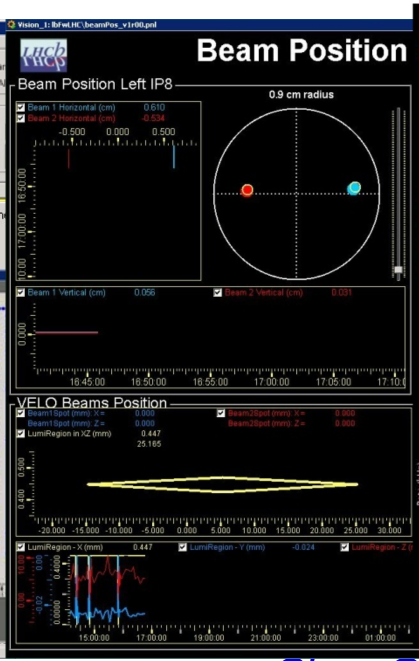
➡ **But naming, grouping and technology can be different in the different experiments…**

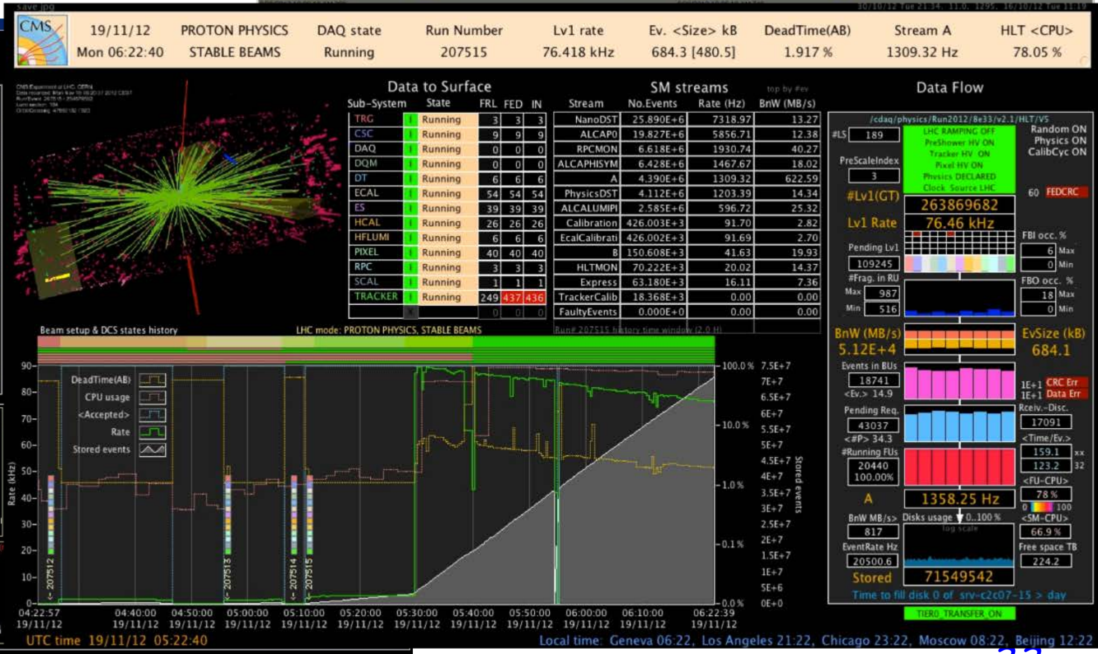
User Interfacing

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

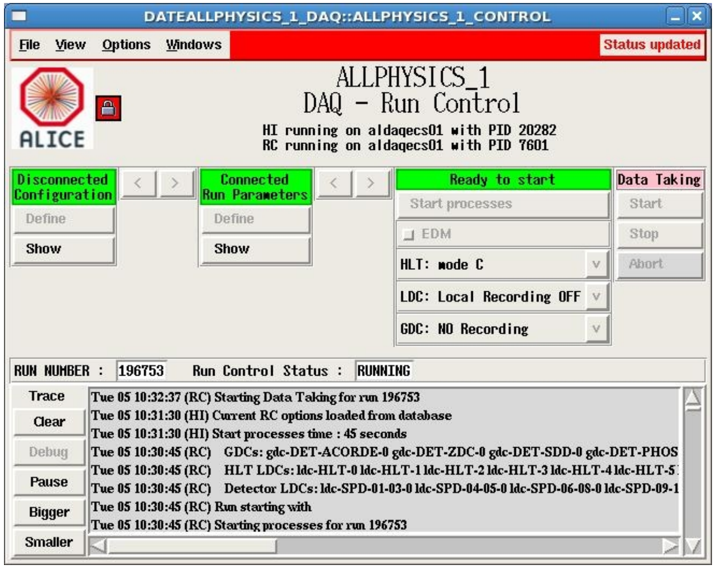




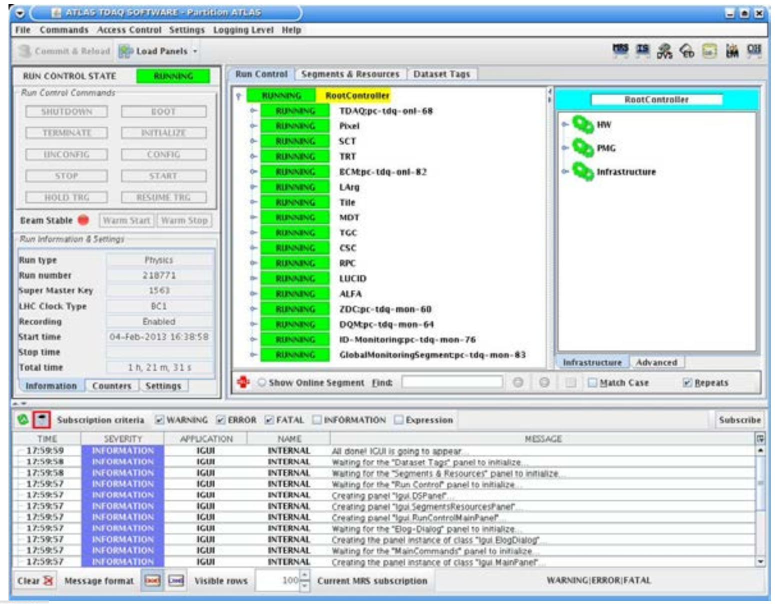




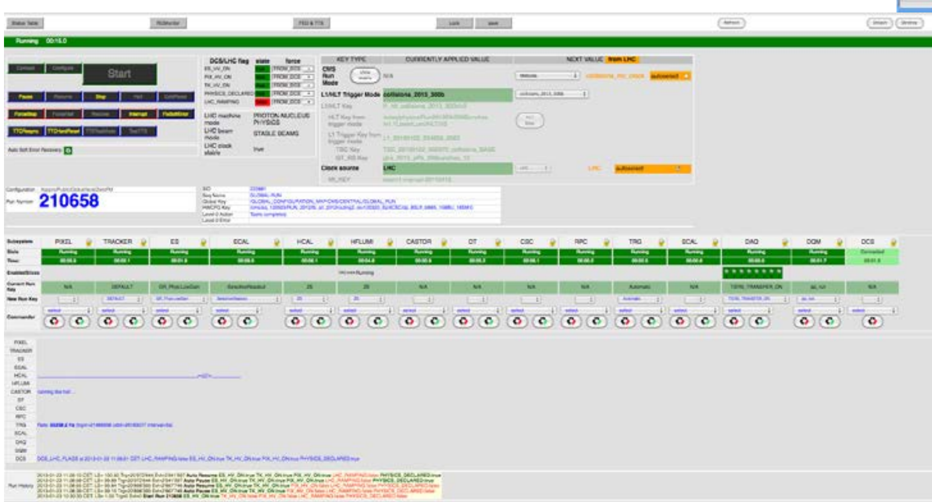
Run Control



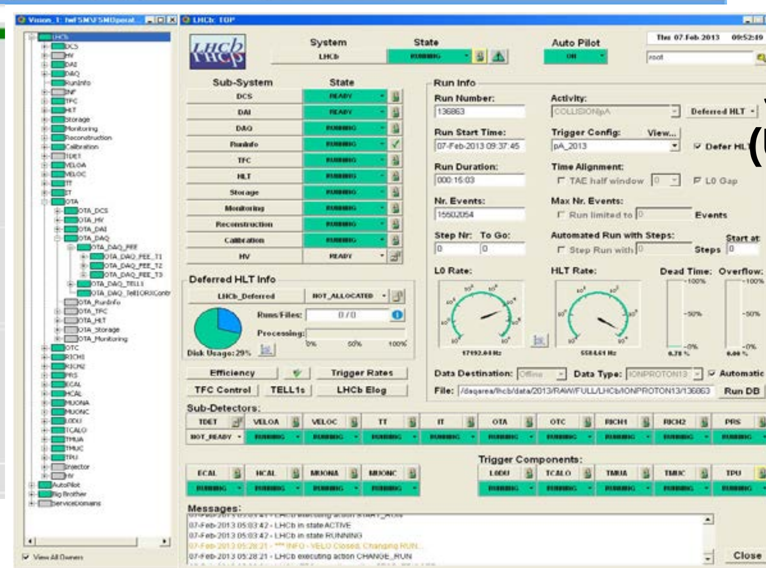
ALICE: Tcl/Tk



ATLAS: Java (modular)

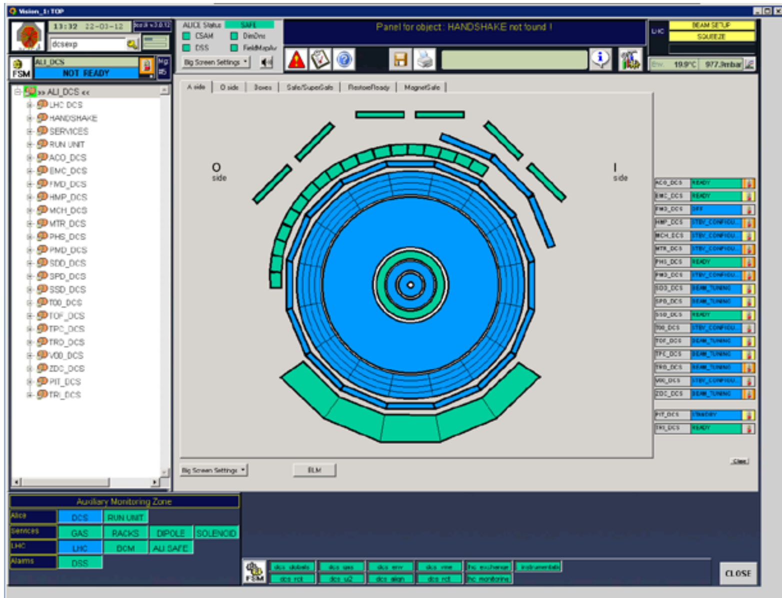


CMS: Web Tools (JavaScript+HTML)



LHCb: JCOP FW (UI builder)

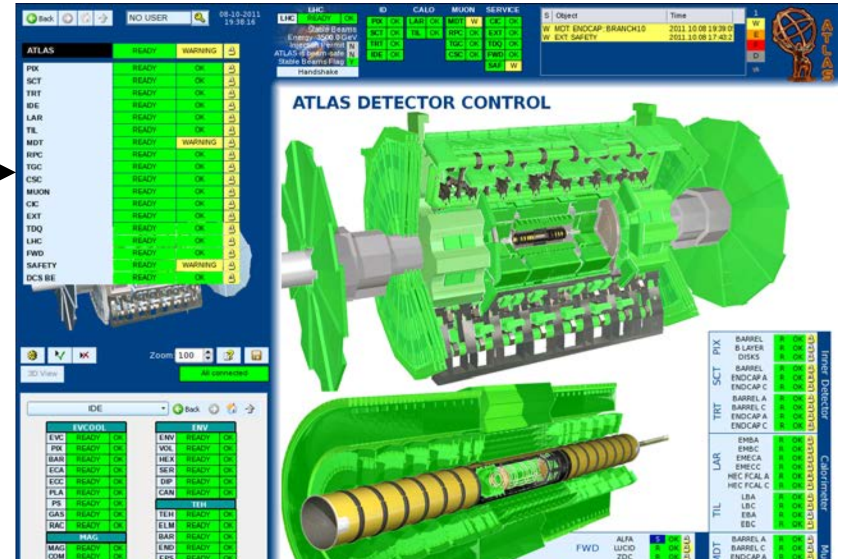
Detector Control System

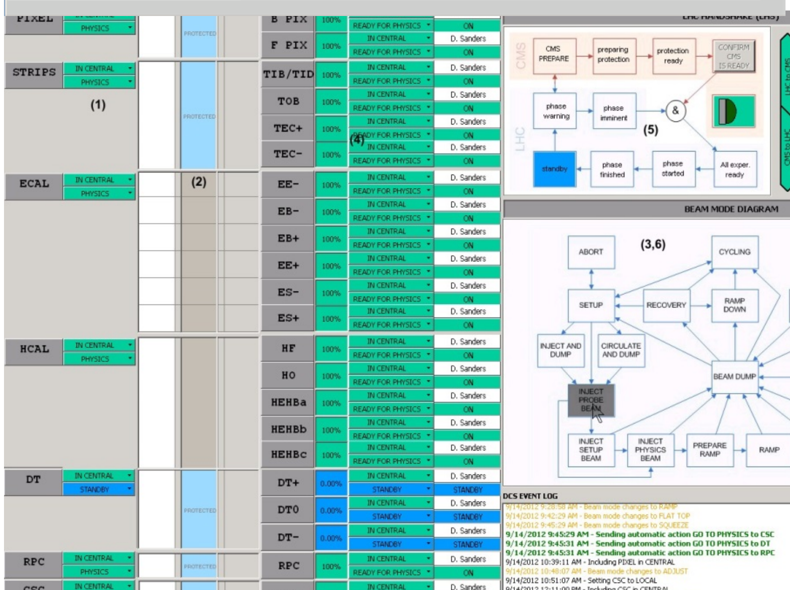


JCOP FW

← ALICE

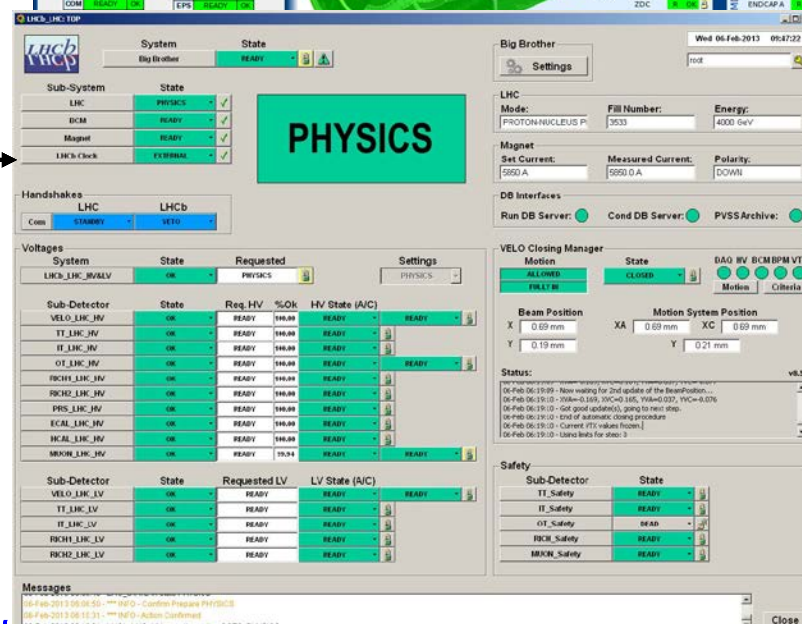
ATLAS →





← CMS

LHCb →



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 DCS (HV) automatically handled driven by the LHC State
    - In most cases Run Control Shifter manually Configures/Starts the Run

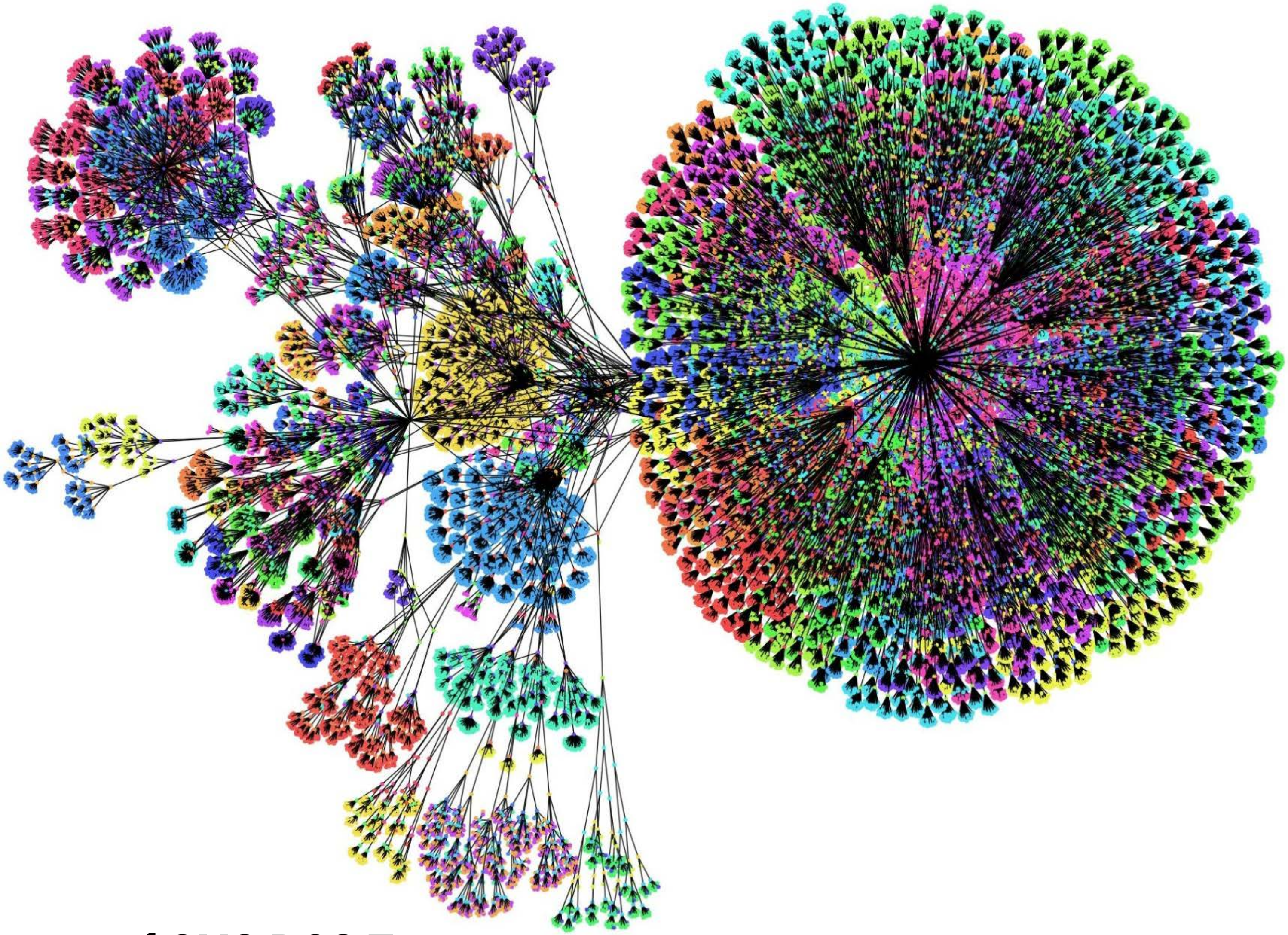
Size and Performance

- **Size of the Control Systems (in PCs)**
  - ALICE: 1 DAQ + ~100 DCS
  - ATLAS: 32 DAQ + 130 DCS
  - CMS: 12 DAQ + ~80 DCS
  - 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

➡ **All Experiments work Beautifully!**

LHCb Control System



- **Courtesy of CMS DCS Team**