Accelerator Controls

Stephane Deghaye (BE-CSS-FST)

29th & 30th of November 2021
Agenda

1. What is an accelerator made of?
2. Control System Requirements
3. Implementation Philosophy
4. A bit of History
5. Hardware & Software Architecture
6. CERN examples & Key Components
7. Detailed implementations and concepts
CERN Accelerator Complex
What is an accelerator made of?

Particle accelerators are either linear or circular.
What is an accelerator made of?

Linac4

LEIR
What is an accelerator made of?

Particle accelerators are either linear or circular

Need equipment to:
- Accelerate the particles
- Bend and steer the beam
- Focus the beam
- Observe and measure its characteristics
- Inject in and extract from the accelerator
What is an accelerator made of?

A beam pipe in which the beam circulates

Elements in and around the pipe to:

• Steer and focus the beam
• Accelerate the beam and give it structure
• Measure the beam
• Inject and extract the beam
• …

Oversimplified view!!
What is an accelerator made of?

A beam pipe in which the beam circulates

+ 

Elements in and around the pipe to:

- Steer and focus the beam
-Accelerate the beam and give it structure
- Measure the beam
- Inject and extract the beam
- …

Oversimplified view!!
What is an accelerator made of?

A beam pipe in which the beam circulates

+ 

Elements in and around the pipe to:

• Steer and focus the beam
• Accelerate the beam and give it structure
• Measure the beam
• Inject and extract the beam
• …

Oversimplified view!!
What is an accelerator made of?

A beam pipe in which the beam circulates

Elements in and around the pipe to:

• Steer and focus the beam
• Accelerate the beam and give it structure
• Measure the beam
• Inject and extract the beam
• …

Oversimplified view!!
What is an accelerator made of?

A beam pipe in which the beam circulates

+ Elements in and around the pipe to:
  • Steer and focus the beam
  • Accelerate the beam and give it structure
  • Measure the beam
  • Inject and extract the beam
  • …

Oversimplified view!!
Why a Control System?

- Particle accelerators are made of many components to control and monitor the beams produced.
- The physicists and operators need to be able to remotely control and monitor these elements — this is the role of the Control System.
Control System Requirements
Control System Key Requirements

Act on accelerator elements (settings & states)

➢ Minimum: direct access to the hardware values
➢ Ideal:
  ➢ Model-driven control to work at a higher level
  ➢ Global transactional synchronisation
Control System Key Requirements

Monitor the elements (instruments & actuators)

- Minimum: Display raw acquisitions
- Ideal: Time-tagged, coherent acquisitions, post-processing for quick detection of abnormal situations
Control System Key Requirements

Long-Term Memory of Settings & Acquisitions

- AKA Logging
- Accelerator performance & post-mortem analysis, fine tuning of the machines
- Minimum: structured time-series in a simple format (CSV, SDDS, etc.)
- Ideal: Years of data (settings & acquisitions) with performant data extraction & analysis tools
More Control System Requirements

➢ Automation
  ➢ Generate initial values, play sequences, feedback loops, etc.
  ➢ Minimum: Non-interactive scripts
  ➢ Ideal: Model-driven generation, flexible sequencer (almost like a debugger), automated actions (decision tree, machine learning)
More Control System Requirements

- Safety for machine protection & operational availability
  - Minimum: Machine interlock to protect the hardware
  - Ideal: High-level fast-reaction interlocks and role-based access to prevent the wrong action at the wrong time
More Control System Requirements

➢ Diagnostics

➢ Detection, identification, and follow-up of problems in the controls infrastructure
➢ Minimum: Non-interactive status screens
➢ Ideal: Online monitoring, remote interventions (e.g. power cycle), failure prediction (Machine Learning), analysis tools
And many more...
Many requirements from physicists and operators

Accelerators made of many elements

- Early accelerators, e.g. Proton Synchrotron (PS), were small
  < 5'000 devices
- Latest accelerators, e.g. LHC, are much more complex to operate
  30’000+ elements

The Control System’s job is to hide the complexity and help you to do your job as efficiently as possible.
Implementation Philosophy
Implementation Philosophy - Hardware

As much as possible:

➢ Apply vertical industrial control system solutions
  ➢ PLCs for industry-like process control (electricity, cooling & ventilation, vacuum, cryo)

➢ Restrict home-made HW development to specific applications
  ➢ Beam optics controls (i.e. all power converters),
  ➢ Injection and Extraction systems,
  ➢ Beam instrumentation,
  ➢ RF,
  ➢ Collimation,
  ➢ Timing Systems,
  ➢ Etc.

➢ Base the HW architecture on available standards and Commercial Off-The-Shelf (COTS)
  ➢ Standards for complex embedded I/O systems with high performance demands
  ➢ COTS electronic modules for generic features (CPUs, serial controller boards, ADCs, etc.)
  ➢ COTS desktop PCs & servers for control rooms and application servers
  ➢ Standard fieldbuses for applications requiring real-time features and radiation hardness (e.g. WorldFIP), and less stringent applications (Profinet/Ethercat)
  ➢ Standards for cost-effective I/O systems for networking (fieldbus controllers)
  ➢ GPS for time stamping and overall accelerator synchronization
Implementation Philosophy - Software

As much as possible:

➢ Apply vertical industrial control system solutions
  ➢ Supervisory Control and Data Acquisition Systems (SCADA) for commands, graphical user interfaces, alarms, etc. of industrial systems

➢ Rely on common technologies and tools
  ➢ Important for aspects such as recruitment, education & training
  ➢ DBs & Storage solutions (e.g. Hadoop)
  ➢ Communication protocols
  ➢ Monitoring solutions used in the industry (e.g. ICINGA)

➢ Privilege Open-Source Software
  ➢ Avoid vendor lock-in
  ➢ Control license cost
  ➢ Manage the Total Cost of Ownership (TCO)
A bit of History
Moore's Law

Prehistory

Early days

Modern days

The data visualization is available at OurWorldInData.org. There you find more visualizations and research on this topic. Licensed under CC-BY-SA by the author Max Rosar.
Control System Prehistory

- Accelerators are small and overall less complex (e.g. no superconducting magnets)
  - No more than a few thousands of devices to control

- No computing infrastructure and limited possibility to model

- Actuator and monitors are physically in the local control rooms (e.g. buttons, knobs, analogue oscilloscopes, etc.)
Control System Prehistory
The Early Days

Beginning of remote controls

- Still limited by the available performance
- Lack of standards and common frameworks → more DIY and custom solutions
- Emergence of several controls solutions, aiming at different types of accelerators (at first)
  - EPICS (driven by US labs),
  - Tango (driven by ESRF (Fr) – synchrotron light sources)
  - CERN¹

¹ non-exhaustive list
Modern days

➢ Hardware has become powerful
  ➢ E.g. embedded systems at CERN in late 90s had 64 MB of RAM; Nowadays, they have 8 GB
  ➢ Most of the needs are covered
  ➢ Yet, users want more and more data (turn-by-turn acquisitions, big-data solution for the long-term storage, etc.)

➢ Software industry has become a major actor worldwide.
  ➢ We can rely on many readily available technologies that open the doors to much more powerful systems

➢ We still need to integrate and customise them to the very specific domain of particle accelerator controls
  ➢ Not all solutions are appropriate; Need to remember accelerator controls ≠ selling plane tickets
  ➢ Mastering the different solutions with their evolution, limitations, etc. is a major challenge
  ➢ The rhythm of updates is no longer under our control. E.g. recent Linux CentOS changes
50 years of technology evolution
High-Level Architecture
of a modern control system
High-Level Hardware Architecture

Client Tier
Tip of the iceberg
Keywords: Console Terminal

Server Tier
Central computing infrastructure
Keywords: Back-End Computer (BEC)

Resource Tier
Electronics close to the accelerator
Keywords: Front-End Computer (FEC)
High-Level Hardware Architecture

Resource Tier

- **Open enclosures**
  - Easy access
  - Better cooling and power available
  - But expensive

- **Closed enclosures**
  - Possible for simple functions (e.g. fieldbus control)
  - Cost effective; when deployed in big number, e.g. LHC power converter control gateways
High-Level Hardware Architecture

Middle Tier

- IT-Computer-centre type of hardware
- High-density
- Highly available (redundancy and hot-swap)
High-Level Hardware Architecture

Control Room Computers

- As much as possible COTS desktop PCs but MTBF requirements might be difficult to satisfy
- Users expect modern reactive GUIs
- Several layers of screens to have as much data as possible available

Wall screens – Non-interactive summary applications

2nd row – Non-interactive detailed applications

Front row – Interactive applications
High-Level Software Architecture

**Front-end Tier**

Real-time control and acquisition

- Limited, local scope
- Fast reaction possible (interrupts)
- Limited computing power (compared to other tiers)
- Equipment processing to provide a high-level view of the hardware
- Real-time (RT) applications rely on frameworks, which capture the recurring aspects (react to events, publish new data, etc.)
  
  E.g. Front-End Software Architecture (FESA)

- Based on technologies closed to the hardware (C for drivers, C++ for RT, etc.)

---

**Diagram:**

- CERN’s general purpose Ethernet network (GPN)
- Operators’ consoles
- Equipment experts’ consoles
- Fixed displays
- Application servers
- Databases
- File servers
- FECs’ boot servers

---

**RT Application**

(e.g. Kicker system control)

**RT Frameworks**

(e.g. FESA)

**Device Drivers**

(e.g. Timing receiver kernel module)

**Operating System**

(Linux)

---

**Digital Interfacing**

- Analogical
- Switched
- Optical

---

**Accelerator Equipment**

- VME crates
- PCIe crates
- Fieldbuses

---

CERN’s Gigabit Ethernet network (TN)
High-Level Software Architecture

**Business Tier**
General purpose services & Specific business logic

- Broader scope; able to coordinate the entire accelerator
- Powerful computers
- Less reactive (network) and at a higher-level of abstraction
- Based on technologies that are better suited for high-level business logic (e.g. Java)

---

CERN's general purpose Ethernet technical network (TN)

### Business Tier
- **Application servers**
- **Databases**
- **File servers**
- **FECs' boot servers**

### Settings Management
(CNA/LSA)

### Long-term Logging
(NXCALS)

### Automation
(Sequencer, SIS)

### Etc.

---

Front-end computers (FEC)

PLCs

VME crates

PICMG1.3 platforms

Accelerator Equipment

Analogue & digital interfacing

Fieldbuses

---

29 & 30-11-2021

Accelerator Controls - S. Deghaye
High-Level Software Architecture

Presentation Tier
Graphical applications

Different technologies available
- Java Swing, Java FX
- Qt, PyQt
- Web ecosystem (Angular, View.js, etc.)

Keywords:
- Graphical User Interface (GUI)
- Command-line interface (CLI)

Communication

- Accelerator-specific protocols for the lower layers
- Controls Middleware (CMW)
- Potentially, more generic technologies for the higher layers
  - RMI/JMS
  - REST API
  - gRPC
  - …

Controls Middleware & more generic solutions

Fieldbuses
A few examples
And the key components used...
Example 1 – Control

New value = $x$

InCA/LSA Server
FESA Class
VMEBus board

GUI

Server

RT Application

RT Application

RT Application

Database

Hardware

Hardware

Hardware

Oracle Database

Control now!

General Machine Timing (GMT)
Example 2 – Monitoring

Here are a, b, c, and Everything is OK

UCAP* Server

a', b', c' @ t_i

GUI

UCAP: Unified Controls Acquisition & Processing framework

FESA Class

RT Application

RT Application

RT Application

0x61
0x62
0x63

Hardware

Hardware

Hardware

VMEBus board

Read now! (time = t)

Measure now!

General Machine Timing (GMT)
Example 3 – Logging

Values on the 06/06/2021?

SWAN Notebook

NXCALS

Hadoop

FESA Class

RT Application

RT Application

RT Application

Hardware

Hardware

Hardware

Timing System

*SWAN: Service for Web based ANalysis
Optimise the throughput of the accelerators

- New requirements: Change references of almost all equipment from pulse to pulse
- The control system must be hard real-time
- The central timing system sequences the entire beam production

Accelerators Optimisation

Physics

mise the throughput of the accelerators

- requirements: Change references of almost all equipment from pulse to pulse
- The control system must be hard real-time
- The central timing system sequences the entire beam production
Program for Tomorrow…

• Review the examples but looking at the technical implementation details
  • Which technologies
  • Workflows
  • Design choices
  • Etc.

• If possible, review
  • The high-level architecture (get used to the names)
  • The examples
  • The timing system
Want to know more?

CERN Beams Department (https://beams.cern/)


Tango Controls (https://www.tango-controls.org/)

EPICS (https://epics-controls.org/)
Day 2
Agenda

1. What is an accelerator made of?
2. Control System Requirements
3. Implementation Philosophy
4. A bit of History
5. Hardware & Software Architecture
6. CERN examples & Key Components
7. Detailed implementations and concepts
High-Level Hardware Architecture

**Client Tier**
Tip of the iceberg
Keywords: Console Terminal

**Server Tier**
Central computing infrastructure
Keywords: Back-End Computer (BEC)

**Resource Tier**
Electronics close to the accelerator
Keywords: Front-End Computer (FEC)

---

Keywords:
- Front-End Computer (FEC)
- Server Tier
- Client Tier
- Resource Tier
- Accelerator Equipment
- Accelerator Controls - S. Deghaye

29 & 30-11-2021
High-Level Software Architecture

Front-end Tier
Real-time control and acquisition

- Limited, local scope
- Fast reaction possible (interrupts)
- Limited computing power (compared to other tiers)
- Equipment processing to provide a high-level view of the hardware
- Real-time (RT) applications rely on frameworks, which capture the recurring aspects (react to events, publish new data, etc.)
  E.g., Front-End Software Architecture (FESA)
- Based on technologies close to the hardware (C for drivers, C++ for RT, etc.)
High-Level Software Architecture

**Business Tier**
General purpose services & Specific business logic

- Broader scope; able to coordinate the entire accelerator
- Powerful computers
- Less reactive (network) and at a higher-level of abstraction
- Based on technologies that are better suited for high-level business logic (e.g. Java)
High-Level Software Architecture

**Presentation Tier**
Graphical applications

- Different technologies available
  - Java Swing, Java FX
  - Qt, PyQt
  - Web ecosystem (Angular, View.js, etc.)

Keywords:
  - Graphical User Interface (GUI)
  - Command-line interface (CLI)

**Communication**

- Accelerator-specific protocols for the lower layers
- Controls Middleware (CMW)
- Potentially, more generic technologies for the higher layers
  - RMI/JMS
  - REST API
  - gRPC
  - …

---

**Controls Middleware & more generic solutions**

- Application servers
- Databases
- File servers
- FEs’ boot servers

---

**Controls Middleware**

- Front-end computers (FEC)
- PLCs

---

**Operator’s consoles**

- Equipment experts’ consoles
- Fixed displays

---

**Recap**
Use Case 1: Control

- **Who:** PS operator
- **What:** Change the strength of the extraction septum towards the SPS for the Fixed Target beam
- **Involved controls components:**
  - InCA/LSA (Setting management)
  - CMW (Controls Middleware)
  - FESA (Real-time hardware control)
  - Timing (Synchronisation)

In all the examples, we assume all the configuration is already done and we focus on the run-time aspects
Example 1 – Control –

More strength on my extraction septum ➔ 0.02978668

K = 0.02978668

Current ➔ K ➔ Current

Current = 15280

0x3BB0*

Control now!

*not a real value
Example 1 – Control - Communication

Remote Method Invocation (RMI) to call the InCA server
! Limited to Java to Java ➔ ReST

Trim request (new value, context, etc.)

Save the new values (K, I, current) along with some contextual info (who, what time, etc.)

1. Parameter hierarchy
2. Current values (p, K, I, …)
3. Name of the Makerule to apply
4. Mapping between cycle and timing user ("MTE 21" = CPS.USER.SFTPRO1)

Drive the new value by setting the device PE.SMH16 for timing user
CPS.USER.SFTPRO1 (Property: SettingPPM)

Makerule: Java class to compute parameters based on other parameters, hardware characteristics (calibration curve) and cycle specific values

InCA Server

BEC

FESA Class

FEC

Remote Method Invocation (RMI) to call the InCA server
! Limited to Java to Java ➔ ReST

Trim request (new value, context, etc.)

Save the new values (K, I, current) along with some contextual info (who, what time, etc.)

1. Parameter hierarchy
2. Current values (p, K, I, …)
3. Name of the Makerule to apply
4. Mapping between cycle and timing user ("MTE 21" = CPS.USER.SFTPRO1)

Drive the new value by setting the device PE.SMH16 for timing user
CPS.USER.SFTPRO1 (Property: SettingPPM)
Example 1 – Control – Low-level

- **InCA Server**
  - CMW-RDA based on ZeroMQ
  - Supports three basic operations:
    - Get (read)
    - Set (write)
    - Subscribe (monitor)

- **CMW Directory service**
  - Stores the mapping between the server names and their addresses (TCP://xxx)
  - Very critical ➔ Redundant

- **Controls Configuration DB**
  - ORACLE
  - Stores the mapping between the device names and the servers hosting them
  - CCDB is at the heart of our database-driven control system

- **FESA Server**
  - PE.SM316
  - Buffer Manager

- **FEC**
  - Set request (new value, context, etc.)
  - Set Action
  - New value
  - Value (t-1)
  - Value (t-2)
  - Internal storage
Example 1 – Control – Beam production

Timing System (GMT)

FESA RT Action

Internal storage
New value
Value (t-1)
Value (t-2)

Control now!
Extract now!

SPS
PS
PSB

t_{ctrl}

t_{extract}

Septum
Power converter

29 & 30-11-2021
Accelerator Controls - S. Deghaye
Use Case 2: Acquisition

- **Who:** PS operator

- **What:** Keep an eye on acquisition values of many control devices. The low-level data needs post-processing and will be displayed as a graph in a web page

- **Involved controls components:**
  - Timing (Synchronisation)
  - FESA (Real-time hardware control)
  - CMW (Controls Middleware)
  - UCAP (Unified Controls Acquisition & Processing)
  - WRAP (Web Rapid Application Platform)

*In all the examples, we assume all the configuration is already done and we focus on the run-time aspects*
Example 2 – Monitoring

Here are the losses, everything is OK

GUI

UCAP® Server

FESA Class

FESA Class

FESA Class

a @ t_i

b @ t_i

c @ t_i

0x61

0x62

0x63

Hardware

Hardware

Hardware

VMEBus board

Timing System (GMT)

*UCAP: Unified Controls Acquisition & Processing framework
Example 2 – Monitoring – Low-level acquisition

Timing System (GMT)

FESA RT Action

Rolling Buffer
- New value
- Value (t-1)
- Value (t-2)

FESA Server

Notify for t_acq

CMW-RDA updates

SPS
PS
PSB

Notify for t_acq

Acquire now!

Read now!

CMW-
RDA updates

Notify for t_acq
Example 2 – Monitoring – Post-processing

- How to group the incoming data?
  ➔ Start Cycle timestamp (AKA cyclestamp)

- When to trigger the post-processing?
  ➔ Once all the data is there or after a time-out

- How long to wait for late comers?
  ➔ Configurable time-out

- What to do if no data is published?
  ➔ Out-of-the-box monitoring
Example 2 – Monitoring – Diagnostics
Example 2 – Monitoring – Graphical User Interface

- **Device Server**
  - UCAP Node
    - CMW-RDA updates
  - SWING
    - REST for command/response
    - HTTP/2 push events for live updates

- **Web back-end server**
  - ORACLE
    - BEC
      - Controls Configuration DB
      - Stores the configurations of the web dashboards

- **Simplified flow**

29 & 30-11-2021
Accelerator Controls - S. Deghaye
Use Case 3: Logging

- **Who:** Accelerator physicist
- **What:** Store acquisition values of many control devices long-term and perform analysis
- **Involved controls components:**
  - Timing (Synchronisation)
  - FESA (Real-time hardware control)
  - CMW (Controls Middleware)
  - NXCALS (Data Logging)
  - SWAN (Web-based analysis tool)

*In all the examples, we assume all the configuration is already done and we focus on the run-time aspects*
Example 3 – Logging

Values on the 06/06/2021?

SWAN Notebook

NXCALS

Hbase/Hadoop

a, b, c @ 06-06-21

a, b, c @ 06-06-21

FESA Class

Hardware

FESA Class

Hardware

FESA Class

Hardware

a @ t_i

b @ t_i

c @ t_i

Already covered in the previous example

*SWAN: Service for Web-based ANalysis
Example 3 – Logging - Ingestion

Data source → Ingestion API → ETL → To storage

- High-Availability (fault tolerant & self healing)
- Load balancing
- Monitoring (see UCAP)

- Local storage for 36 hours
- Data reliability in case of storage failure

NXCALS Cluster

~20 BEC (48 cores, 512 GB RAM)

2.5 TB/day

CMW-RDA Update

FESA Server

100s FECs

Local storage for 36 hours

Data reliability in case of storage failure
Example 3 – Logging - Storage

- **ETL**
  - Continuous buffered writes

- **ETL**
  - Periodic writes (~2 mins)

**NXCAL Storage (CERN IT)**

- **HBASE**
  - Once/day

- **Hadoop**
  - Compactor

**Parquet**

- Nov-2021:
  - ~40 servers
  - 1.1 PB logged data

- Recent data ~ last 4-5 days

- All of the data after compaction

---

29 & 30-11-2021

Accelerator Controls - S. Deghaye
Example 3 – Logging - extraction

Analysis

Data

(Duration: Courtesy Ph. Elson)

NXCALS Storage

Recent data

Older data

Apache HBase

Spark

Apache Hadoop
Want to know more?

CERN Beams Department (https://beams.cern/)


Tango Controls (https://www.tango-controls.org/)

EPICS (https://epics-controls.org/)