Copyright statement and speaker's release for video publishing

The author consents to the photographic, audio and video recording of this lecture at the CERN Accelerator School. The term “lecture” includes any material incorporated therein including but not limited to text, images and references.

The author hereby grants CERN a royalty-free license to use his image and name as well as the recordings mentioned above, in order to broadcast them online to all registered students and to post them without any further processing on the CAS website.

The author hereby confirms that the content of the lecture does not infringe the copyright, intellectual property or privacy rights of any third party. The author has cited and credited any third-party contribution in accordance with applicable professional standards and legislation in matters of attribution.
Agenda

1. Control System Requirements
2. Implementation Philosophy
3. A bit of History
4. Hardware & Software Architecture
5. Key Components & CERN examples
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.

The Control System's job is to provide to the physicists and operators a means to:

- set reference values (aka setting) and states in active elements (e.g. power converters),
- read instruments,
- monitor the health of sub-systems,
- diagnose faults,
- etc.
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
Monitor the elements (instruments & actuators)

- Minimum: Display raw acquisitions
- Ideal: Time-tagged, coherent acquisition, post-processing for quick detection of abnormal situations
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

- 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

- 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

- **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. CERN 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.)
  - 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
  - COTS desktop PCs & servers for control rooms and application servers
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

Early days

Prehistory

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 Roser.
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

**Resource Tier**
Electronics close to the accelerator
Keywords:
- Front-End Computer (FEC) (CERN)
- I/O Controller (IOC) (EPICS)
- Device Server (Tango)

**Client Tier**
Tip of the iceberg
Keywords:
- Console (CERN)
- Terminal

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

**Application servers**
- Databases
- File servers
- FECs' boot servers

**Front-end computers (FEC)**
- VME crates
- PICMG1.3 platforms

**Accelerator Equipment**
- Analogue & digital interfacing
- Fieldbuses

**PLCs**
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. FESA @ CERN, POGO with Tango
- Based on technologies closed 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)
Human-Machine Interface (HMI)
Command-line interface (CLI)

Communication

- Accelerator-specific protocols for the lower layers
- Channel Access (EPICS)
- CMW (CERN)
- Potentially, more generic technologies for the higher layers
  - RMI/JMS
  - REST API
  - gRPC
  - …
Key Components
A few examples from CERN Control System
CERN Example 1 – Control

New value = x

Fig. 1. The CERN example 1 of a control system. The GUI sends a new value (x) to the server, which is then transferred to the database. The database provides the data (a, b, c) to the RT applications, which control the hardware (0x61, 0x62, 0x63). The timing system receives the control signal and sets the general machine timing (GMT).
Here are a, b, c, and Everything is OK

UCAP Server

GUI

a', b', c' @ t

FESA Class

RT Application

RT Application

RT Application

Hardware

Hardware

Hardware

VMEBus board

0x61

0x62

0x63

Measure now!

Read now! (time = t)

Timing System

General Machine Timing (GMT)

"UCAP: Unified Controls Acquisition & Processing framework"
CERN example 3 – Logging

Values on the 06/06/2021?

Jupyter Notebook

NXCALS

Hadoop

a, b, c @ 06-06-21

a @ t_i, b @ t_i, c @ t_i

RT Application

Hardware

0x61

0x62

0x63

Read now! (time = t_i)

Measure now!

General Machine Timing (GMT)
Want to know more?

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


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

**EPICS** ([https://epics-controls.org/](https://epics-controls.org/))