Chapter II.12

Accelerator controls

Elke Zimoch
Paul Scherrer Institute, Villigen, Switzerland

All particle accelerators depend on control systems to integrate different devices and the autonomous controllers that are distributed throughout the facility into one coherent infrastructure. The control system provides an abstraction layer between hardware and the operators, and creates the environment that allows scientists to carry out their experiments. It also enables the technical support groups to compare previous data with current one in order to enhance the performance of their systems. This chapter provides an introduction into the structure and design of such control systems.

Why are accelerator control systems important? Everyone working with a particle accelerator will need to interact with a control system to measure data, control equipment, read out sensors, or run experiments. This chapter should raise a minimum awareness about the control system, its requirements and its limitations.

II.12.1 What is an accelerator control system?

Checking Wikipedia for answers does not help. There are interesting and relevant facts to learn, but no information on accelerator control systems are present at the moment. The best match is the article about distributed control systems [1], outlining one of the most popular software architectures for accelerator control systems.

As there is not quick and easy answer, lets rephrase the question: **What does an accelerator control system do?**

It makes the accelerator run in a controllable, reliable, flexible, and safe way. It has to provide good performance while being reasonably low priced.

Examples for controlled hardware are (examples from SLS 2.0 upgrade plan of the Swiss Light Source):

- 1300 power supplies for magnets,
- 300 motors for girder movers,
- 300 vacuum pumps and 600 vacuum sensors,
- 150 beam position monitors,
- 5 complex beam monitors (for parameters like size, tune, filling pattern of the storage ring, beam current),

This chapter should be cited as: Accelerator controls, E. Zimoch, DOI: 10.23730/CYRSP-2024-003.1637, in: Proceedings of the Joint Universities Accelerator School (JUAS): Courses and exercises, E. Métral (ed.),
CERN Yellow Reports: School Proceedings, CERN-2024-003, DOI: 10.23730/CYRSP-2024-003, p. 1637.

– 21 beamlines with 11 insertion devices and more than 1200 motors.

In addition the distance between components can be easily hundred meters and might be more for linear accelerators.

The control system provides means to control the accelerator: graphical user interfaces, feedback systems, experiment support (like scan tools and data acquisition), alarm handling and machine protection, dedicated programs for conditioning of RF components, and a timing system for real time interaction between different devices.

The accelerator control system connects the operator in the control room with the accelerator hardware. The control room might not be near the accelerator: SLS is 200 m away from PSI main control room, for SwissFEL the control room is a kilometre away from the accelerator, and some experiments are controlled remote from all over the world.

Conclusion of what an accelerator control system does: it provides the connection between the control room (operator) and the physical hardware of the accelerator. But everything that is not shown by the control system can not be seen by the operator. The control system provides a keyhole view of the accelerator. This fact makes it important to think through, which parameters of a new device are needed to get a comprehensive view of the status and functionality needed by the user of the device.

This leads to the question **who uses an accelerator control system** and what are the requirements of the users.

Obviously, the operators use the control system to control the accelerator. These operators are often people with technical background, but not necessary scientists. They want an easy and intuitive usage of the system and reliable alarm handling to make the usage safe. Accelerator physicists and experiment users (mostly scientists but not necessarily physicists) want to implement complex algorithms (e.g. feedback systems), want access to nearly all functionalities of the hardware, and strive for continuous extension of the control system with new hardware and new functionalities. System experts responsible for accelerator systems like vacuum, magnets, and RF want as well access to all functionalities of the hardware and in addition want a system that allows for easy maintenance.

A special group of people with interest in the control system might never even touch it themselves: the sponsors, investors, and the general public (who might be the sponsor due to taxes). These stakeholders are interested in a safe usage and reliable alarm handling, and they want the control system to be cheap in cost and in crew.

To summarize these thoughts: There are different groups of users or stakeholders with interest in the control system, and they have very different and partially contradictory requirements toward the system.

Definition:
An accelerator control system is a computer environment that allows remote access to the accelerator hardware with a lot of different functionalities to satisfy the requirements of several different user groups. In addition, a modern accelerator control system tries to unify the access to the different hardware.

II.12.1.1 Test questions

A1 Why is it advisable to include as many functionalities of a new device into the control system as possible?

A2 List the requirements that a vacuum system expert might have for the accelerator control system.

II.12.2 Control system architecture

To design a fitting accelerator control system the **requirement of different user groups** have to be taken into account. The most common requirements are:

Reliability (if I have beam time scheduled, I want it to happen),
Good Performance (speed, responsiveness, modern possibilities),
Flexibility (hey, I have a good idea, lets try . . .),
and **Easy maintenance** (this comes with limited resources).

Of these four, the last one is the most commonly overlooked criterion.

The reason lies in the long lifetime of accelerators compared to computers. For computer or IT infrastructure in general the quote from Lewis Caroll's Through the Looking-Glass comes to mind [2]:

> "Well, in our country," said Alice, still panting a little, "you'd generally get to somewhere else—if you run very fast for a long time, as we've been doing." "A slow sort of country!" said the Queen. "Now, here, you see, it takes all the running you can do, to keep in the same place. If you want to get somewhere else, you must run at least twice as fast as that!"

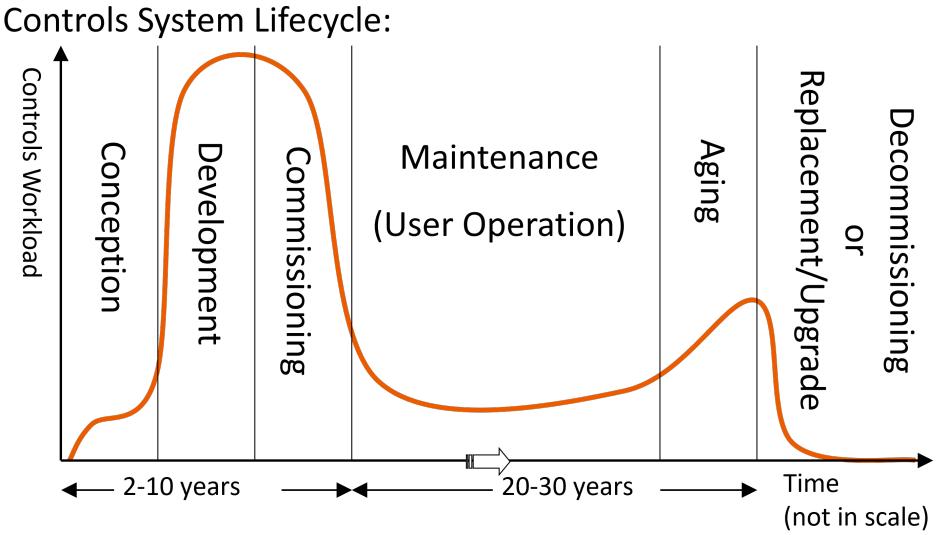

Fig. II.12.1: The lifecycle of an accelerator control system and the necessary workload during the different phases in the life of an accelerator.

Unfortunately, the workload for any part of the control system never goes back to zero during the accelerators lifetime (compare to the curve in Fig. II.12.1). The reason lies in the different time frames: a normal accelerator lifetime is approximately 20 to 30 years, while Linux distributions will change every

few years. As an example: Currently (2023) the Debian distribution of Linux gets new stable releases every two years with three years of official support. An accelerator control system running on Debian would need to be updated seven to ten times during the lifetime of the accelerator (if the Linux version should be kept up to date). Therefore, it is essential to design and build a control system for maintenance.

The maintenance aspect is also closely related to the need for flexibility. Both can be achieved by designing the control system from building blocks that come with well-defined interfaces and can be replaced independently. If implemented consistently, the result could be a collection of microservices [3]. Unfortunately, each interface between components will take some processing time, and this may not be desirable.

Processing time is one parameter of the "good performance" requirement. For some data, it is essential that it be processed in real time, for example for beam positions that are used in feedback loops to correct the particle beam trajectory. Other data, such as temperatures, changes relatively slowly and does not benefit from fast readout. Since fast processing is usually much more expensive than slow readout, it is imperative to identify which instruments will benefit most from these resources.

In general, the detailed requirements for an accelerator control system come from the various expert groups and users who want to use the accelerator for their experiments. However, these stakeholders rarely see the big picture, focusing instead on their own areas of expertise. It is up to the control system experts and developers to create a consistent framework that satisfies all these different needs.

Beside the user requirements there are additional technical requirements that have to be considered when designing an accelerator control system.

Most accelerator control systems use open-source software and firmware as base for specific applications that are often developed locally. The reasons are threefold:

- The developers have greater control over open-source software and can customize the official distributions to fit their requirements;
- Proprietary software will only be supported for a limited time and may require control system experts to upgrade their systems at the end of that period. This could potentially disrupt the planned operation of the accelerator;
- Licence fees may become unmanageable over time and could be restricted by the budget available.

In most cases there is no clear decision on a new hardware or software standard to be used in an accelerator. It is common for useful approaches to become de-facto standards without anyone really thinking about consequences. Therefore, any new technology used at an accelerator might end up as a new standard—even if it was only intended to be used at a single experiment set up by a postdoc. When using new hard- or software it might be useful to briefly think about the following questions: is the solution long-term maintainable (and keep the lifetime of the accelerator in mind)? Is there only one vendor or are there multiple (is the setup dependent on one company providing spare parts or updates)?

The control system is (usually) distributed over the accelerator and the length of signal cables or bus cables might be a criterion for one architecture over another. As a rule of thumb: no signal cable longer than 30 m—some hardware has shorter limits. Sometimes this might not be achievable then tests have to make sure that longer cables work correctly. Cable length is usually not an issue for network

cables.

Different possible architecture layouts all come with positive and negative sides:

- **Centralized systems** have one central part that is connected to all other parts. New parts can easily be added as they only need one connection to the central part. However, if the central part fails, the entire system goes down, making it a single point of failure;
- Controls systems organised in **columns** are often the result when systems are build up one after another. For example there might be one system for the Linac, another for the Booster ring, and a third one for the storage ring in a synchrotron facility. The columns itself might be easy to set up as they do not need to provide for extensions. But it might be impossible to exchange information between different systems - beside the outdated way of using a slip of paper to note down important parameters;
- **Peer-to-peer networks** connect every part to every other part of the system. Such a network allows all parts to share information with each other. But if any part fails, the entire system is impacted, and new parts require changes across the board;
- Especially in large collaborations the **layer** architecture is common: the central team defines an interface and all collaborators have to provide their contributions (often hardware plus some expert software layer) to fit to that interface. Such a common interface is a huge advantage if you have to exchange one layer. But it can create tremendous trouble if the interface itself needs to be changed (for example due to new operating systems or IT developments).

The practical solution for most accelerator control systems is a mix of architecture layouts: layers that still have network and centralised components included are common.

The current "*standard control system*" consists of three tiers or layers (see Fig. II.12.2): a client layer for control room software, a network or glue layer for communication and rearranging information, and a server layer that communicates with the hardware (e.g. sensors, motors, actuators, etc.). When this terminology was developed in the 1990s the hardware was considered "dumb". Meanwhile, the hardware comes with its own controllers and chips with more computing power than whole server racks had before. Therefore, now we can talk about four-tier systems with a "hardware" layer capable of logic as well.

Each of the four layers or tiers can contain application logic: calculations can run everywhere. But where should they run?

In general it is advisable to move the logic as close to the hardware as possible. This makes feedback loops faster and prevents failure due to missing or unstable connection between the architectural layers. Of course, there are some problems that can only be solved at the higher levels because they need input from different parts of the accelerator or control system. Examples include comparisons of beam orbits with simulations, energy feedbacks (connected to many or all RF stations) in linear accelerators like FELs, alarm systems that monitor the overall status of the accelerator, etc.

II.12.2.1 Test questions

A3 Why is it important that accelerator control systems are designed for maintenance?

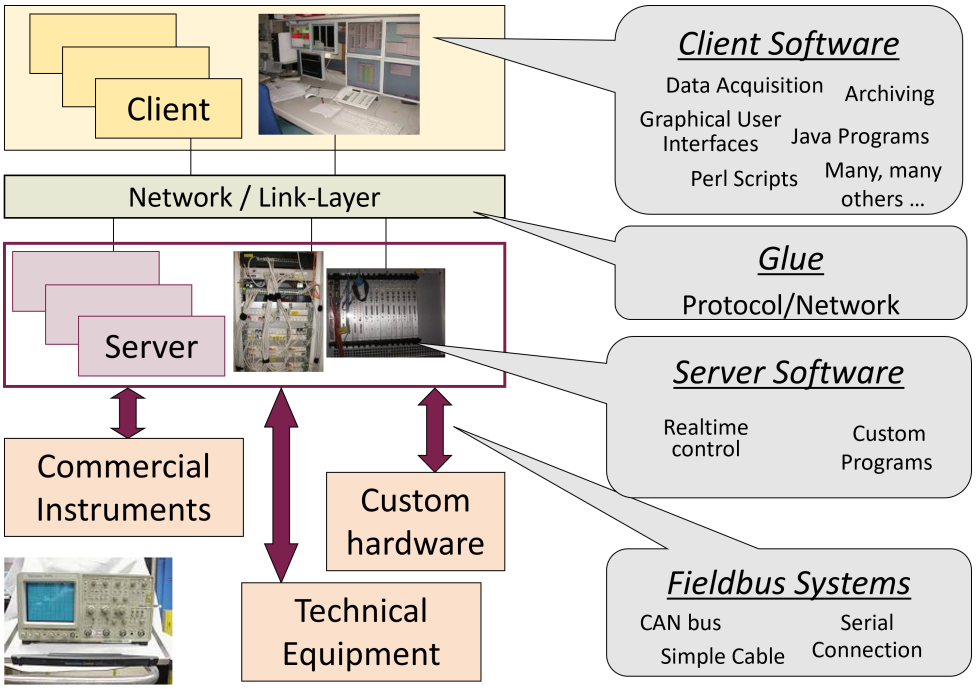

Fig. II.12.2: The Control-system Layer Model, sometimes called 3-Tier Model even though there are meanwhile four layers that can contain control system logic.

A4 Why are sometimes control systems designed as independent columns and what can be a disadvantage of such architecture?

II.12.3 Examples of control systems

The story of accelerator control systems began with the human brain. It served as a feedback loop linking the sensors (eyes) and actuators (potentiometers) when Donald Kerst tuned his first betratron in 1940 at the University of Illinois (USA) [4]. The next step were larger accelerators with dedicated control rooms where all potentiometers were multiplexed over long cables and "loading" a setup involved a list of set values in a paper notebook and patience while using whole walls full of turning knobs. Only in the 1970s computers got involved into running accelerators (mostly for simulating physics). Feedback loops were still implemented directly in electronics if they existed at all [5]. In the 1980s computers became smaller and more common—and all of a sudden it made sense to collaborate with other facilities by sharing software.

The first conference dedicated to accelerator controls happened in 1987 at CERN (late, compared to the first betatron in 1940). From then onward, every second year there is an ICALEPCS (International Conference on Accelerator and Large Experimental Physics Control Systems) conference. You can find out about the next conference on [6].

In general, accelerator control systems of different facilities can be grouped into three types: collaboration systems, single site systems, and commercial systems. Each type has its positive and negative sides, so there is no optimal solution for all cases. But in some cases there have been mixed systems, where for example, a TANGO beamline was run at an EPICS accelerator. So, the choice is not necessary

only one system—but of course the maintenance costs of two different systems should be taken into account when such a mix is chosen or allowed.

Collaboration systems: They are designed and used in more than one laboratory. Usually there is some kind of organisation around working together and sharing software. Examples are EPICS [7] and TANGO [8];

- Positive sides are less effort because a lot of software is already available. In addition, there are other people working on the same software and similar problems which enables discussions,
- Negative sides are the general type of software that has to fit "all" accelerators. Usually that includes areas where it does not fit well.

Single site systems: This type of system is designed for the use in one facility. Sometimes, such system is later shared with labs that collaborate with the facility on their soft- or hardware, but it was not originally meant for those labs. Single site systems are often the consequence of small facilities that do not have any contact to other accelerators [9];

- Positive sides are the perfect fit of the system for the problems of the facility. There is no reason to consider operating systems or hardware that is not used,
- Negative side is that the complete system has to be developed in-house. For bugs and problems there is no one to help or to even discuss with. This is especially problematic for small facilities with only two or three developers.

Commercial controls systems: Meanwhile, one can buy the needed software from a company. Some facilities decide to buy whole sub-components of the accelerator (like a linac for a synchrotron) or a complete experimental station. Such turn-key systems can come with a commercial control system if not specified differently (an example system would be [10]);

- Positive is that a commercial control system requires less work on the developers of the controls Groups—at least initially,
- Negative sides are the same as for all commercial solutions inside a control system: the lifetime of an accelerator surpasses that of any computer based system by an order of magnitude. With the likely use of proprietary components and software, the company providing the system might be needed for upgrades and changes. Of course, they will charge a substantial amount of money for every line of code.

Which type of system is the best fit for a new facility depends mostly on non-technical boundary conditions. For example: if a new accelerator is built in an existing institute that already has one working accelerator, it might be beneficial to use the same control system for both accelerators to minimize maintenance. This was the case when SwissFEL was designed at the Paul Scherrer Institute in Switzerland after the Swiss Light Source was already twelve years in user operation. Other reasons might be a close collaboration with another facility or if a substantial part of the system users are already familiar with a certain control system.

The next four sections will describe different control systems currently in use. They are examples

of the possibilities that new facilities can choose from today (year 2023).

II.12.3.1 EPICS

EPICS is the abbreviation for "Experimental Physics and Industrial Control System". It is the oldest collaboration type accelerator control system developed since 1989 when the collaboration was started by the Los Alamos National Laboratory (USA) and the Argonne National Laboratory (USA). In 2004 EPICS became open source and can now be found in Ref. [7].

EPICS uses Client/Server and Publish/Subscribe techniques to communicate between the various computers in the Control System. Most servers (called input/output controllers or IOCs) perform real-world I/O and local control tasks, and publish this information to clients using robust, EPICS specific network protocols Channel Access and pvAccess. These protocols are designed for high bandwidth, soft real-time networking applications that EPICS is used for, and is one reason why it can be used to build a control system comprising hundreds of computers.

EPICS is used for example in the following laboratories: Advanced Photon Source (United States of America), Diamond Light Source (United Kingdom), European Spallation Source (Sweden), ITER (France), NSLS-II (United States of America), Paul Scherrer Institute (Switzerland), SLAC National Accelerator Laboratory (United States of America). The EPICS collaboration includes more than hundred members all over the world.

II.12.3.2 Tango

Tango stands for "TAco Next Generation Objects" is a collaboration type control system that was developed based on the single-site system called TACO used at the European Synchrotron Radiation Facility (France). The collaboration started in 2001 with ESRF (France) and SOLEIL (France), and was from the beginning an open-source project. Downloads, trainings, and background information can be found in Ref. [8].

Tango is a device-oriented software toolkit that is written in the software languages C++, Java, and Python, and is therefore strictly operating-system independent. Originally Tango was build to use CORBA network objects but has been rewritten to use ZMQ to communicate between device server and clients.

The Tango community consists of more than fourty collaborators. Some of them are: ALBA (Spain), Deutsches Elektronen-Synchroton DESY (Germany), ESRF (France), MAX-IV (Sweden), Solaris (Poland).

II.12.3.3 DOOCS

DOOCS ist the abbreviation for "Distributed Object-Oriented Control System" and was developed as a single-site system for the FLASH at DESY (Germany) in 2008. Meanwhile, it is used for the European XFEL accelerator control as well (the photonics part is controlled by Karabo). More information can be found in Ref. [9].

Like Tango DOOCS facilitates a device-oriented view on the accelerator. It is written in C++ which supports close interfaces to hardware, and it uses RPC (Remote Procedure Call) to implement a

client-server architecture. As this control system was specifically written for an FEL it has an integrated support for timing related topics like triggers and precise clock information.

II.12.3.4 WinCC open architecture

WinCC Open Architecture is a commercial control system solution offered by the company Siemens and used at CERN [10]. Of course, this software is used in industry as well for non-accelerator related systems. The architecture is based on a SCADA (Supervisory Control And Data Acquisition) system that is very similar, but not identical to a distributed control system. Wikipedia provides a good comparison on differences and similarities of such systems [11].

CERN started using the predecessor of WinCC Open Architecture called PVSS II (Prozessvisualisierungs- und Steuerungssystem 2) in 1990s. At that time it was developed by a small Austrian company called ETM that got bought by Siemens AG in 2007. This history hints at the possible problems that might occur with commercial solutions: the company might decide to discontinue development or support of the software, especially if the company is sold or bought by other companies [12].

II.12.3.5 Test questions

A5 What are the three general types of control systems?

A6 Which accelerator control system is the best one?

II.12.4 Some individual components of control systems

The following sections discuss some of the components of an accelerator control system. The list is by no means complete, but concentrates on a few pieces of software and hardware that are important and need to be considered.

II.12.4.1 Client software

Client software includes all the applications and programs used in an accelerator control room. Some may be dedicated to a single experiment, while others may be used for a wide variety of tasks. The general question that needs to be asked is: on which level in the control system is the logic located?

The answers can be broadly grouped into three categories, which in turn define the type of client software that is required:

1. The logic is implemented directly in the hardware, for example in FPGAs (Field Programmable Gate Arrays), PLCs (Programmable Logic Controller), or other controllers that are just integrated "as-is" into the control system. The advantage lies in the fast execution, the disadvantage in the more laborious adaptability. In this case the client software is relegated to just display results or set calculation parameters;
2. The logic is implemented in software that runs at the server level of the control system, in most cases programmed in a language that is closely linked to the hardware, such as C or C++. The execution can still be real-time if the server's operating system supports it. Modifications and

maintenance can be difficult as hardware-specific knowledge is needed. Again, the client software is limited to just display results or set calculation parameters;

3. The logic is implemented in the client software. The advantage here is that the software is easy to adapt, debug, and change. On the other hand, execution is always slowed down by communication over the network.

In the first two cases, the client software is considered a **thin client**, while the third case requires a **fat client**. A thin client is just a display tool for the results calculated elsewhere. In contrast, a fat client calculates the result itself which usually leads to more complex code.

Thin clients are needed for applications that require hard real-time, or where users want unfiltered access to all the capabilities of the hardware. In this cases the clients are only GUIs (Graphical User Interfaces) without any application logic. They are most often implemented as GUI builders, where the panel creator can simply click together pre-defined widgets. Examples are JDDD (used with DOOCS at XFEL and DESY) [13], BOY (the user interface for the control system studio used in several EPICS-based facilities) [14] and caQtDM (used at PSI) [15].

Fat clients are often science applications (measurements, comparisons with simulations, etc.) that need complicated calculations which are difficult to implement without scientific and mathematical libraries. They are usually written directly by scientists rather than control system experts. Programming languages vary, but currently MATLAB and Python are dominant. A very powerful library, ROOT, has been developed by CERN [16]. It is originally written for C++ but can be integrated in Python as well.

II.12.4.2 "Server" hardware

This section discusses the computers connected to the hardware in the accelerator. In the 3-tier model they are referred to as servers (see Fig. II.12.2), although they have limited similarities to the common understanding of servers. In some facilities they might be called front end computer or just short front-end, it might be useful to check the terminology used in your lab.

In many cases, the application logic runs on these servers to provide real-time capability or to reduce the delays caused by the network. What these servers actually look like depends very much on the area of application. Broadly speaking, three categories of servers can be distinguished: high-end systems, PC-based systems, and embedded solutions.

High-end systems: These systems are used when speed of computation or feedback systems are important. They are based on very specialised hardware. The classic choice was VME (Versa Module Eurocard), an old standard that is no longer able to keep up with the increasing demands of accelerator systems, but which is still in use at facilities around the world. Its successors are not used in such a uniform way: μTCA (used at DESY, for example), ATCA for physics (used at SLAC, for example) and cPCI-s (used at PSI). All these hardware platforms support easy integration of FPGA-based cards and fast bus systems, and all are quite expensive;

PC-based systems: PCs or virtual machines are a cheaper solution, but they lack the real-time capabilities of the high-end systems. Many accelerator hardware components now come with a controller for the internal feedback loops and a network or serial interface for remote control and data read-back. With clear standardised interfaces such as the TCP/IP protocol, the shorter lifetimes are not

as important as the systems can be exchanged for their successors. Data rates are strictly limited in these systems and might only reach 10 Hz, which is for sure enough for the readback of a temperature, but not for fast-changing data like beam positions in single-pass accelerators;

Embedded solutions: These systems are sometimes combined with the PC-based readout systems mentioned above. Here the logic and any feedback loops run on chips (often FPGAs) embedded in the hardware. For accelerators, this raises the issue of radiation hardness of the chips, and the hardware may need to be specially placed (e.g. under heavy girders) or shielded. The biggest disadvantage is the tedious task of upgrading an embedded operating system to the next version. Often the chips used are specialised and require certain libraries to be available—which may not be the case in the future. On the other hand, compact design may be preferred in tight spaces. Embedded solutions need to be thought through from all angles and require good local support throughout their life.

II.12.4.3 Hardware—the fourth tier

The fourth tier was introduced in section II.12.2. It includes any hardware that has or uses some form of built-in logic, which usually means it contains programmable electronics. Of the various systems, three will be discussed here: FPGAs, PLCs and fieldbus systems. Of course, this list is not exhaustive.

FPGA (Field Programmable Gate Arrays): This category overlaps with the embedded systems discussed in the last section. FPGA-based electronics can perform feedback loops very fast (microseconds or better), but maintenance can be a problem: the FPGA code is written in a hardware description language that is chip-specific and needs its own development environment (running on a PC). If the chip changes or if the development environment cannot be upgraded to the latest PC operating system, the functionality has to be redesigned;

PLCs (Programmable Logic Controller): PLCs are industrial computers adapted for deterministic control loops. They are essential for interlock and safety systems and can handle systems with thousands of inputs and outputs. The speed of the program loop is currently limited to a few milliseconds. Unfortunately, products from different manufacturers cannot be mixed and even modules from the same manufacturer may not be backwards compatible, resulting in whole systems being replaced as updates;

Field-bus systems: Field busses connect the hardware in the fourth-tier with the server layer (see Fig. II.12.2). These busses use real-time protocols for distributed communication and are standardized as by the International Electrotechnical Commission as IEC 61784/61158. Commonly used examples are Profibus, MODBUS, and EtherCAT. They differ in the maximal number of possible devices, the speed, the allowed cable length, and possible topologies (e.g. ring, star, linear). USB and Ethernet are no field busses, as they do not follow the above mentioned standardization. Nevertheless, they are used in accelerator control systems in places of field busses when the real-time capability is not needed.

II.12.4.4 Test questions

A7 Under what circumstances would you use a fat software client?

A8 If you are looking to integrate a new beam-loss monitoring system that supports an embedded

control system, what do you need to consider?

II.12.5 Summary and conclusion

The control system is a central part of all accelerators. Its main task is to integrate the autonomous controllers distributed throughout the facility into a coherent infrastructure.

The control system provides an abstraction layer between the hardware and the operators, and creates the environment in which scientists can carry out their experiments. It also allows the technical support groups to compare past data with current data to improve the performance of their systems. Ease of use, reliability and security are the guiding principles for the design of an accelerator control system. Beyond pure functionality, maintainability of the whole system and portability for new developments in computer science are basic requirements. This is additionally complicated by the vastly different lifecycles of accelerators and computer equipment.

To reiterate the definition from the first section of this chapter: an accelerator control system is a computer environment that allows remote access to the accelerator hardware with a lot of different functionalities to satisfy the requirements of several different user groups.

A basic knowledge of how such control systems work is useful to optimise their use and to be aware of their limitations when carrying out experiments. All devices that need to be monitored or controlled during the operation of an accelerator must be integrated into the accelerator control system.

More detailed and historic information about accelerator control systems have been published in 2008 in the “Beam Dynamics Newsletter” No.47 by the International Committee for Future Accelerators [17].

II.12.5.1 How to become a controls expert

What do controls experts do?

We have roughly three different roles in controls: the integrators work closely with the expert groups or the experiment group and integrate the hardware of these experts into the control system. The integrators might work (rarely) in the control room but more often in test hutches or labs. The second groups are back-end programmers that care for data pipelines, deployment tools, and special applications for operators or beamlines. They work at their computers, in office or from home. The third group are infrastructure experts, that care for installation of computers and blades. They work sometimes in server rooms, but mostly in their offices.

What are the requirements to become a controls expert?

- Be curious about what your customers do (accelerator physics, experiments, medical treatment, etc.);
- Enjoy programming and be proficient in
  - One script language (most used is currently python),
  - One object-oriented language (Java, C++, etc.),
  - Experience in programming close to hardware is always welcome,

– Enjoy working with computer environments in general.

Additional useful skills include (all of the following is non-essential):

– Basic knowledge in accelerator physics or general physics,
– Linux and/or Windows operating system administration,
– Network administration,
– PLC, FPGA or DSP programming (nearly electronics),
– Graphical User Interface design and usability theory/praxis.

Is a master degree enough? Yes. And it does not even need to be in physics, it can be in computer science or engineering as well. Of course, a background in physics, especially in accelerator physics, helps to understand what we are programming. It makes discussions with scientists easier. There are many ways that lead into the control system, and we tend to be a diverse group concerning our backgrounds.

II.12.6 Answers to test questions

A1 The accelerator control system does only provide a keyhole view of the accelerator to the user: everything that is not connected to the control system can not be seen and used. In addition, some user groups like accelerator physicists or system experts want to access all functions that the hardware provides. Therefore, including all functions into the control system is advisable.

A2 A vacuum system expert wants to access a lot—or even all—functionalities of the installed hardware. And the expert wants to have the equipment easy to maintain, as the life cycle of the accelerator will for sure be longer than the maintenance cycle of the controlled hardware.

A3 As the lifetime of an accelerator is an order of magnitude higher than the average lifetime of IT components (Hard- and Software), the IT systems will need to be changed several times. Therefore, the design for maintenance will save effort during the whole accelerator lifetime.

A4 Independent column systems are often a result of sequentially build up systems. It happens when accelerators are later extended and the extensions get their own (modernized) control system instead of being incorporated into the older one. The consequence can be a limited intercommunication between the column systems which might limit the performance and chance for more automatization.

A5 Each control system belongs to one of the following three types: collaboration systems, single site system, and commercial control systems.

A6 All control systems described in section 3 enable the successful operation of accelerators. Therefore, they are fulfilling their specifications and deliver results. A control systems might be better suited for a certain purpose or style of setup, but all of them are equal in terms of technical quality. The reasons to prefer one over the other are often found in facility management (e.g. being part of a collaboration) or user preferences (“if most of my users already know a system, it makes sense to use it as well”).

A7 Fat software clients are used for two reasons: when you have complex computations that require scientific or mathematical libraries to implement, and when you are not yet sure that the computation you are implementing is what you will need in the long run. The latter use is often part of a more lengthy development process, where you test the algorithm first and then port it later to the server level, where it is harder to change. In general, you use a fat software client when you need to implement the logic and calculations of your application at the client level of the control system.

A8 Embedded control systems have a general problem with radiation hardness and maintenance of the software running embedded in the system. You need to consider the physical location of the devices containing the electronics, perhaps they need extra shielding or can be placed under some iron beams or concrete slabs. The lack of easy maintainability requires someone dedicated to updating and maintaining the equipment and software. Good documentation would also be an advantage.

References

[1] Distributed control system, Wikipedia, (retrieved 2023-10-18).

[2] Red Queen's Race, Wikipedia, (retrieved 2023-10-18).

[3] Microservices, Wikipedia, (retrieved 2023-10-18).

[4] Picture of Professor Donald Kerst making adjustments on the first betatron, University of Illinois, https://distributedmuseum.illinois.edu/exhibit/betatron/, (retrieved 2023-10-18).

[5] Picture of the PS controls room at CERN, Feb. 1974, https://cds.cern.ch/record/916847, (retrieved 2023-10-16).

[6] List of ICALEPCS conferences, https://www.icalepcs.org/conferences.html, (retrieved 2023-10-16);ICALEPCS proceedings, JACoW (retrieved 2023-10-16).

[7] EPICS homepage, https://epics-controls.org/, (retrieved 2023-10-18).

[8] TANGO homepage, https://www.tango-controls.org/, (retrieved 2023-10-16).

[9] DOOCS homepage, https://doocs.desy.de, (retrieved 2023-10-16).

[10] SIMATIC WinCC Open Architecture product information, https://www.winccoa.com/product-information.html, (retrieved 2023-10-18).

[11] SCADA (Supervisory Control And Data Acquisition), Wikipedia, (retrieved 2023-10-18).

[12] History and background of the company ETM which developed PVSS, the basis of todays WinCC-OA, https://www.winccoa.com/company.html, (retrieved 2023-10-18).

[13] Links to the documentation and repository of the GUI builder JDDD, https://doocsweb.desy.de/index.html#documentation. (retrieved 2023-10-16)

[14] Introduction to the GUI builder BOY, https://github.com/ControlSystemStudio/cs-studio/wiki/BOY. (retrieved 2023-10-16)

[15] Homepage of the GUI builder caQtDM, http://epics.web.psi.ch/software/caqtdm/. (retrieved 2023-10-16)

[16] ROOT: An open-source data analysis framework, https://root.cern.ch/. (retrieved 2023-10-16)

[17] ICFA Beam Dynamics Newsletter Number 47 (December 2008) on Control System, https://icfa-usa.jlab.org/archive/newsletter/icfa_bd_nl_47.pdf, (retrieved 2023-10-16)