ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA06

CONFIGURATION MANAGEMENT FOR THE INTEGRATED CONTROL SYSTEM SOFTWARE OF ELI-ALPS

Lajos Schrettner*, Balázs Bagó, Balázs Erdőhelyi, Tamás Gaizer, Attila Heidrich, Gergely Nyiri
ELI-ALPS, Szeged, Hungary

Abstract

ELI-ALPS (Extreme Light Infrastructure - Attosecond Light Pulse Source) is a new Research Infrastructure under implementation in Hungary. The infrastructure will consist of various systems (laser sources, beam transport, secondary sources, end stations) built on top of common subsystems (HVAC, cooling water, vibration monitoring, vacuum system, etc.), yielding a heterogeneous environment.

To support the full control software development lifecycle for this complex infrastructure a flexible hierarchical configuration model has been defined, and a supporting toolset has been developed for its management. The configuration model is comprehensive as it covers all relevant aspects of the entire controlled system, the control software components and all the necessary connections between them. Furthermore, it supports the generation of virtual environments that approximate the hardware environment for software testing purposes. The toolset covers configuration functions such as storage, version control, GUI editing and queries.

The model and tools presented in our paper are not specific to ELI-ALPS or to the ELI community, they may be useful for other research institutions as well.

INTRODUCTION

The primary aim of the ELI-ALPS research facility, currently under implementation in Szeged, Hungary, is to make a wide range of ultrafast light sources accessible to the user groups of the international scientific community. Laser driven secondary sources emitting coherent extreme-ultraviolet (XUV) and X-ray radiation confined in attosecond pulses is a major research initiative of the facility. The primary laser pulses will be provided by laser sources operating in the regime of 100 W average power in the near-infrared (NIR) and at 10 W in the mid-IR (MIR).

The constructed buildings will house the laser equipment, beam transport, secondary sources, target areas, laser preparation and other special laboratories. These state-of-the-art facilities require specialized design and cutting edge implementation of the latest technology for vibration levels, thermal stability, relative humidity, clean room facilities and radiation protection conditions.

The typical layout of the laser systems are shown in Fig. 1: each laser source is connected via a beam transport system to one or more beamlines consisting of a secondary source and an end station. Each laser system can contain a high number of controlled devices (translation stages, motorized mirrors, cameras, vacuum pumps, valves, gauges, etc.).

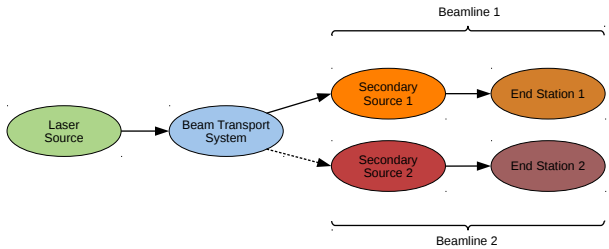

Figure 1: Simplified view of a laser system.

The control system as a whole consists of several interconnected and interoperating systems, each of these having several subsystems, such as vacuum control, optical alignment or an optical configuration subsystem. Each subsystem has a layered architecture, on the bottom being the hardware devices themselves. Above them are the logical devices, which are software components representing a single hardware device on the software side. The functionality of several logical devices are collected in higher level components as necessary, and a resulting system has a single service component representing it towards the rest of the system complex. The Central Control System uses the services of all the other systems and is responsible for monitoring, resource allocation, archiving, alarm handling, etc.

CONFIGURATION IN CONTROL SYSTEMS

Background

Typically configuration plays an important role in the command and control ecosystem, its purpose is to give a static description of properties and relationships of the relevant components. The command and control system accesses these settings from a configuration repository.

The configuration of an advanced control system should describe the system as complete as possible from a structural aspect. On one hand the main goal of the control system is to control hardware devices and various equipment contained in the facility. Networks, computers, motors, CCD cameras and other detectors build up the hardware equipment that has to be controlled, therefore the description of the hardware equipment must be part of the configuration. On the other hand the user requirements must be implemented in the control system. These are usually high level requirements regarding functionality, appearance, expert-mode vs. automated mode, logging, alarms, GUI and/or CLI, data visualization and analysis, etc. The high level architecture of the software that is being developed to implement the requirements of the users can be described in the configura-

* lajos.schrettner@eli-alps.hu

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA06

tion as well. Based on the above, it is natural to differentiate components related to hardware and research infrastructure and that of the software and control system.

Additionally, it is advantageous if configuration has features that assist in covering the full development process lifecycle, cope with regular changes and updates. In particular it should be possible to design, develop, test and to some degree operate the control system even if the hardware to be controlled is partially or completely unavailable. A possible solution to this problem is virtualization, that is the ability to replace some or all of the hardware devices with virtual (software) devices that in all relevant aspects provide the same interface and behaviour as the hardware devices they stand for. Virtualization will always be a (mostly rudimentary) approximation to the real behaviour of the devices, as completely realistic simulation is neither desirable nor attainable in a cost-effective way.

Configuration in the Control System Lifecycle

Based on the observations above, we can determine that the lifespan of the data that is used in or related to a control system can be categorised as follows:

- Transient: runtime data, that is only available during one single execution of the control system software.
- Persistent: Values of this kind must be preserved between consecutive executions of the control system, e.g. saved motor positions or detector settings.
- Permanent: static data that does not change for several executions of the control system and modifications are performed only when the control system is down, e.g. position and size of the buildings, rooms, computers.

In our opinion that this is the category of data that can and should be covered by configuration.

Concentrating on the software development phases, configuration is used mainly under the following circumstances:

- During the design of the control system.
- Creation of a virtual environment: in the development phase of the control system, for manual or automated testing in a Continuous Integration environment.
- Deployment in a virtual environment or on phisical hardware in the target environment.
- During operation (in either a virtual or target environment): devices can discover their connections, synaptic views can be generated from the configuration, etc.

Related Work

Different institutions used configuration management on various data and for different purposes. Hardion et al. [1] consider data from the operating system to Tango device properties as part of the configuration.

Krempaská et al. [2] use an installation tool (called swit) to connect to a database and depending on the information retrieved, the right files (or symlinks) are created to install the operating system, libraries or EPICS configuration files to the Input Output Controllers (IOCs).

Mader et al. in [3] regard only EPICS process variables and values as part of the configuration of their telescope control system. In contrast, in [4, 5] the configuration database contains information from the hardware, software and even up to the GUI. Special tools are considered to be part of the configuration as well, like Data Editing or Data Browsing Interfaces, APIs and Scrips, Data Security, Testing, etc.

Zaharieva et al. describe a system in [5] where the actual configuration is sent back to the central database through an online feedback mechanism, which enables detection of differences compared to the stored configuration.

Makeev et al. presented a software configuration tool in [6] that combines the advantages of centralized and decentralized approaches. They also state that their Centralized Storage is a graph-based database.

Beltran et al. created a database for cabling installation, and later extended it to be the central repository for the whole computing installation [7, 8].

Proteus was introduced by Vuppala et al. [9] to manage configuration data during design, commissioning, operation, and maintenance.

ELI-ALPS CONFIGURATION MODEL

Requirements

After studying existing solutions and drawing conclusions during the development process taking place at ELI-ALPS, we came up with the following requirements for our configuration model.

Content related: The model must represent all items of interest, in particular:

- Representation of physical reality.
  - Space subdivision, locations (building, area, room, chamber, etc.).
  - Hardware for executing the control system (networks, computers).
  - Hardware to be controlled (sensors, actuators).
- Representation of the control system structure
  - Representation of high level software components of the control system (currently TANGO device servers and devices).
  - Representation of virtualization elements (i.e. replacement of controllable hardware with software components).
- Representation of all relevant connections between the above.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA06

Usage/process related: The model should support the following (sub)tasks that occur frequently during development:

- Identification: unique, comprehensible identifiers for all elements in the configuration. These identifiers will be used by users in communication to refer to the elements.
- Data extraction API: Standardized interface for creating tools that access the configuration data in a uniform way.
- Integrity/consistency checking: it should be possible to define and automatically check constraints.
- Storage in a text-based human-readable format: for occasional low level manual editing and to enable version control.
- Creation/editing: text editor, custom graphical editor to enhance user experience and productivity.

Implementation

In the following, we describe the solution we came up with to satisfy the requirements above.

Following the structure of the ecosystem, the configuration model also consists of three main parts to model the corresponding parts of the ecosystem (that is, physical equipment, control system, virtualization – see Fig. 3). Besides, the model contains elements that are common in the three sections.

A configuration metamodel defines the possible elements, connections, and integrity rules of the configuration model. An important advantage of the metamodel is that it makes it possible to impose data integrity rules. This is made possible by defining the optional and mandatory attributes of elements and connections, also defining constraints for the valid node-edge combinations.

There is a primary hierarchy (a directed tree) within each section. As the sections serve different purposes, these primary hierarchies have quite different interpretations (see explanation below). Following the hierarchical structure, each node can be identified by its path starting from the root node similar to a directory tree structure. This type of identification is unique as long as there are no identical node names for siblings in the hierarchy and also easy to use for humans and machines as well. Besides the primary hierarchy there are other connections (edges) among the configuration elements. Edges are not restricted to one section, they may also connect elements of different sections.

The flow of configuration data is illustrated in Fig. 2. It has two representations, a physical one called the configuration database and a logical one called the configuration model (ConfigModel for short). The purpose of the configuration database is only to store the configuration data, the actual use of data takes place through the configuration model interface. Any time a software component of the ecosystem is started and uses some element of the configuration, it first loads the configuration database into a compound internal object structure (the ConfigModel). The ConfigModel hides the physical implementation of storage from the "users" (that is, the software components using the model), they access model elements only through the properties and methods of ConfigModel objects.

The ConfigModel is a directed graph, consisting of nodes and edges. Nodes represent the elements of the ecosystem (devices, networks, hosts, etc.) and edges represent the connections between them (e.g. physical or logical containment, propagation path of light between components, etc.).

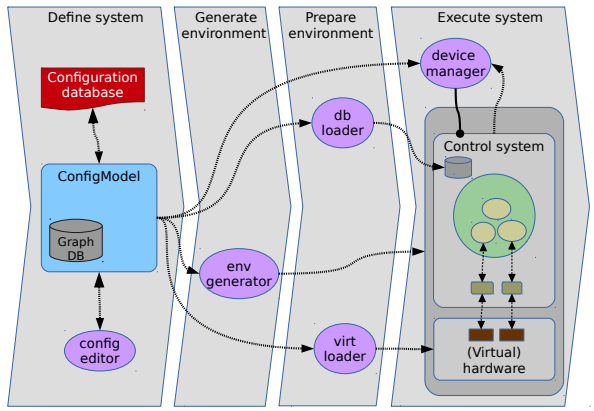

Figure 2: Configuration model in the operating environment.

Main Model Sections

Below we summarize the most important elements of the model.

Equipment The equipment section describes the physical world, that is the building, rooms, chambers and the hardware devices contained in them. The basic hierarchy among the elements is the physical containment relation: the rooms are contained within the building, vacuum chambers within the rooms, etc. The equipment section also defines so called logical configurations which are the possible states for the translation stages resulting in paths for the laser beam.

- Space subdivision:
  - building: Defines a building in the equipment section.
  - area: Defines an area inside the building.
  - room: A room in a building or area (typically a clean room).
  - chamber: A vacuum chamber, it generally contains other elements (devices, translation stages, etc.). Chambers may be connected to each other if there is a vacuum pipe between them in the physical world.
- Control hardware:
  - network: Description of the network inside the equipment section.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA06

  - computer: Defines a physical computer. Computers are located in rooms, or racks, and hosts can be deployed on them.

- Controlled hardware:
  - device: Physical device, which can be controlled either directly or indirectly. Devices may have optical connections which define the path of the laser beam.
  - controller: Controllers are connected to the controlled devices by special connections.

Control System The control system section describes structure of the control system from a software architectural point of view defining the logical layout of the deployed software (containers, components) and the network interfaces of the containers to networks defined in the equipment section. The hierarchy of elements is the logical containment between the components (e.g. hosts are contained within a region, software components are contained within a host, etc.).

- region: a logically isolated part of the Control System (which corresponds some larger system in the physical world, such as the Beam Transport System)
- host: Describes a host, that may include software containers and network interfaces to the network of the controlled devices.
- container: a logical group of software components within a host that can run independently of other similar units. Containers make it possible to define a starting sequence on device drivers, for example. In our current implementation the containers are Tango Device Servers.
- component: a software unit which we do not want to subdivide further statically. A component may be the software counterpart of a physical device, or it can provide functionality to other components. In our current implementation the components are Tango Devices.

Virtualization The virtualization section describes software components for virtualization of the hardware elements of the equipment section. The hierarchy of elements is similar to the control system section.

- Virtualization region: same as the role in control system section
- Virtualization host: similar as the host in the control system section except that this element contains virtualization services instead of containers
- Virtualization system: a software component responsible for virtualization of a subtree of the equipment section

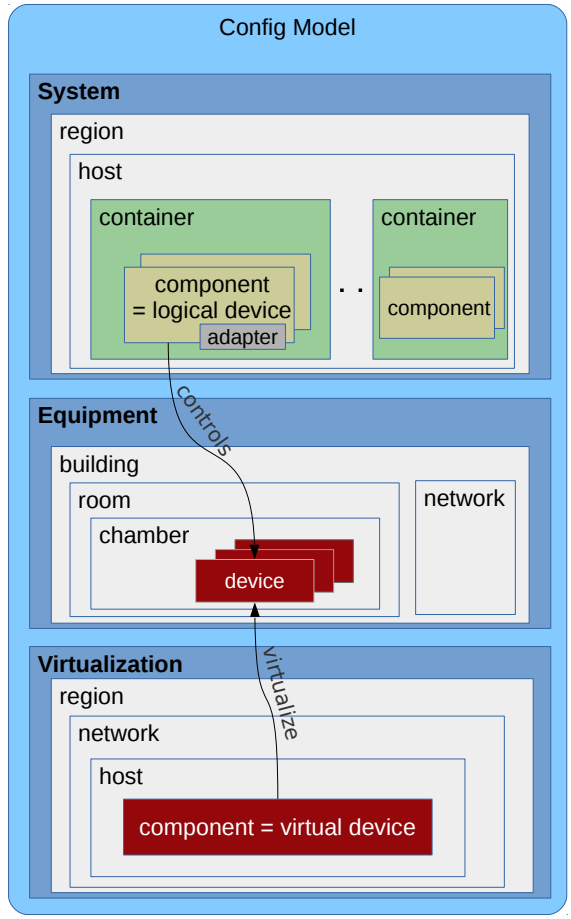

Figure 3: High level structure of the configuration model.

Connections Between Elements of Different Sections

The following connections span across different sections of the configuration.

- controls: A link starting from the controlling software component pointing to the controlled physical device.
- virtualize: The connection between the simulation device and the physical device being simulated.

The controls and virtualize connections are shown on Fig. 3

CONFIGURATION MANAGEMENT TOOLKIT

ConfigModel API

The API of the Config model provides methods to query and modify the nodes and connections of the Config model. The method calls that modify the content are propagated to the graph database backend. The API provides validation and consistency checking of the model through the usage of a previously defined metamodel. Based on these metamodel rules, automatic checking model instances is possible, and

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA06

carried out for example when the configuration database is read or written, and when the configuration editor is used.

Database Backend

The content of the configuration contains two kind of data: components and the connections between them. Components are the basic building blocks of the ecosystem (like hardware devices, software processes, hosts, networks, virtualization configurations, etc.). Connections represent some kind of relationship between components. Examples of this relationships are: spatial or logical containment, control, virtualize, optical path to, wired to, etc. Both components and connections can have arbitrary attributes.

The data structure above is basically a graph where components are nodes, and connections are edges, and as such it is best stored in a graph database. For our purposes we have chosen the neo4j graph database [10].

The graph database is filled with data from the configuration database (see Fig. 2), which is stored as a XML file. Keeping the configuration in a text-based, human readable format has the advantage of low level editing and that it is easy to add to version control systems.

ConfigEditor

A dedicated application is being developed for the visual construction and maintenance of configuration descriptions. The name of this application is ConfigEditor (see Fig. 4). As shown in Fig. 2, the ConfigModel serves as input and as output for the application. The aims of the ConfigEditor are as follows:

- Visualize the items of the configuration. This visualization includes a classical tree view, and a special hierarchical view where rectangles are embedded into each other. The latter is preferred since it allows the visualization of connection links beside the hierarchy.
- Edit the configuration structure: add and remove elements to existing ones, add and remove connections
- Add, remove and edit the attributes of any element or connection.

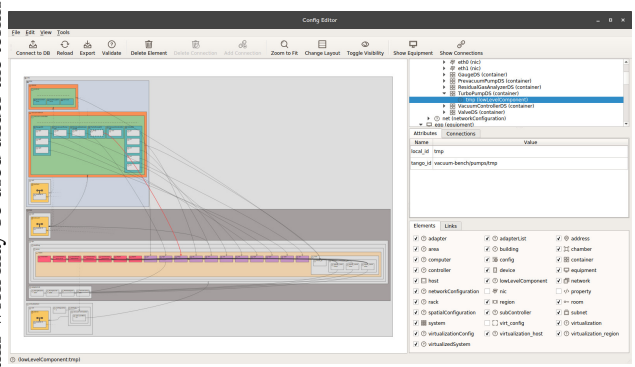

Figure 4: ConfigEditor application to visualize and edit a configuration.

Virtual Environment Generator

The creation of the virtual environment has the following steps (see Fig. 5):

The Environment Generator software uses the information from the Config Model to generate Environment description file. This file contains the description of the networks, virtual hosts, environment variables, and other data that is necessary to create the Virtual Environment. The description file uses docker-compose's format to specify the virtual environment.

Docker-compose [11] is used to run the Virtual Environment. Its input is the description file, and it creates the containers and networks of the Virtual Environment.

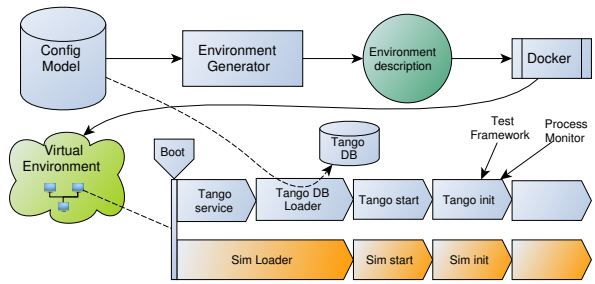

Figure 5: Virtual environment creation.

Virtual Environment Loader

The aim of the virtual environment loader is to start the software processes which take the place of physical devices. IP address, port, channel index, axis, serial number and other device specific data is passes to the virtual device. After the virtual device is started, it behaves, in all relevant aspects, like the hardware device it is simulating.

DB Loader

The aim of the DB Loader Tool is to prepare the control environment. The tool is run on every host defined in the equipment section, and it registers all the software containers and components into a Database. In our current implementation Tango ecosystem is used, and so the containers become Tango Device Servers, components turn into Tango Devices and registration is done into the Tango Database. This step enables the Tango starters to start the required software automatically and also provides means to pass additional properties to the Tango Devices via the Tango Database.

Device Manager

A dedicated application is being developed for system monitoring and management purposes. The primary goal is to supervise the Tango ecosystem by:

- monitoring the status of the Device Server processes and Tango Devices
- starting/restarting Device Server processes

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THBPA06

- setting the status of Tango Devices

The application can also be used for some diagnostic purposes such as validating the output of the Configuration Process. The tool has both CLI and GUI (see Fig. 6).

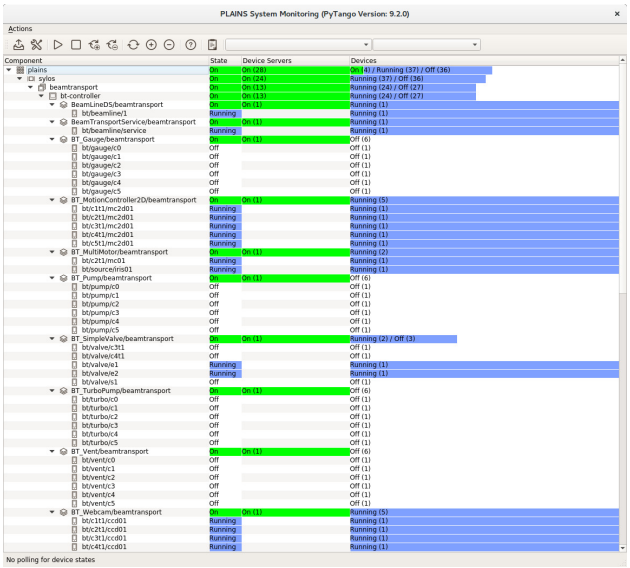

Figure 6: Device Manager GUI.

USAGE SCENARIOS

Typical usages of the configuration are summarized in the following.

- During software development, the control system is started and executed on the local machine of the developer. The networks and computers described by the configuration are substituted by docker networks and docker instances, devices are virtualized. The virtualized devices must be functional before the control system is started, since the control system has no information whether the underlying devices are virtualized or physical ones. As shown in Fig. 3, the software components (logical devices) use adapters to implement communication with the physical or virtual device.

- Automated test for continuous integration. This is the same as the above except that instead of a single developer machine the system uses multiple virtual machines as dictated by the Equipment section of the configuration description.

- Deployment in production environment differs from the above process since some or all of the devices are already available as physical devices. In this case, the real addresses (IP, port, channel, axis, etc.) of the devices have to be provided for the control system. To switch from development to production environment, only the adapters are to be replaced. Since the business logic is implemented in the component, the adapters remain only a thin software layer that translates between the component and the device.

CONCLUSION

Some form of configuration is typically used in every control system, although the term does not have a well defined meaning in every context. In this paper we summarized the usage scenarios when configuration appears in the literature and attempted to give the term a more strict interpretation by roughly identifying configuration with permanent data that has a lifespan comparable to the installed control system. Further, we compiled a list of requirements that should be fullfilled by the configuration subsystem, among these are ones that relate to the specific types of elements and connections between these elements, as well as others that relate to the typical usage scenarios of the configuration. Based on the requirements, we gave details of the complex solution we developed for the configuration of the ELI-ALPS control system. The configuration allows us to model all the relevant aspects of the research infrastructure as well as the control software structure. It supports all the usage scenarios that are important during the development of the control system, among these are the support it gives in the design phase by a graphical configuration editor, the assistance it provides during development and testing by being able to virtualize the equipment to be controlled, and as supporting database for guiding deployment. The development of the control system and configuration management in ELI-ALPS is ongoing, and we expect that the overall approach and high level structure will cover our future needs.

REFERENCES

[1] V. Hardion *et al.*, "Configuration Management of the Control System", in *Proc.* ICALEPCS'13, pp. 1114-1117.

[2] R. Krempaská et al., "Control System Configuration Management at PSI Large Research Facilities", in *Proc.* ICALEPCS'13, pp. 1125-1126.

[3] J. Mader et al., "Database-backed Configuration Service", in *Proc.* ICALEPCS'13, pp. 627-629.

[4] Z. Zaharieva et al., "Database Foundation for the Configuration Management of the CERN Accelerator Controls Systems", in *Proc.* ICALEPCS'11, pp. 48-51.

[5] Z. Zaharieva, S. Jensen et al., "Advantages and Challenges to the Use of On-line Feedback in CERN's Accelerators Controls Configuration Management", in *Proc.* ICALEPCS'13, pp. 600-603.

[6] A. Makeev et al., "Centralized Software and Hardware Configuration Tool for Large and Small Experimental Physics Facilities", in *Proc.* ICALEPCS'13, pp. 591-593.

[7] D. Beltran et al., "ALBA Control & Cabling Database", in *Proc.* ICALEPCS'09, pp. 423-425.

[8] S. Rubio-Manrique et al., "A Bottom-up Approach to Automatically Configured Tango Control Systems", in *Proc.* ICALEPCS'11, pp. 239-241.

[9] V. Vuppala et al., "Proteus: Frib Configuration Database", in *Proc.* ICALEPCS'13, pp. 623-626.

[10] `https://neo4j.com/`

[11] `https://www.docker.com/`