CORE
Metadata, citation and similar papers at core.ac.uk
Provided by DIFFER: Publications

Software architecture for control and data acquisition of linear plasma generator Magnum-PSI

P.W.C. Groen, V. van Beveren, A. Broekema, P.J. Busch, J.W. Genuit, G. Kaas, A.J. Poelman, J. Scholten, P.A. Zeijlmans van Emmichoven

FOM Institute DIFFER - Dutch Institute For Fundamental Energy Research, Association EURATOM-FOM, Partner in the Trilateral Euregio Cluster, P.O. Box 1207, 3430 BE, Nieuwegein, The Netherlands, www.differ.nl.

Abstract

The FOM Institute DIFFER - Dutch Institute for Fundamental Energy Research has completed the construction phase of Magnum-PSI, a magnetized, steady-state, large area, high-flux linear plasma beam generator to study plasma surface interactions under ITER divertor conditions. Magnum-PSI consists of several hardware subsystems, and a variety of diagnostic systems. The COntrol, Data Acquisition and Communication (CODAC) system integrates these subsystems and provides a complete interface for the Magnum-PSI users. Integrating it all, from the lowest hardware level of sensors and actuators, via the level of networked PLCs and computer systems, up to functions and classes in programming languages, demands a sound and modular software architecture, which is extendable and scalable for future changes. This paper describes this architecture, and the modular design of the software subsystems. The design is implemented in the CODAC system at the level of services and subsystems (the overall software architecture), as well as internally in the software subsystems.

Keywords: Magnum-PSI, plasma generator, data acquisition, CODAC, experiment control

1. Introduction

The FOM Institute DIFFER is completing a new apparatus to study plasma wall interactions in the parameter regime relevant for ITER and beyond. This linear machine, Magnum-PSI, consists of a series of vacuum chambers with on one end the plasma source and on the other the target manipulator [1, 2]. The plasma is magnetized by an axial magnetic field and is transported from the source to the target. Cooling is an important aspect of the device, since up to a few 100 kW can be dissipated [3]. An extensive description of the status of the Magnum-PSI device is given by [4]. Several diagnostics have been implemented to study the properties of plasmas, plasma wall interactions, and targets after plasma exposure. The first experiments have been performed [5, 6]. Safety plays an important role in running the machine, because of the presence of large magnetic stray fields, handling of hydrogen and deuterium, and use of intense laser beams. The complexity of the device makes a central COntrol, Data Acquisition and Communication (CODAC) system necessary. At the start of the Magnum-PSI project in 2005, a broad experience in in-house development of software was available, based on earlier projects like the Rijnhuizen Tokamak Project (RTP), see e.g. [7]. It was therefore decided to continue this full in-house development.

The Magnum-PSI CODAC system consists of services, subsystems (the cooling, vacuum, magnet, target station, and plasma source systems) as well as diagnostics [8] that have their software representations in the CODAC software architecture. Hereafter the term 'subsystems' is mainly used for the software part of subsystems. The term 'systems' will refer to both software subsystems and services.

To handle the complexity, and expandability of Magnum-PSI CODAC, a general setup is needed, which is realized by a modular design. This design applies to both the software architecture, and to its subsystems.

The authors of [8] treated the overall Magnum-PSI CODAC system architecture, with a strong emphasis on technology choices, the hardware, and the communication layers. This paper will place emphasis on the software architecture and will then zoom in on the software subsystem design.

The CODAC systems of devices like JET [9], ASDEX Upgrade [10, 11], TEXTOR [12], and W7-X [13] contain elements comparable to that of Magnum-PSI: complexity, local and central control, distribution of systems, flexibility, extendibility, and a major role for safety. It seems, however, that most CODAC systems are custom made for the device in question. The 'local components' of [13] and the 'Application Processes' of [10, 11] are similar to the subsystems in Magnum-PSI. Very interesting also on a more detailed level of building blocks for distributed control systems, next to JETRTApp [9], are FireSignal [14, 15] with 'Nodes' resembling subsystems, the tool kit TANGO [16] and the set of tools and applications called EPICS [17], on which the ITER CODAC Core System is based

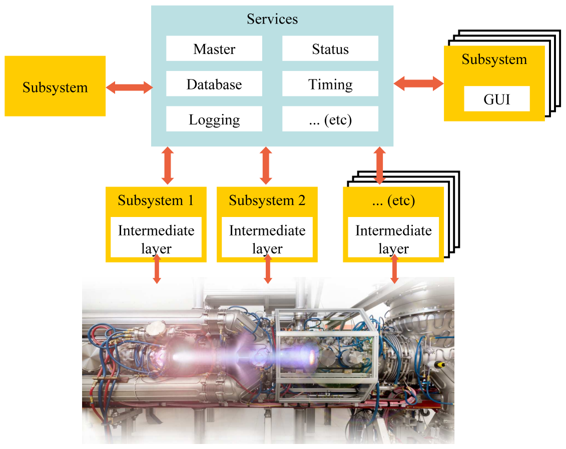

Figure 1: The Magnum-PSI CODAC software architecture with central services, subsystems and the Magnum-PSI device.

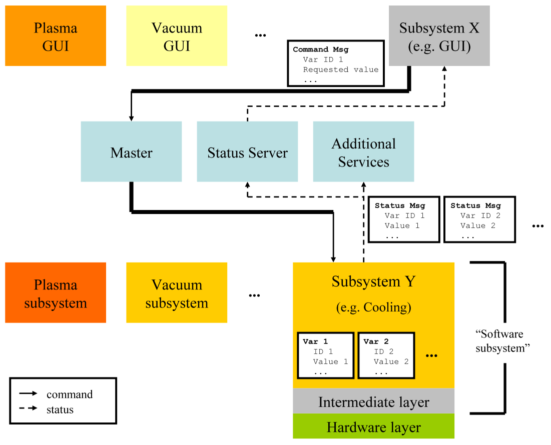

Figure 2: The communication within the Magnum-PSI CODAC software architecture via command and status messages.

[18]. In literature there is however somewhat less focus on the details of the design for the software of subsystems, which makes it difficult to compare to Magnum-PSI.

In the following two sections the software architecture for the Magnum-PSI CODAC and the software subsystem design will be described. A concise overview of its implementation, and choices made, will be given in the last section.

2. The architecture

The Magnum-PSI CODAC software architecture contains services and subsystems (fig. 1). The **Master** and the **Status service** are the main services. **Additional services** are the database, timing, and log services. The services are surrounded by (software) **subsystems**. More services and subsystems can easily be added. The control of subsystems is governed by setting values of data structures called **variables**. Systems communicate via messages, that refer to a variable. A system can send a request to set the variable's value via a **command message** to a subsystem that handles the variable internally (section 3). A subsystem shares its own variables' actual values with the other systems via **status messages**. Subsystems do not communicate directly to one another, but only via the Master and Status service. Command messages always pass through the Master. Status messages go directly to the Status service. Fig. 2 shows the architecture in an operational context with the communication via messages.

The operation of Magnum-PSI is governed by two modes, and their respective states (fig. 3). Magnum-PSI starts in the Off state of the Service mode. Operation is performed in the Plasma mode. Switching modes is only possible in the Off state, in which Magnum-PSI is idle. When leading Magnum-PSI from the Maintenance to the Target Exposure state, e.g., the operator first has to bring it to the Off state, then switch from the Service to the Plasma mode, and then take it from the Off state, through the subsequent states, to the Target Exposure state. A state not only represents the state of the experiment, but also governs certain 'business' rules of the underlying software systems. Examples of such state related rules are that in the Target Exposure state it is not allowed to retract the target manipulator, that vacuum valves are not allowed to be closed in the Plasma Tuning state and that extra data is written to the database in the Target Exposure state. Rules for safety and protection on the other hand, are dealt with at a lower level. The Master determines the state and brings the other systems in that state. It also dispatches the command messages. Each system registers itself at the Master by telling its hostname and port number, so that other systems are able to find it by asking the Master. The Status service holds the operational data of Magnum-PSI, i.e. a cache of the actual values of the variables sent by the systems. Subsystems retrieve information of other systems from the Status service.

The database service manages the underlying data files, and receives data and meta data from the subsystems for persistent data storage. Each system, including future extensions (e.g. new diagnostics) is able to write to, and read from, the database. The timing service manages the time synchronization of Magnum-PSI. It holds the Magnum wall clock time, and manages the trigger inputs and outputs. The log service manages, and stores, log messages from the systems.

Subsystems are responsible for a specific task within the Magnum-PSI CODAC system. A subsystem with a hardware counterpart is an abstraction of that hardware. An example of a subsystem is the vacuum subsystem, that controls the pressure in the Magnum-PSI system. Each system owns a number of variables and has its own specific task associated with its variables. Only a system that owns the variable is allowed to assign an actual value to the data. E.g. the cooling subsystem owns the 'request pump on' variable, and after a request it switches on the pump when the system state and rules allow it. A variable

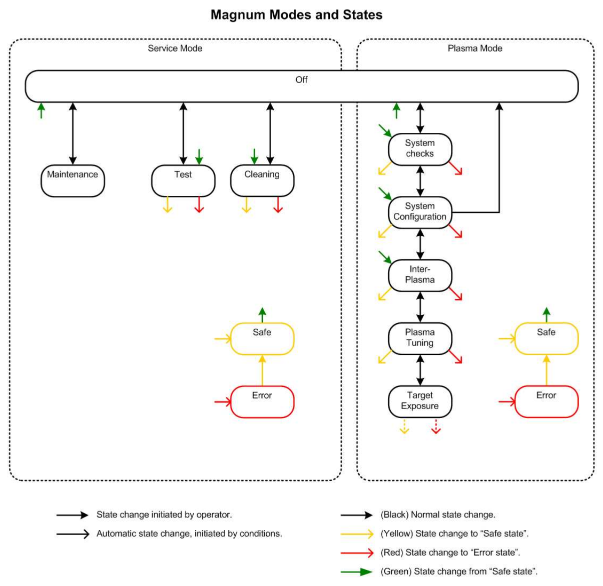
Magnum Modes and States
Service Mode
Plasma Mode
Off
Maintenance
Test
Cleaning
System checks
System Configuration
Inter-Plasma
Plasma Tuning
Target Exposure
Safe
Error
Safe
Error
State change initiated by operator.
Automatic state change, initiated by conditions.
(Black) Normal state change.
(Yellow) State change to "Safe state".
(Red) State change to "Error state".
(Green) State change from "Safe state".

Figure 3: The Magnum-PSI CODAC modes and states. The Service mode has 6 states, the Plasma mode has 8 states. Only the Off state is shared.

resides in only one system, contains data (e.g. value, time stamp), and has specific properties (e.g. a unique ID, the value's data type).

A special type of subsystem is the Graphical User Interface (GUI). It uses all general components that make up a subsystem. The information that a GUI displays, is retrieved from the Status service. A GUI is allowed to send command messages only to the corresponding subsystem and is typically used by an operator to control the corresponding subsystem, e.g, opening a valve by pushing a button in the GUI. GUIs are allowed to display any data from the Status service.

3. Subsystem design

Subsystems are the generic components of the architecture. This section shows how subsystems are built up themselves. The subsystem's internal components resemble the overall architecture components. They are called **managers** and **modules** and play the role of services and subsystems respectively (fig. 4). Their communication is also realized by sending, and receiving messages. Modules do not communicate directly with one another, but always via the managers. Managers are central to the subsystem. The two main managers are the Command manager, which acts as a gatekeeper, like the Master, and the Variable manager that holds and shares the variables' actual values, like the Status service.

A subsystem always contains a Master module, that communicates with the Master, a Status service module, that communicates with the Status service and a State-mode module, responsible for handling the (allowed) state of the subsystem. A module owns variables, has its own

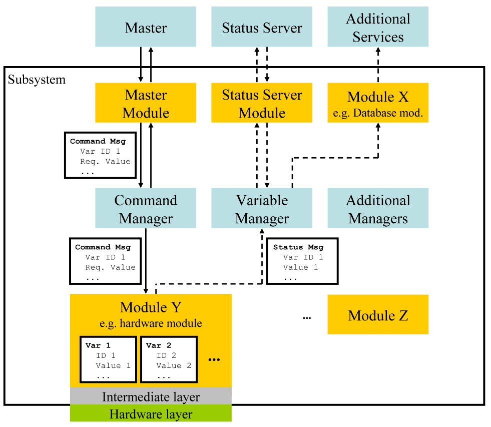

Figure 4: The design of the subsystem, with the basic elements, and the way they communicate.

specific task and is only allowed to update a variable that it owns. A subsystem can be extended by adding modules, e.g. a database module, a log module, or a hardware module.

The generic set-up of the subsystem also allows the services to be built up this way. Services like the log service and the timing service are in fact implemented this way.

The extensibility and flexibility in the design applies both to the software architecture and to the internal structure of subsystems themselves.

4. Implementation

This section will give a brief overview of the software architecture implementation. More details can be found in [8]. Key decisions at the start of the project were: using PLCs for real-time hardware control and safety (next to interlocking at hardware level) and HDF5 for data storage. This was followed by choosing Python as the main, platform independent, programming language (e.g. with PyTables for HDF5) and ZeroC Ice for client-server communication. Servers run on Linux, which in turn runs on a redundant, virtualized platform. GUIs (PyQt) run on Windows. When non-PLC based subsystems were added, and a more modular approach was needed, the architecture was further developed into its final form.

Servers and subsystems are written in Python. Subsystems connected to hardware via a PLC use an in-house developed client and additional code to glue the subsystem to the C coded PLC. This communication layer is an example of the implementation of the intermediate layer for the subsystem PLC hardware module. The command and status messages between systems are sent and received using Remote Procedure Calls (RPCs) with ZeroC Ice. A subsystem acts as a server to RPCs, receiving messages for its own tasks, or as a client when sending a message to the Master. For timing and synchronization of triggered events (not of separate clocks) a timing server is used, which holds the GPS based reference time. The connection between the timing server Python layer and its underlying (LabView based) FPGA is another example of an intermediate layer.

At the moment 19 subsystems (including 9 GUIs) and 5 servers (2 of which are based on subsystem design) are implemented. Each subsystem holds on average 5 modules. The systems carry together about 1500 variables related to hardware I/O and circa 20,000 variables that only live in software. Roughly 500 of these variables are stored in the database.

5. Conclusion

The basic CODAC system for the linear plasma generator Magnum-PSI is finished. The system manages the control of the device, arranges the data storage, and provides a complete interface for the Magnum-PSI users. The CODAC system works well. The modular design of the software architecture enables large flexibility, implying that additional services and subsystems can easily be added.

Acknowledgements

The authors would like to thank the Magnum-PSI team for their input. This work, supported by the European Communities under the contract of Association between EURATOM/FOM, was carried out within the framework of the European Fusion Programme with financial support from NWO (Netherlands Organisation for Scientific Research, funded by the Netherlands government). The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] A. Kleyn et al., Plasma-surface interaction in ITER. Vacuum, 80 10 (2006), pp. 1098.
[2] J. Rapp et al., Construction of the plasma-wall experiment Magnum-PSI. Fusion Eng. Des., 85 (2010), pp. 1455.
[3] O.G. Kruijt et al., Thermal effects and component cooling in Magnum-PSI. Fusion Eng. Des., 86 (2011), pp. 1724.
[4] J. Scholten et al., Operational status of the Magnum-PSI linear plasma device. This conference (2012).
[5] G. De Temmerman et al., Overview of the first results from the Magnum-PSI high flux device. This conference (2012).
[6] M.A. van den Berg et al., Study of the influence of target tilting and castellation on power load distribution. This conference (2012).
[7] P.C. van Haren and F. Wijnoltz, TRAMP, the next generation data acquisition for RTP. IEEE Trans. Nucl. Sci., 39 (1992), pp. 95.
[8] G.W. van der Linden et al., Design of the Magnum-PSI Safety, Control and Data Acquisition System. Fusion Eng. Des., 83 (2008), pp 273.
[9] G. De Tommasi et al., A flexible and reusable software for real-time control applications at JET. Fusion Eng. Des., 74 (2005), pp 515.
[10] W. Treutterer et al., The new ASDEX upgrade real-time control and data acquisition system. Fusion Eng. Des., 66-68 (2003), pp. 755.
[11] G. Raupp et al., Control process structure of ASDEX Upgrade's new Control and Data Acquisition System. Fusion Eng. Des., 74 (2005), pp. 697.
[12] M. Korten et al., JDAQ, the new TEXTOR data acquisition program. Fusion Eng. Des., 81 (2006), pp. 1723.
[13] J. Schacht et al., Overview and status of the control system of WENDELSTEIN 7-X. Fusion Eng. Des., 82 (2007), pp. 988.
[14] A. Neto et al., FireSignal-Data acquisition and control system software. Fusion Eng. Des., 82 (2007), pp. 1359.
[15] A.S. Duarte et al., FireSignal application Node for subsystem control. Fusion Eng. Des., 85 (2010), pp. 496.
[16] The TANGO website, http://www.tango-controls.org/, 01-Aug-2012.
[17] EPICS, Experimental Physics and Industrial Control System, http://www.aps.anl.gov/epics/, 01-Aug-2012.
[18] CODAC Core System documents in the CODAC Section of the ITER website, http://www.iter.org/org/team/chd/cid/codac, 01-Aug-2012.