ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP024

TECHNICAL DESIGN CONCEPT AND FIRST STEPS IN THE DEVELOPMENT OF THE NEW ACCELERATOR CONTROL SYSTEM FOR PETRAIV

R. Bacher, J. Behrens, T. Delfs, T. Tempel, J. Wilgen, T. Wilksen
Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany

Abstract

At DESY, extensive technical planning and prototyping work is currently underway for the upgrade of the PETRAIII synchrotron light source to PETRAIV, a fourth-generation low-emittance machine. As part of this planned project, the accelerator's control system will also be modernized. This paper reports on the main decisions taken in this context and gives an overview of the scope of the development and implementation work.

INTRODUCTION

With PETRAIII, DESY operates one of the best storage ring X-ray radiation sources in the world. PETRAIII is a 2300-metre-long storage ring feeding 24 user beamlines. It is operated either in brightness mode (480 equally distributed bunches, 120 mA stored beam) or in timing mode (40 equally distributed bunches, 100 mA stored beam), the latter a unique feature of PETRAIII. Research groups from all over the world use the particularly brilliant, intense X-ray light for a variety of experiments - from medical to materials research.

DESY plans to expand PETRAIII into an ultimate, high-resolution 3D X-ray microscope for chemical and physical processes. PETRAIV will extend the X-ray view to all length scales, from the atom size to millimetres. Researchers can thus analyse processes inside a catalyst, a battery or a microchip under realistic operating conditions and specifically tailor materials with nanostructures. PETRAIV offers outstanding possibilities and optimal experimental conditions for industry. Special emphasis is also placed on the sustainable operation of the facility.

PETRAIV will replace the PETRAIII facility and will be housed by the existing PETRAIII buildings. An additional experimental hall will provide space for another 18 user beamlines. A new synchrotron (DESYIV) will serve as booster between the existing electron source LINACII and PETRAIV. In addition, research has begun on the development and construction of a 6 GeV laser plasma injector.

In 2020, a preparatory phase for the future project PETRAIV was initiated with the aim of finishing a Technical Design Report by mid-2024. In the meantime, extensive planning work has been carried out with regard to buildings and facility infrastructure, specification of technical equipment and prototyping or beam-physical optimizations and simulations. In addition, the estimated project costs were determined and first steps were taken for the political decision-making process. The dismantling of PETRAIII and construction of PETRAIV will result in a two-year shutdown of operations at the PETRA complex.

GENERAL CONCEPT

To consolidate and simplify the whole accelerator control system landscape at DESY and to take advantage of synergies between the accelerator facilities operated by DESY the control system of PETRAIV will closely follow the control system concept implemented at the European XFEL, which is a pulsed linear accelerator and free-electron laser. Therefore, the control system concept of PETRAIV will be adapted where necessary to the special needs of a storage ring X-ray radiation source. While many existing generic and basic software solutions from the European XFEL can be re-used for PETRAIV purposes, some have to be adapted or newly created to accommodate the different machine type, i.e. synchrotron vs. pulsed linear accelerator. These include, for example, the measurement of the beam positions or the fast orbit feedback system, but also the control of the magnetic power supplies. In addition, central control system services such as data acquisition and archiving, alarming or configuration management are to be extended and prepared for future requirements.

CONTROL SYSTEM FRAMEWORK

The Distributed Object-Oriented Control System (DOOCS) [1] will form the basis of the accelerator control system. DOOCS is the leading accelerator control system at DESY. The initial development of DOOCS dates back to 1993. Since that time, it has steadily developed into a powerful, reliable and versatile control system. Recently, a road map was established to meet the increasing user demands over the next decade and to continue to keep pace with the rapid developments in IT and the controls community.

Architecture

DOOCS follows an object-oriented design paradigm. Devices and data are objects. The basic entity is a device server representing some control system hardware or logic. The DOOCS naming scheme is hierarchical. The layout implements a three-tier approach (Fig 1). The front-end tier (resource layer) contains the device interface applications (device servers) that are connected to the accelerator devices through various field buses and hardware interfaces. The service tier (middle layer) provides common control system services (e.g. data archives) and cross-component or cross-system control functions (e.g. beam orbit). At the client tier (user interface layer) graphical user applications and tools for monitoring and operating the accelerator are located. The three tiers are interconnected through the network-based control and data bus of the control system.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP024

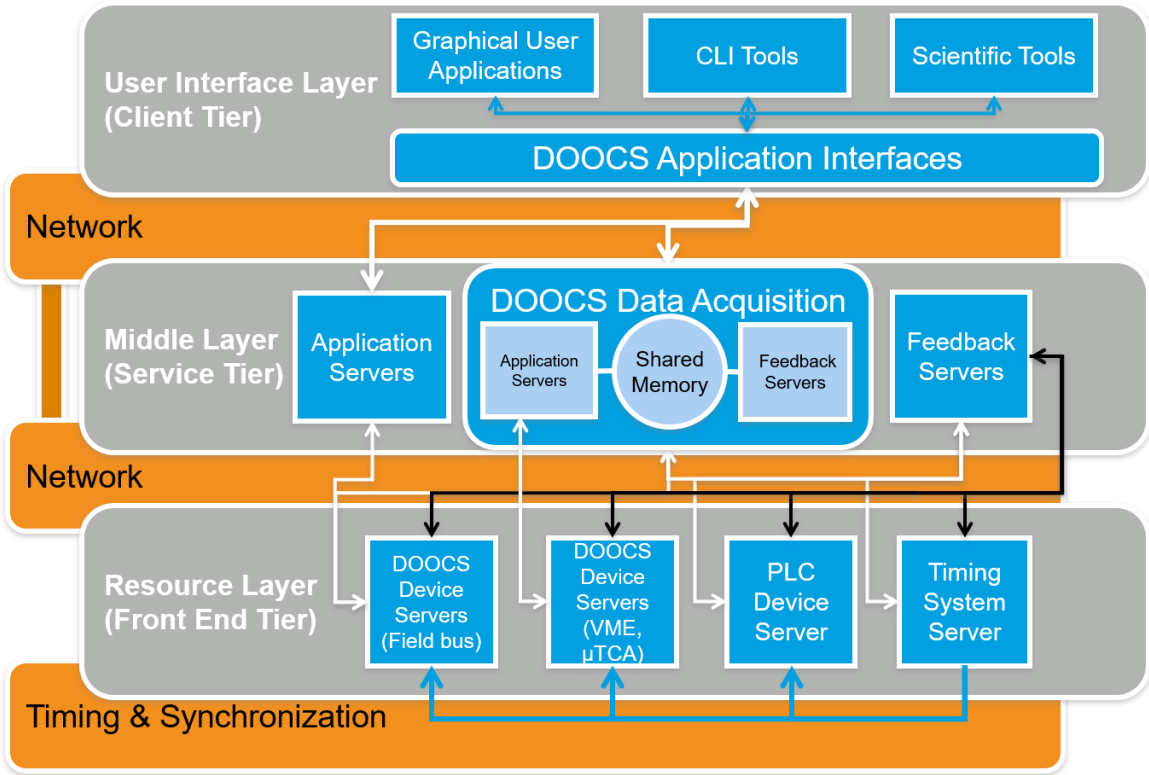

Figure 1: DOOCS architecture.

DOOCS provides a special middle layer application (aka DOOCS DAQ) that collects and synchronises tagged (e.g. time stamp, pulse number, revolution counter number) data supplied by different device servers. This service follows the design concept of a data acquisition system for high energy physics experiments and will be able to process data at a continuous rate of O(kHz). These generated data sets are made available to other middle-layer processes connected to the DOOCS DAQ for further synchronised data processing (e.g. software feedback loops).

Communication via the data and control bus of DOOCS is currently based on the standardized, industrial RPC protocol with XDR data representation. An integration of the ZeroMQ [2] protocol as an alternative and future default is in progress. DOOCS server applications implement a C++ application programming interface (API). An equivalent Python server API is under development. API libraries for creating client applications in C++, Java, Python or MATLAB [3] are available either as a separate implementation or as C-Bindings.

Application Development and Management

The DOOCS framework has extensive capabilities and tools for control system browsing, starting and configuring applications, process-based application monitoring, managing MicroTCA [4] crates and modules via Intelligent Platform Management Interface (IPMI), or package and driver management.

The software development and deployment chain used for DOOCS complies with open source standards. The main development platform is Linux, so the build tool chain is the standard GNU [5] chain. Code development is done using a GitLab [6] instance. Utilizing software development standards, like version management, code reviews, unit and system tests, continuous integration, package provision and management, together with nightly builds allows for modifications, bug fixing and enhancements to be rapidly deployed. Full releases of the DOOCS framework are periodically done, typically every three months.

Setup and installation of DOOCS-based control system nodes is supported by a fully automated installation service. In order to exploit synergies with the central IT department, the Puppet-based service [7] used throughout DESY is to be evaluated for PETRAIV.

Interoperability

The DOOCS control system is only responsible for the accelerator. The DOOCS client API is providing native access to EPICS control system instances. EPICS is used for facility and utility control, i.e. electrical power and water distribution, ventilation and air conditioning. TANGO is the standard control system framework for operating the beam line components and the experimental equipment. Access to TANGO device servers can be implemented either as dedicated gateways or as native implementation into the DOOCS client API. The latter has been done already for the Java-based DOOCS client API.

Development and Simulation Environment

A virtual PETRAIV accelerator infrastructure will be used to test new concepts, extensions or simply modified and improved applications before they are put into the field, which can lead to considerable time savings in commissioning and machine studies. In addition, the PETRAIV

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP024

virtual accelerator will include both static and dynamic modelling of specific devices as well as a digital model that predicts the behaviour of the beam in terms of, for example, orbit, tuning, coupling, lifetime, or emittance as a function of the virtual actuator settings.

DEVICE INTERFACES

In general, the device interfaces for triggered, high-performance applications (e.g. beam diagnostics, injection / ejection system, feedback systems, timing / synchronization system, machine protection system, RF control) will be compliant with the high-end MicroTCA.4 technology. MicroTCA.4 is the accepted long-term standard for the DESY accelerators and is enjoying growing popularity within the accelerator community and the related industry. The operating system for server hosts running within the MicroTCA platform is Linux. The base configuration of a MicroTCA system includes a power-supply, a Management-Controller Hub (MCH), a CPU as an Advanced Mezzanine Card (AMC), a Timing System AMC, and optionally a Timing System Rear Transition Module (RTM) if required. General-purpose analogue-to-digital conversion and multi-purpose digital input-output processing AMC boards are available. Dependent on the location in the accelerator, the MicroTCA systems will be equipped with additional application-specific AMC, FMC (FPGA Mezzanine Card) and RTM modules for specific read-out, measurement and control tasks, e.g. interface boards to the beam position monitor front-end electronics or feedback controllers. All of these modules can be managed remotely via IPMI interfaces and the MCH. Access and read-out of the front-end electronics from the host CPU is done by means of PCIe and Direct-Memory Transfer (DMA) using either commercially available or in-house developed Linux drivers.

The device interfaces for slow-control applications (e.g. magnet power supplies, vacuum system devices, temperature sensors, drives of movable girders) will be compliant with industrial process control standards, providing a well-established and widely-used industrial API (e.g. OPC UA [8], Profibus [9], EtherCAT [10], Modbus [11], CANopen [12], RS232/485). The preferred one will be the OPC UA interface technology. All power converters for magnets as well as power supplies for getter pumps of the vacuum system will implement an OPC UA server. DOOCS provides a generic bridge server, which seamlessly integrates OPC UA devices into the accelerator control system. PLC-type hardware systems based on the Beckhoff [13] controller technology using EtherCAT or OPC UA will provide control of motors driving insertion devices or the movable girders supporting all accelerator components. In addition, other PLC systems have to be interfaced using generic bridge servers to the accelerator control system.

GRAPHICAL USER INTERFACES

The Java DOOCS Data Display (JDDD) [14] is chosen as the graphical user interface (GUI) for the standard beam operation as well as operating technical accelerator devices and systems. JDDD follows a thin-client approach. Even complex accelerator control system GUI can be easily created through a versatile integrated development environment (JDDD Editor) without the knowledge of any programming language using a functional and rich set of widgets. In addition to standard control widgets such as buttons or dynamic value labels, JDDD offers a large number of plot options.

While JDDD is the GUI tool for standard operation, Python-based rich-client applications using the DOOCS Python API are becoming increasingly the solution for rapid prototyping and visualization of scientific procedures and data. As with PETRAIII, MATLAB-based rich client applications using the MATLAB middle layer library [15] will be the preferred tools for accelerator development or optimization studies, e.g. for performing a beam-based alignment procedure.

Web-based graphical user applications are becoming more and more accepted and promising alternatives to traditional GUI applications. JDDD already offers a secure HTML5-based web application for remote access via the browser. Furthermore, novel responsive, interactive, cross-platform progressive web apps based on the web development platform React [16] are investigated with respect to performance and user acceptance and first dashboard-like applications (Fig. 2), such as operational overviews or an app for viewing archived operational data, are being implemented.

ACQUISITION AND ARCHIVING OF OPERATIONAL DATA

Operational data will be acquired and archived either continuously as time series or as snapshots triggered by certain events.

In most cases, time series data from about 10,000 devices must be collected at a rate of typically 10 Hz to a maximum of 100 Hz. Some data must be stored for an indefinite period of time, others only for limited periods of time. Various time series databases have been explored and benchmarked. The open-source database Timescale [17], an extension of PostgreSQL [18], has proven to be particularly suitable. The reasons for this include the fact that it is optimized for time series data and is built on mature and robust database technology, can handle a variety of data types including arrays, has very good scaling behaviour due to special design concepts, and can be operated as a distributed cluster. It also offers a large number of specific processing functions and has a high level of interoperability with other software tools.

A high-availability cluster is currently being set up for the European XFEL to archive operational data in order to gain experience with the overall performance and scalability of this database system. A data pipeline between DOOCS device servers and the database will be designed and implemented. Other conceptual considerations, such as data reduction strategies through filtering and compression or supplementing the central database with local buffers on the device servers, are still under discussion.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP024

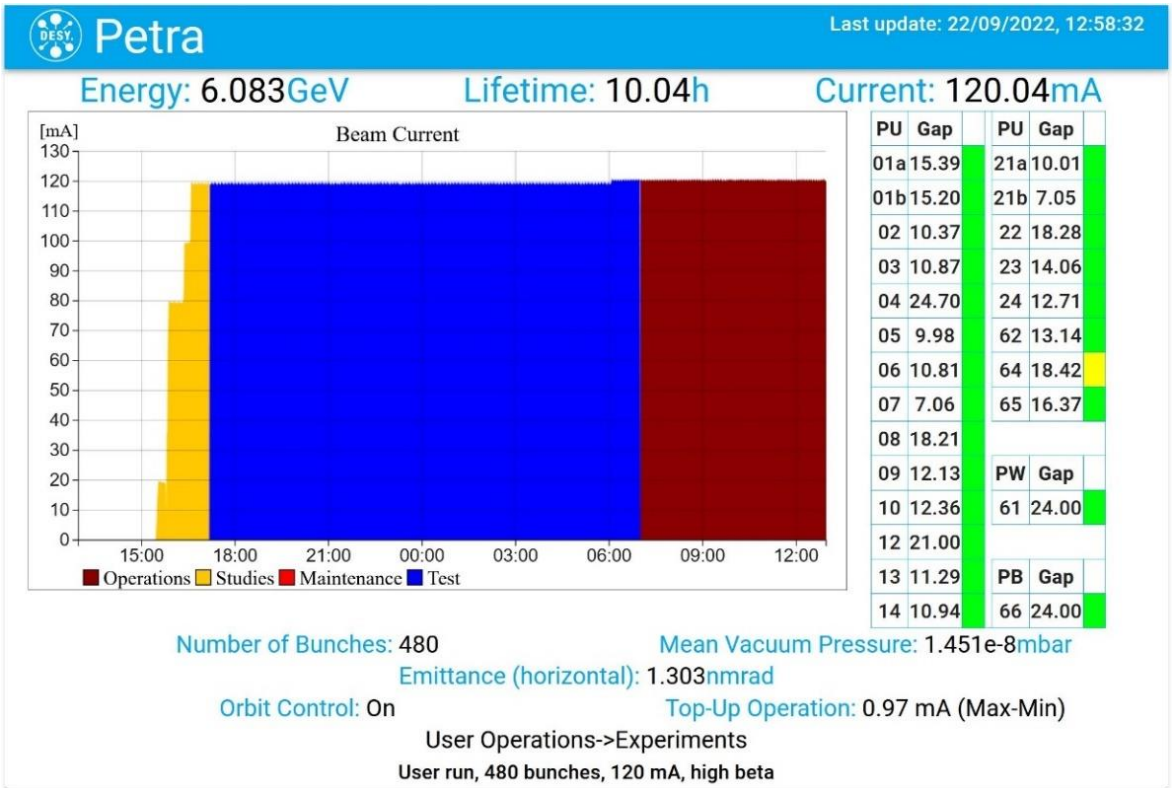

Figure 2: Progressive Web Application.

Only in very rare cases, data from selected devices must be acquired at a rate of O(10kHz) for a limited period of time. These data are transmitted by the control system via UDP multicast and will be processed and stored by the DOOCS DAQ system whose performance is currently being improved.

Snapshot data such as post-mortem beam orbit data recorded just before and after a beam loss is typically stored in a multi-purpose relational database or in a file system.

In both cases, an efficient API for data retrieval and versatile visualization and analysis tools must be provided. Particular emphasis will be placed on the ability to support data science applications, such as learning feedback or learning tuning routines that operate online, or applications to facilitate system analysis, fault prediction, or predictive maintenance.

MANAGEMENT OF CONFIGURATION DATA

The entire collection of configuration data describes the current state of the PETRAIV storage ring and its components. Configuration data includes hardware-related data such as calibration parameters of magnets or beam position monitors or software-related data such as control system addresses or configuration parameters of server installations. State-dependent operational setpoint data (e.g. magnet current values) that is managed by a save-and-restore tool is not considered configuration data.

Some configuration data, such as the position of an optical component in the accelerator lattice, is quasi-static and can only be changed through formalized change request workflows. On the other hand, there is also configuration data such as the collection of steerer coils used for orbit correction, which serves as reference data for parameters that can be changed via the control system during regular operation.

All configuration data will be managed by a central configuration data management system. To allow easy rollback to a previous state, the configuration management system must maintain a history of the changes made. Each configuration item therefore has its unique, immutable identifier and an associated temporal validity range. The configuration data will be stored in tables contained in one or more relational databases. The design of a suitable database structure has been started and first data like the lattice parameters of PETRAIV have already been added.

Different versions of software applications will not be managed by the central configuration database management system. These will be stored and tracked by a repository or a DevOps tool like GitLab, which is, however, connected to the configuration management system as necessary.

ACKNOWLEDGEMENTS

The authors thank the PETRAIV project team at DESY for constructive discussions and the Helmholtz Association of German Research Centres and the Federal Ministry for Education and Research for supporting the preparatory phase for the future project PETRAIV.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUPDP024

REFERENCES

[1] DOOCS, https://doocs.desy.de

[2] ZeroMQ, https://zermq.org/

[3] MATLAB, https://www.mathworks.com/

[4] MicroTCA, https://www.picmg.org/

[5] GNU, https://www.gnu.org/

[6] GitLab, https://about.gitlab.com/

[7] Puppet, https://www.puppet.com/

[8] OPC UA, https://opcfoundation.org/

[9] Profibus, https://www.profibus.com/

[10] EtherCAT, https://www.beckhoff.com/

[11] Modbus, https://modbus.org/

[12] CANopen, https://www.can-cia.org/canopen//

[13] Beckhoff, https://www.beckhoff.com/

[14] JDDD, https://jddd.desy.de/

[15] MATLAB Middle Layer, https://github.com/atcol-lab/MML

[16] React, https://react.dev/

[17] Timescale, https://www.timescale.com/

[18] PostgreSQL, https://www.postgresql.org/