Proceedings of PCaPAC 2005, Hayama, Japan

# STATUS OF THE TTF VUV-FEL CONTROL SYSTEM

K. Rehlich, DESY, Hamburg, Germany

*Abstract*

The second phase of the TESLA Test Facility (TTF) at DESY started recently the operation. TTF was built by an international collaboration as a test bench for super conducting cavity technology. In addition this linear accelerator will be operated as a VUV-FEL user facility. The FEL needs a sophisticated diagnostics with a single bunch and shot to shot resolution and a high demanding data transfer rate. On the other hand the control system has to provide the flexibility for the test facility.

As a further requirement the control system has to integrate the different systems of the collaboration partners. This paper gives an overview and describes the used technologies of the Distributed Object Oriented Control System (DOOCS) as the integrating part. To handle the high data rates a novel combination of an accelerator control system with a Data Acquisition system (DAQ) has been implemented.

## INTRODUCTION

The TESLA Test Facility (TTF) was initially designed and built as a test facility for superconducting cavity technology including a 100 meter long linac [1]. During the last years the accelerator was extended in length to provide a 1GeV electron beam. Six undulator sections are installed to operate the machine as a VUV-FEL for user experiments. In January 2005 the first lasing at a wave length of 30nm was demonstrated.

This leads to the requirements for the control system. To test cavities, diagnostics, and to perform numerous measurements the system has to be flexible. As a machine for user experiments a very reliable and stable operation is required. Since this accelerator is a short prototype for a future XFEL and a linear collider also, the design of the control system has to take these requirements into account as well.

Furthermore a FEL requires a high peak current beam with low emittance and stable operation. This demands a lot of complex diagnostics. All the values have to be recorded and stored for later analysis. In case of TTF, 18000 values are permanently stored in distributed archivers. In addition, a high performance data acquisition system (DAQ) is under development. First results of the DAQ are available. A further requirement for the control system is the automation of the machine operation. Especially the RF system has a quite high complexity and the automated setup and fault recovery is helpful for the linac operation.

TTF was designed and build by the TESLA collaboration. A lot of contributions from other labs had to be integrated into the control system. For an operator and an application programmer the differences of the different components of the partners are hidden by the control system.

All these requirements were the motivation to develop during the last years the Distributed, Object Oriented Control System DOOCS. The architecture and the main design ideas are described in the following chapters.

## ARCHITECTURE

The DOOCS architecture is based on three main layers. Application programs with a user interface are in the top layer. All these programs connect to the other two layers via a common API. On the lower layer all services with device connections are located. The layer in between is called the middle layer and contains higher level services. Effort was put into a clean division of functions and tasks in the three layers to crate a manageable system with clean and simple interfaces.

Most of the common functions in the layers are based on libraries. This leads to a modular design of the whole control system. The error handling is an example for this approach. A library function is provided to set an error condition of a device that prints a line into the log file. Extending this function to send in additional message to a central alarm server is done by adding an additional feature to the library. The device servers have just to be recompiled to inherit this new function.

A further design goal of DOOCS is to operate a server ‘self-contained’. Device servers maintain a local configuration file and archive. This allows to restart a server with the recent settings after a crash for instance. Since the servers keep their configuration parameters local, all parameters are online configurable. Also this is provided by the library.

DOOCS is fully designed with object oriented technologies. Device servers are composed by device objects. Data objects are used to provide the network access of the device server variables.

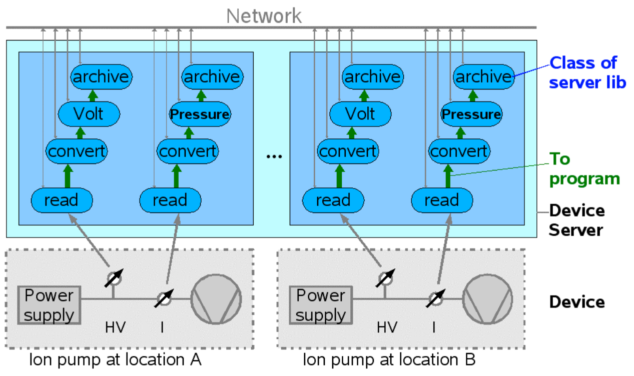

Figure 1: Object orientation of device servers.

The common application programming interface is based on objects too. It consists of a communication class, an address class and a data class with the respective methods to exchange the information on the network.

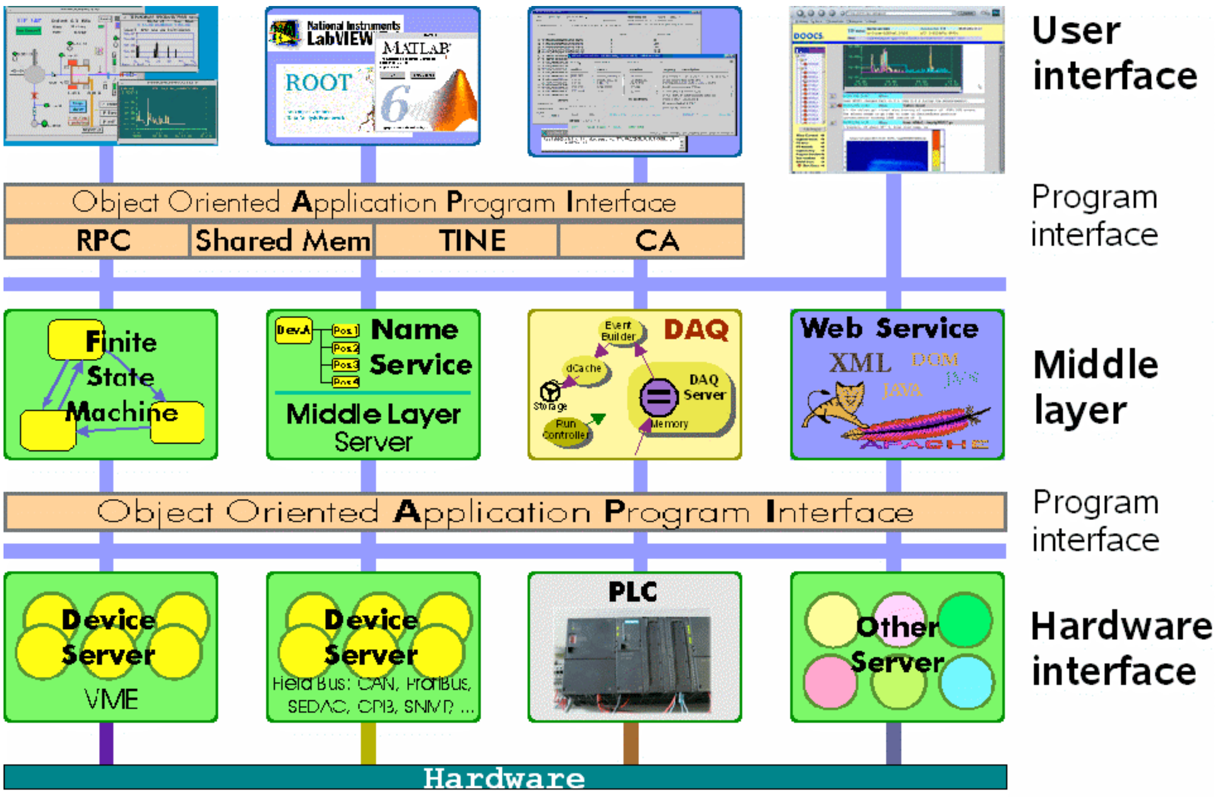

Figure 2: DOOCS architecture.

Internally the API has implemented four communication protocols: RPC, Shared memory, TINE and EPICS. The selection of the protocol is done within the API by a call to a name server. This name server provides all server names and protocols to use. It also delivers the information for online name queries.

All applications and libraries of the DOOCS are available on Solaris and LINUX operating systems. DOOCS is partly available on Windows systems as well [2]. At TTF about 40 VME crates with SPARC computers and some LINUX PCs are used as front-ends. File and middle layer servers are SPARC and PC (LINUX) based.

The whole control system was designed from scratch ten years ago. The experience with TTF demonstrates that this object oriented design approach is an advantageous technology for future FELs and linear colliders.

## DEVICE SERVER LAYER

A DOOCS device server is a (UNIX) process written in C++ that controls one or more devices of a certain type. All device classes are derived from the device base class. The base class defines a number of properties and functions to provide a common standard part in all servers, e.g. the error handling. The individual properties of a device are accessible on the network. Properties inherit from a data class. This base data class deals with the communication to the local configuration file and the network. The server library contains a large number of data class types for e.g. floats, data conversion, archives, filters, and field bus access. Up to now the following field busses are implemented in DOOCS: CAN, ProfiBus, RS232, GPIB, TCP, SNMP, FireWire [3], and FNAL 'Classic Protocol'.

Since DOOCS is a distributed system these device servers are independent of the network. Each server has its own local configuration file and archive. In case of a network failure a server can restart and gets from the configuration file the most recent parameters like set-points. Also the archiving uses a local file [4]. In case of a crashed or hanging server a watchdog process automatically starts/restarts the device servers. This watchdog itself is a device server and therefore visible on the network.

All the standard parts of servers are defined in a library. This library is multi-threaded to provide 'soft real time' with no blocking. Threads and the mutexes are hidden from the normal server programmer.

Since most of the code of a device server is in the server library, it is possible to create server C++ code by a script too. An example of such a script based generation of servers is implemented for PLCs [5]. In a text file all data points of the PLC are defined together with the foreseen property names and data types of the server. The script reads this file and creates the C++ code and a configuration file. After a 'make' command the server is started and reads the configuration file to create the required number of device instances.

Almost 200 different device servers have been written so far. The applications cover simple digital and analog IO, communication to complex devices like DSPs [6], gateways to other subsystems, and adaptation of instruments.

## MIDDLE LAYER

The middle layer in the control system provides a number of services. A name server (ENS) allows online name queries and the resolving of host names and protocols. The first call of a device request in the API generates a request to the name server. This server returns all the required information for the API to direct further requests directly to the corresponding host with the right protocol (RPC, TINE, ChannelAccess or shared memory). Several copies of the name server are running on several hosts for redundancy.

A further middle layer service is a Finite State Machine (FSM). The FSM is designed by the graphical editor and code generator (ddd). During runtime the editor shows the actual active states. State machines are used for the automation of the operation. An example is a FSM that controls the complete start-up and restart procedures of the RF including the determination of the correct loop phase and the loading of DSP programs etc.

The Alarm server is a further service in the middle layer. This server receives from all others servers in the system XML based alarm and status messages and stores the status in a device tree and in an archive. A Java client application displays the status and the history of the alarm. This service is in a test phase and will be released soon.

Furthermore, Web services like an electronic logbook [7] are implemented as middle layer services. And the central part of the DAQ system is in this layer as well.

## DATA ACQUISITION SYSTEM

The Data AcQuisition system (DAQ) [8] is a novel integration of the technologies from High Energy Physics (HEP) experiments and accelerator controls. The main goal of this project is to

- improve the reliability and to optimize the linac
- correlate data of the user experiments with the machine
- provide all current machine readings in one place to be used by feedback processes and measurements
- store the data for off-line analysis

The TTF diagnostics is mainly based on 10 MHz, 14bit ADCs. The ADCs are sampling every bunch of all machine shots. More than 700 ADC channels are used to readout the BPMs, toroids, and RF electronics. The data is collected by a DMA transfer from VME into a device server. After processing the data is send to a central shared memory with a data rate between 10 and 100 MB/sec (Figure 3). Feedback and other processes [9,10] can be attached to the shared memory and extract selected values. The 'collector' of the shared memory receives the data and is responsible for the synchronization of the data that is coming from about 30 VME crates.

A 'distributor' sends the data from the shared memory to an 'event builder'. This process packs the data in ROOT [11] files for later analysis by ROOT tools and to store them on a tape in the computer center via the dCache system. ROOT is a tool kit developed in CERN for the HEP community. A 'run controller' configures all the data flow and the linac settings. The configuration information is stored on an ORACLE data base [12].

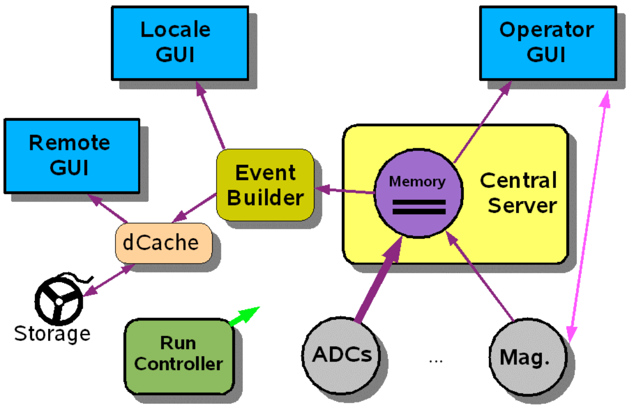

Figure 3: Data Acquisition System.

The DAQ system was developed by a collaboration of Cornell, Mitchigan University, and DESY (Hamburg and Zeuthen) as a 'GAN' project. GAN is a concept to build and operate an accelerator by an international collaboration with the help of remote tools.

## USER APPLICATIONS

As user applications several tools and program packets are used to operate the TTF VUV-FEL. DOOCS provides a graphical editor and synoptic display program (ddd). With the editor graphical displays can easily be designed without any programming (Figure 4). The elements of the drawing can be animated. In the example the magnets are displayed in green if the power is on, red if an error occurred or grey if they are off. Such an animated magnet has to be drawn once and is then reusable. The numbers shown in the figure are BPM readings. By clicking with the mouse on such a value a new window with a plot of the archived data appears. This functionality is simply activated by a checkbox during design time.

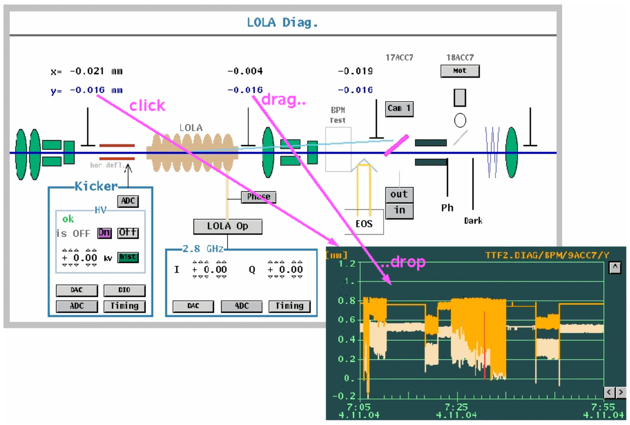

Figure 4: The DOOCS data display (ddd), drag and drop operation.

A further curve might be inserted into the plot by a drag and drop operation from another value. With the context sensitive menu it is possible to modify the plot to a correlation display that displays all the data points of the two channels in the selected time range. Many other plot types are available in ddd. With the help of buttons one can start new windows, overload windows, change data in devices, start UNIX commands or displays help information in a Web browser.

Recently Jpeg and Tiff image displays were added to the feature list of ddd. Images can be static or dynamic from a file or directly be read from a device server via the DOOCS API. Image output is also possible. The content of a window can be saved as Jpeg into a file or directly into the electronic logbook. Further output can be generated as Postscript from a print window. This window can be edited before sending it to a printer.

State machine designs are done with the ddd program as well. The states and transitions defined in a drawing are converted into executable C++ server code by use of the 'make' command. When ddd is switched from the edit to the run mode, the actual active states of the generated program are displayed together with other values of the control system.

Further user application packets are:

- MATLAB, is used for
  - Simulations, e.g. the RF system
  - writing ad-hoc applications
  - complex calculations, e.g. emittance
  - high level procedures for state machines
- ROOT
  - DAQ data analysis
  - Special applications, e.g. orbit display
- LabVIEW
  - Operates the OTR monitor system
  - Used for cavity conditioning and test stands
- Save&Restore Utility to:
  - save and reload linac settings and device configurations
- Java tools for:
  - Alarm display
  - Spread sheet application
  - DAQ run control
- Electronic LogBook
  - For machine operation
  - To store documents

All data accesses of these programs are implemented by the standard DOOCS API. For the Java tools an own version of the API was ported to this language.

## CONCLUSIONS

The TTF VUV-FEL is in operation since end of last year. First lasing was observed beginning of 2005. The DOOCS control system integrates all devices of the accelerator, including the contributions from the collaboration partners, with a standard interface. Except for a few minor problems with the new installed hardware, the systems runs very stable. The object orientated and library based design of the control system was proven to be well manageable an extensible.

The Data Acquisition system is in its test phase and data of the linac diagnostics passes the whole system up to ROOT plots. First experience with this novel design shows the benefits of the fast central collection of all machine data for middle layer services and feedbacks.

The first useful integration of Web technologies for DOOCS was demonstrated with the electronic logbook. As a next step the alarm system will be setup with Web services. More developments in this direction will follow.

With more experience in the operation of the VUV-FEL further automation will be developed in the near future. It is planed to further develop the control system of TTF in the direction of the envisioned XFEL.

## ACKNOLEDGEMENTS

The described work was done by the TTF controls group. I would like to thank all members of the group for their engaged and very successful work. The data acquisition system was designed and implemented by the DAQ collaboration. My gratitude goes to the team members for the very fruitful discussions and the skilful implementation.

## REFERENCES

[1] Rossbach et al, Generation of GW radiation pulses from a VUV free-electron laser operating in the femtosecond regime, Phys. Rev. Lett. 88, 104802 (2002)
[2] V. Kocharyan, WDOOCS: Porting DOOCS to Windows PC, PCaPAC 2005 (Hayama, Japan)
[3] V. Kocharyan, WDOOCS: FireWire Cameras Support for DOOCS, PCaPAC 2005 (Hayama, Japan)
[4] H. Keller, Experience with the Data Archiver in DOOCS, PCaPAC 2005 (Hayama, Japan)
[5] G. Grygiel, TCP/IP Based PLC Connection to DOOCS, PCaPAC 2005 (Hayama, Japan)
[6] G. Petrosyan et al, Hardware and Software Design for the DSP Based LLRF Control, PCaPAC 2005 (Hayama, Japan)
[7] R. Kammering et al, The Electronic Logbook at the TTF VUV-FEL, PCaPAC 2005 (Hayama, Japan)
[8] V. Rybnikov et al., Data Acquisition System for the TTF VUV-FEL Linac, PCaPAC 2005 (Hayama, Japan)
[9] M. Kollewe, Orbit Data Processing using the Data Acquisition System at the TTF VUV-FEL, PCaPAC 2005 (Hayama, Japan)
[10] E. Sombrowski et al. Wire Scanner Control and Display Software, PCaPAC 2005 (Hayama, Japan)
[11] http://root.cern.ch
[12] G. Dimitrov, Application of Oracle Database for TTF DAQ System, PCaPAC 2005 (Hayama, Japan)