EQUIPMENT NAME SERVER AND TESLA TEST FACILITY CONTROL SYSTEM

O. Hensler, K. Rehlich, P. Shevtsov*
Deutsches Elektronen-Synchrotron,
DESY, Hamburg, Germany

1. Introduction.

The TESLA Test Facility (TTF) was designed for the creation of a prototype of the linear accelerator of a new generation [1]. It is equipped with all components for processing and testing superconducting cavities which should be the basic elements of such an accelerator. The main control problems of the TTF are currently solved with the use of the Distributed Object Oriented Control System (DOOCS) developed at DESY. This system is developed in such a general way that any new functionality can be added to it relatively easily [2].

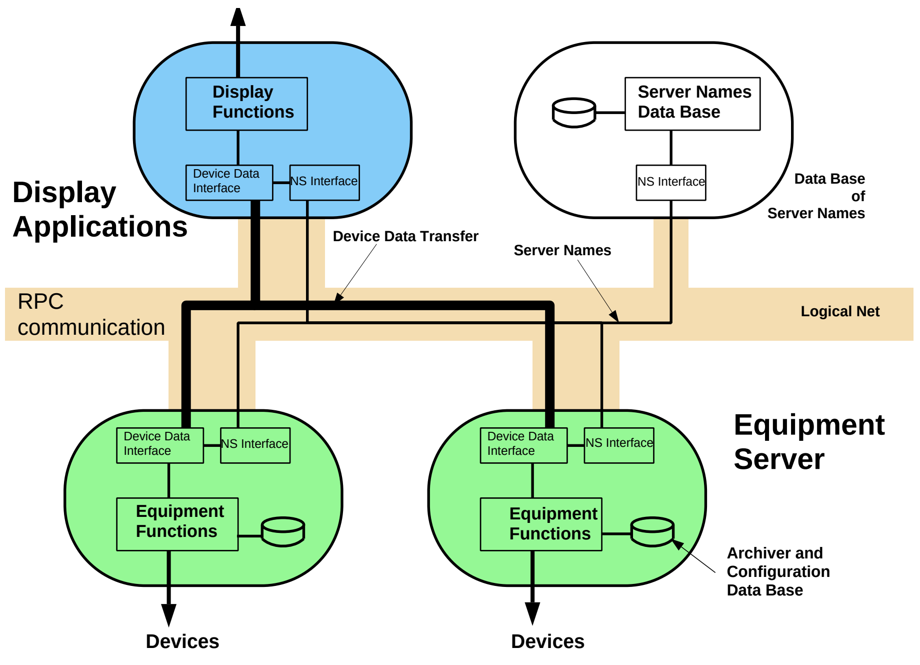

FIGURE 1. DOOCS layout.

* - *on leave from Institute for High Energy Physics, Protvino, Russia*

2. DOOCS and its main principles.

DOOCS programming is done in C++, under operating systems Solaris 2.4 and SunOS. The communication is based on the remote procedure call (ONC/RPC) with the eXternal Data Representation (XDR) network format. DOOCS is a device oriented system because it is based on the next ideas:

- every hardware device is an object;

- all properties of a device are handled by the corresponding device server;

- every such a server can create as many as necessary device instances (locations) of its type.

In DOOCS terms a device server is an independent program that provides the requested data to the network and a client is an independent program which receives these data. DOOCS is a distributed system because the device definition is done on the server side only and is transparent to the client (see Fig.1). Many different servers can run on the same CPU and from the other side a particular server can run on different computers. Even the same type of a device server with different locations is able to run on many CPUs. In a distributed system with mechanisms of calls of remote procedures if a client program wants, for example, to get data from equipment it has to know usually such important parameters as:

- the name of the device server performing control functions on this equipment,
- the network address (or name) of the computer which hosts it,
- the communication protocol,
- the types of data supported by this server.

To obtain and properly interpret this information is not a simple task for a client.

In the frames of DOOCS this problem is solved the other way. The unique DOOCS (ASCII) name in the form "facility/device/ location/property" is assigned to each element of the control system. It is a hierarchical structure similar to the well-known computer file systems. Note that we already mentioned above the last three parts of such a DOOCS name. The parameter "facility" acts as an additional degree of freedom and can be used, for example, to test the same server on different machines or for the effective work of the control system in a multiprotocol environment.

The complete set of valid DOOCS names as well as all the information about the control system elements corresponding to such names are stored by a special DOOCS RPC server - the Equipment Name Server (ENS). A user is provided by the Application Programming Interface library (API). This library gives one a set of standard calls to access the control system. At the same time the software is designed so that the Equipment Name Server communicates exclusively with API. Hidden from a user this communication allows one to effectively resolve DOOCS names contained in client requests and to direct these requests to the proper control devices decoded by such names.

3. Application Programming Interface.

The central part of API (see Fig.2) is the Multiprotocol interface allowing an application programmer not to care about underlying protocols. API decides which protocol to use on the basis of the facility part of a DOOCS name. The interface to client programs is based on the communication (EqCall), address (EqAdr) and data (EqData) library classes. The EqData class has a lot of methods to access, change and convert data. The transferred data also contain type, error and length information. With the EqAdr class a client program specifies a facility, a device, a location and a property of a device. The addressing of the servers and server links are done in the library. A server can be on any computer on the Internet. At the time being the client communication interface (EqCall) has four different calls: two - to read data (a blocking, synchronous call and a monitor call), one - to set data and one - to read the actual list of all facilities, devices, locations, properties defined in the control system.

4. Equipment Name Server.

As it was mentioned above the main task of ENS is to store the information about the whole control system and to supply clients (via API) with this information. ENS was designed with the use of basic DOOCS software. Such library classes as EqAdr, EqData as well as ones manipulating files [2] were used without changes. At the same time it was developed new software which allows ENS to build up and intensively use tree-like structures or tables with the necessary information. The complete set of these tables we call the ENS Information Database or simply the ENS Database (see Fig.4). Note that the ENS Database is built with the use of a set of configuration files describing the control system. The DOOCS classes that deal with files help us to build this database very effectively. Configuration files contain the names of all control elements and device servers handling these elements, the names of the computers on which such servers run and protocols supported by them as well as various additional service information. Let us describe now the ENS specific software in more detail.

ENS_Entry class completely defines each control element (ENS entry) into the ENS Database. The last one is built so that every leaf of tree-like structures points to the corresponding and unique ENS_Entry instance.

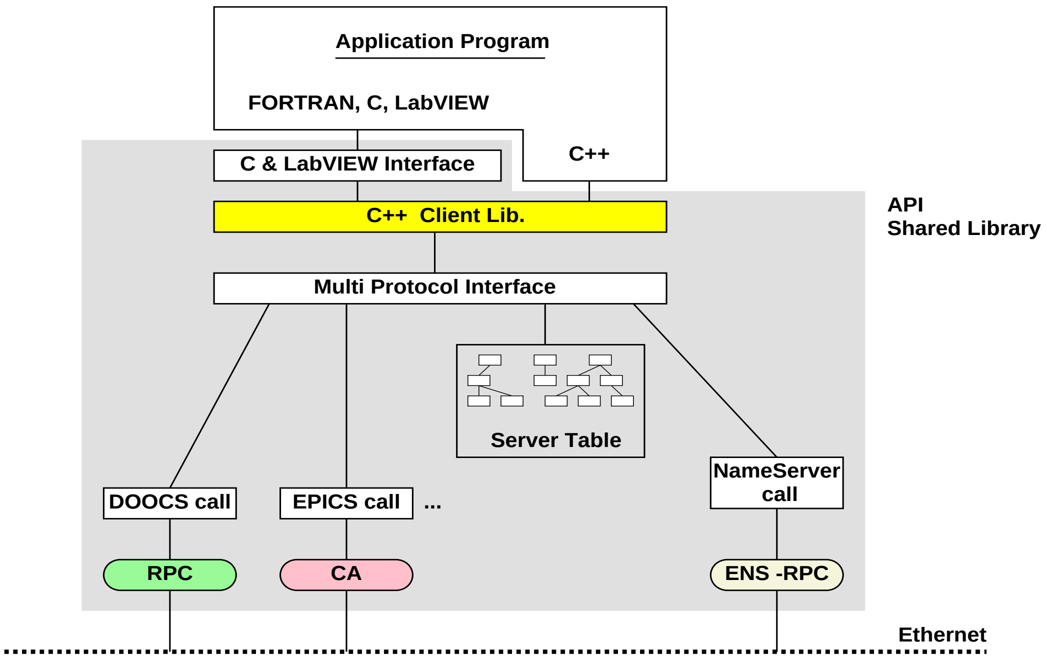

FIGURE 2. *Application Programming Interface.*

In accordance with the DOOCS ideology every instance of the Ens_Facility class is a root of a tree-like structure into the ENS Database. It holds the facility part (or facility entry) of the address of a device server. Ens_Device, Ens_Location and Ens_Property classes hold their specific parts of the address of a device server. In the case if one deals with an EPICS device the Ens_Property also keeps the name of the basic part (without the extension field [3]) of an EPICS channel. At that the extension itself it was naturally to include into the property name (note that the name of an EPICS extension field and the DOOCS property name have practically the same informational and functional meaning). Getting client requests from outside the ENS forms the complete EPICS channel name on the basis of these two parameters.

ENS reacts upon the UNIX signal (-1) by clearing the whole on-line Information Database and building a new ("fresh") one. It is especially useful when one wants to make current changes in the configuration of the control system without significant influencing the running control processes. The standard mechanism of UNIX signals makes it possible to restart ENS with new parameters without preliminary stopping it.

The open access to sensitive devices from all over network is always a reason for the "headache" of control developers. The ENS allows us not only to significantly reduce the information load on API but also relatively simply solve security problems. The list of the users who have the permission to access control devices is prepared by the control personnel and is stored by the ENS in the form of the Authority Table. With the use of this table the tandem API-ENS defines if a client is authorized to perform the requested actions or not. If so, then such actions are performed. Otherwise this client is informed that he is prohibited to do it and then may only ask the list of users authorized for required actions. Such a list gives one a possibility to contact the authorized persons on control problems.

To improve the reliability of the control system the ENS has to run on more than one computer on the network. In such conditions to make the necessary configuration changes and then to restart all ENS is not a simple problem. A lot of actions should be done synchronously on every such a computer. That is why it was developed a special tool (we call it ENS_tool) that allows the automation of all such actions. ENS_tool is based on OpenWindows graphics libraries. It includes the graphical user interface (GUI) as well as a set of Bourne shell scripts and auxiliary subroutines (written in C and C++) providing the information required for this interface. When one invokes ENS_tool then he gets on the computer screen the main interface window (see Fig.3). If one wants then, for example, to restart the ENS on all computers on which it is currently running then it is necessary to push the button "Restart" with the use of a "mouse". ENS_tool also gives one a possibility from one place to edit any configuration file (the "Change" button), add new information into these files ("Add info") and then distribute the changes all over computers housing ENS ("Distribute changes").

5. Conclusion.

First months of the work of the Equipment Name Server in the DOOCS environment show us that the decision to separate the information part of the TTF control system into a special server task allowed one not only to significantly improve the reliability of this system but also make control applications very compact.

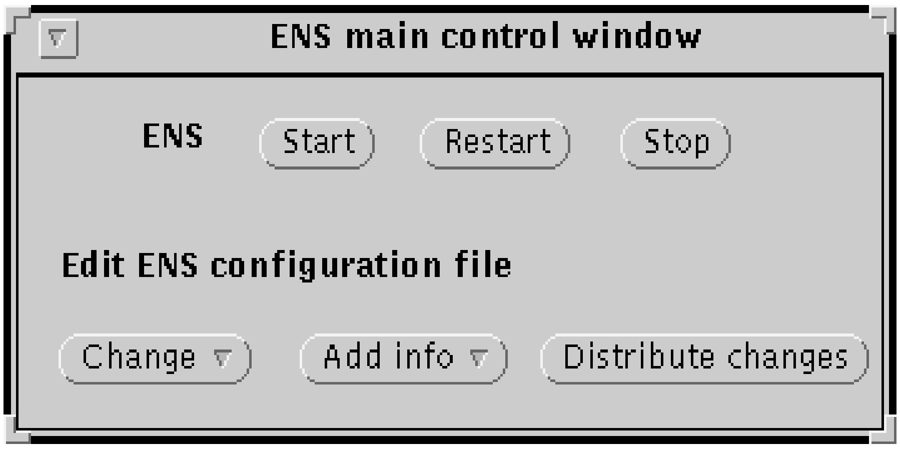

FIGURE 3. ENS_tool main window.

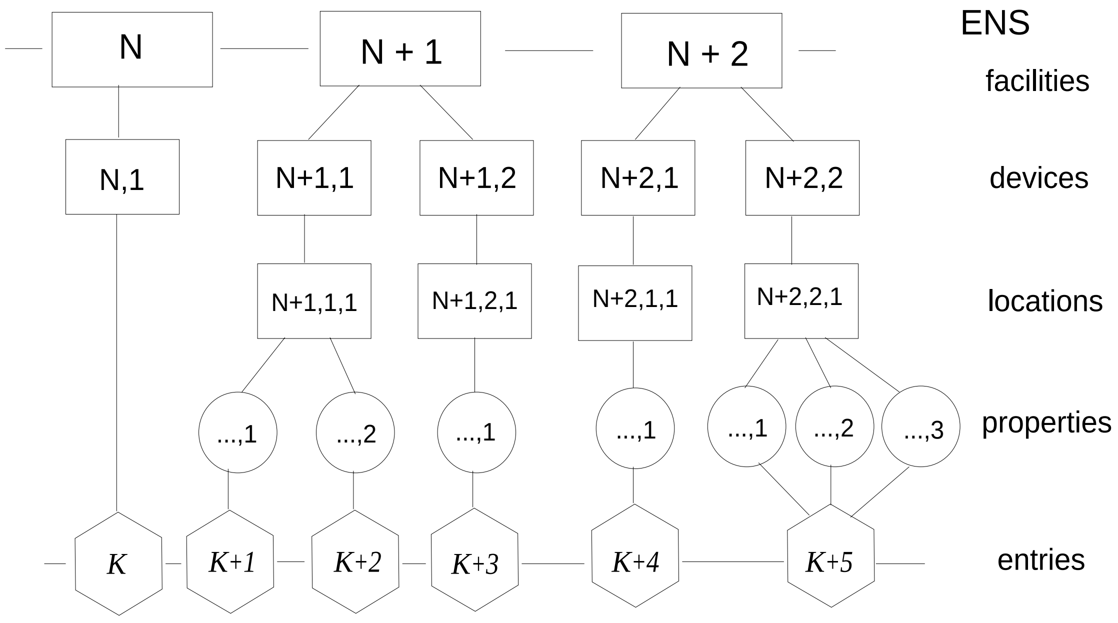

FIGURE 4. Structure of the ENS Information Database.

References

1. TESLA Test Facility Linac - Design Report, Mar. 1995, DESY Print, TESLA 95-02.
2. M.Boehnert, O.Hensler et al. - DOOCS Manual, DESY MVP, 1996, 110 p.
3. L.R.Dalesio et al. - EPICS Architecture, LA-UR-91-3543, Los Alamos, USA, 1991.