6.2 Control System

The following sections describe design considerations, architecture and the components of the CANDLE Control System (CS). The main requirements are operability, reliability, flexibility and to be open to evaluations and hardware modifications. The basic ideas are the following: usage of de-facto standard components, distributed control system, multi-layered structure and modular hardware/software design. Software development is based on object-oriented technology.
Consisting of three layers the architecture implements low-level device servers as independent programs that completely control a number of devices and provide the device data to the network and receive messages from the clients. The subsystems will be highly automated by implementation of finite state machine servers in the middle layer, which will have access to low-level device servers. The application programs in the third layer will have access to the low-level device servers and the middle layer processes.

6.2.1 Architecture

The CANDLE CS will follow the client-server configuration. The main challenge for the design will be to create the software modular with good interfaces to allow easy migrations to new environments. Modular means that some sub modules must be changeable without interference with other parts. It also means to design reusable objects. Basic element in a control system is the communication interface. It defines the Application Program Interface (API) for the data exchange between clients and servers. Because of its important role in a control system, the API must be based on standard protocols. For the network layer this is the TCP/IP Internet protocol. Higher level protocols are transported over the basic TCP/IP services. All client and server programs communicate via it.
Whenever possible standard industrial components should be used for front-end hardware up to the display programs. And the software should be designed with the common tools and programming languages. Object orientation leads to a clean frameworks and follows the industrial paradigm. The architecture will consist of three layers of computers: a top level with display or client programs, a middle layer to implement data bases or sequencers (central computers) and a front-end layer with device servers and I/O (Fig.6.2.1). A CS design goal is to implement a device server as an independent program that completely controls a number of devices and provides the device data to the network and receives messages from the clients.
The upper layer is the interface to the operators. These upper services should be available to the consoles and on the central computers.

Front-end Hardware
The CS should be cost effective. We, therefore, plan to use IBM-type PCs with operating systems Linux and Windows. Most of the device input and output channels are connected via VME modules or fieldbus electronics to the device servers. The VME system is designed as a reliable industrial electronic environment. Almost any processor, analog or digital input/output module and fieldbus interface are available in VME. All data communications between front-end systems, central systems and consoles will be Ethernet based.

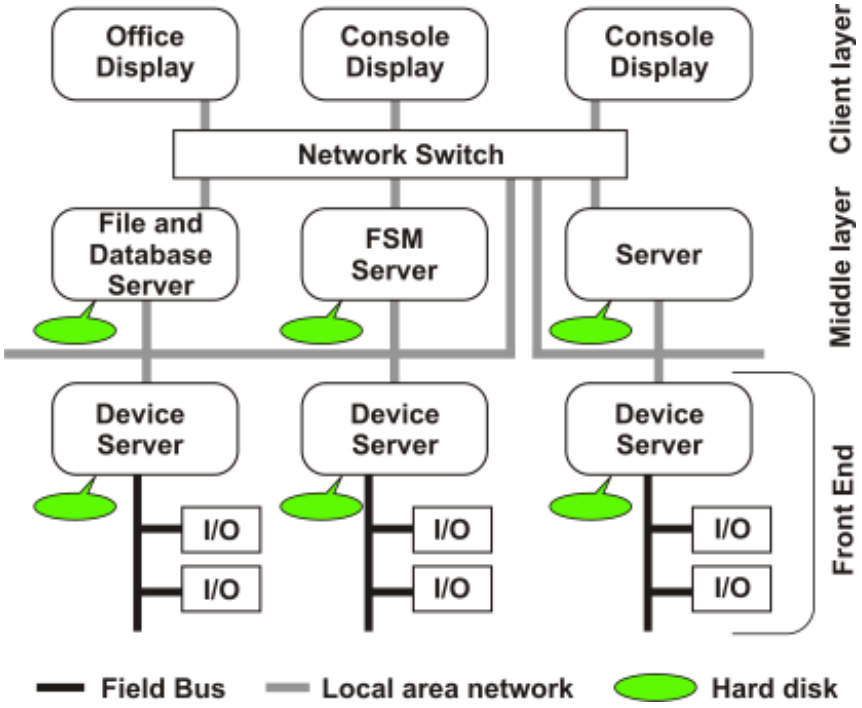

Fig. 6.2.1 The CANDLE Control System architecture.

Server

Sensors and actors (devices) of the accelerator are connected to a server program by a variety of electronics. These programs provide all device services to the network. All requests to a device have to pass to appropriate server program. In general the server resides in VME and communicates to the device by VME module or via a fieldbus. Several independent server processes provide support to various devices of the accelerator. The services vary from simple reads of device data, automatic control of device functions and error handling up to complex calculations of data from several devices. Information from these services will be available on the common Ethernet.

Client

A client is a program that uses services from the servers in the control system. The client programs provide screen graphical interactive display applications and screen drivers. These programs run on the operator consoles and on the central computers. All these programs should use the same communication protocol and the same library to talk to the servers. This is important since it should be avoided to rewrite the same software for different environments. Client software is a wide field and consists of some generic programs (Fig. 6.2.2):

- Tools to display alarms and other device errors,
- Access all data in the system,
- Tools to display historical trends and other plots.

Generic programs will not be changed when the environment changes. They allow access to all devices by standard methods. Other tools are standard programs that allow creation of dedicated applications for:

- generating and displaying synoptic representations of devices,
- storing and recalling machine parameters,

- accessing device data for data analysis frameworks like MATLAB or ROOT,
- interfacing machine simulation programs,
- defining automated sequencing of machine operations.

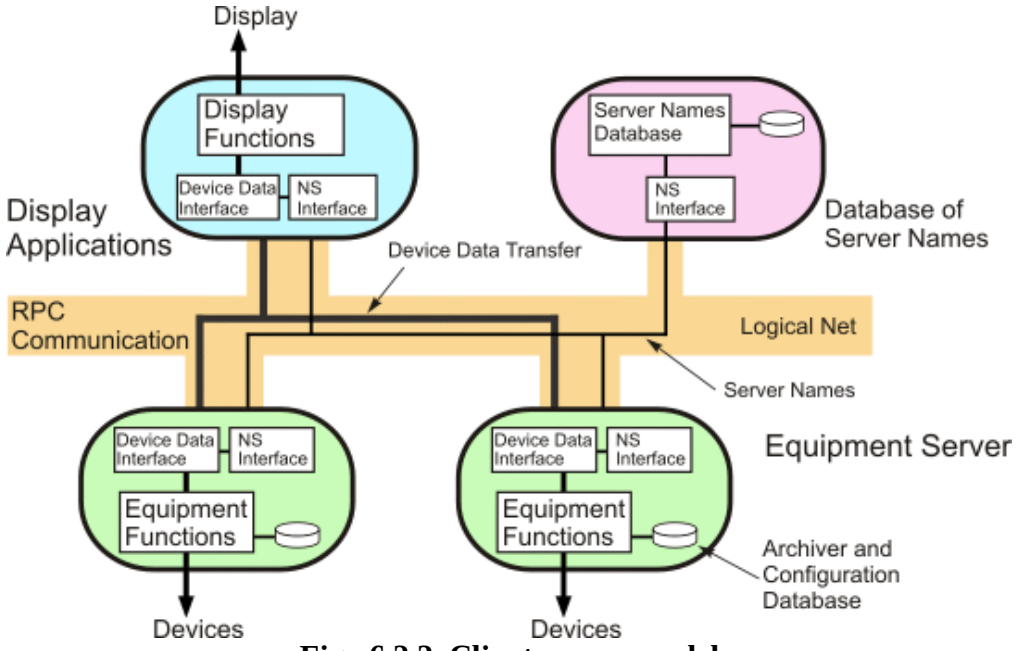

Fig. 6.2.2. Client-server model

6.2.2 Computer Communications

Networks between the consoles, central computers and front-end servers will be based on standard local area network (LAN) technology. This is the 10/100Mbit Ethernet today. Cheap interfaces for all computer types are available. At the front-end this is a fieldbus to connect the local input and output of the devices to the corresponding servers. Standard LAN technology is also used in the links between the servers and all client processes. And the whole network will be connected to the CANDLE LAN.

Fieldbus

A fieldbus is a network that connects sensors, actors or complex front-end input/output devices to the local device servers. There is no single common standard fieldbus on the horizon. On the other hand, control system has to focus on very few standards to reduce the maintenance costs. Modern fielbuses like Profibus or CAN mainly meet our requirements and are convenient to be used in CANDLE. The main merits of the mentioned buses are the high reliability, high-speed deterministic control capabilities and cost effectivity. At the same time these buses are widely used in industry and have various other applications and therefore are supported by numerous leading manufacturers. Some instruments are equipped with the standard GPIB interfaces, which will be used to communicate between the control system and devices such as Voltmeters, Digital Oscilloscopes and Spectrum Analyzers. GBIB/Ethernet adapters will be used to communicate between GBIP devices and ethernet fieldbuses.

6.2.3 Implementation

There are many implementations of control systems that have been developed at various accelerator laboratories, which are too site-specific to be taken into consideration for CANDLE. General control system architectures are of interest. Among those that deal with the console and control layers and the communication between them are Distributed Object Oriented Control System (DOOCS) [2] from DESY and Experimental Physics and Industrial Control System (EPICS) [3] from LANL.
We will evaluate the performance of DOOCS, which has been already ported to the hardware platform of our choice – IBM-PCs in Linux operating system. The DOOCS model is very close (almost the same) to the above description. DOOCS is a distributed control system that was developed mainly for the control of TESLA Test Facility.

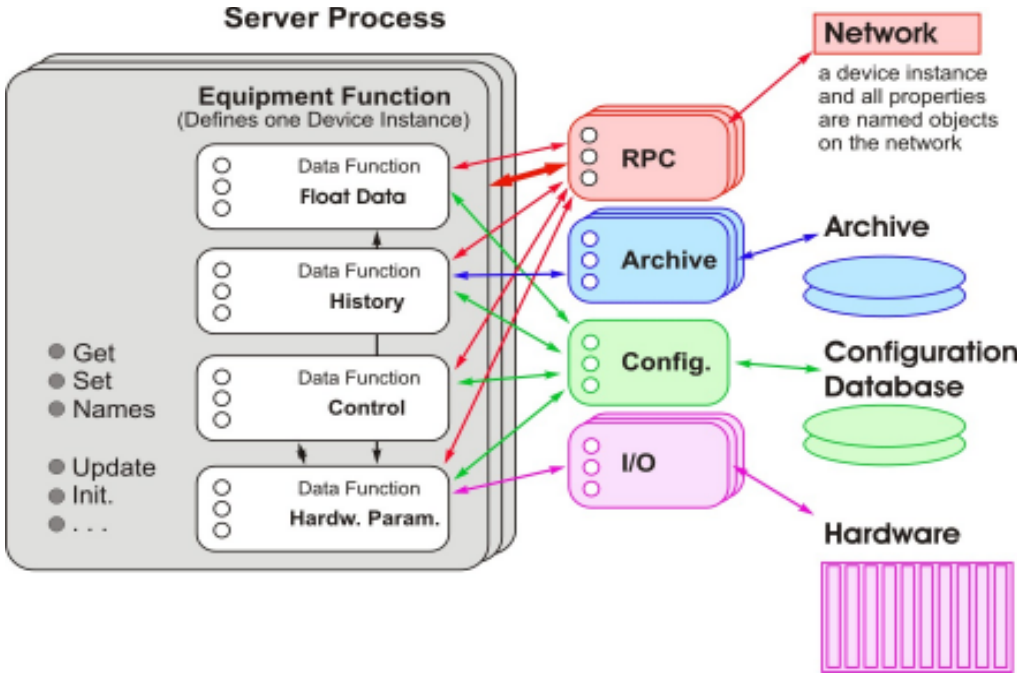

Fig. 6.2.3 Equipment Server Process.

It is an object-oriented system design from the device server level up to the operator console. Class libraries were developed as building blocks for device servers, communication objects and display components. The whole system is written in C++ programming language and runs mainly on Solaris, SunOS and Linux operating systems. The communication is established by a standard set of data and address objects, which are transferred by Remote Procedure Calls (RPC).
This multi-level system consists of both low-level components like device servers, and high-level ones like DOOCS Data Display (DDD) with graphical editor, which was developed to display and control the equipments. DDD allows creation of component libraries in a hierarchical way. The synoptic displays are animated by the status of the devices, sub-windows with detailed information or plots are activated by a single mouse click.
Device servers control different parts of hardware. Server processes are built from a modular library of C++ classes (Fig. 6.2.3). An actual server consists of several entries of a device type at different locations in the system. It can also contain different device types.

Every instance of a device defines a set of properties. These named properties are the access points on the communication network and are implemented as data objects.
The server library is a collection of various classes to provide the network access and an access to the hardware, such as VME, Profibus, SEDAC and CAN. In addition, there are classes to read/write the configuration file and to archive the data types on a hard disk.
The finite state machine concept [4] was introduced into the DOOCS, which resulted in creation of a special class of servers, called FSM servers, to automate the subsystems and to provide high-level control employing several DOOCS device servers.
DOOCS client API is able to handle different network protocols in addition to the DOOCS-RPC protocol. So far, the EPICS Channel Access calls are supported as a second protocol within the API.
Available client programs are DDD, LABVIEW, MATLAB, ROOT and separate tools, which were designed for specific purposes (error-handling, save and restore, etc.).

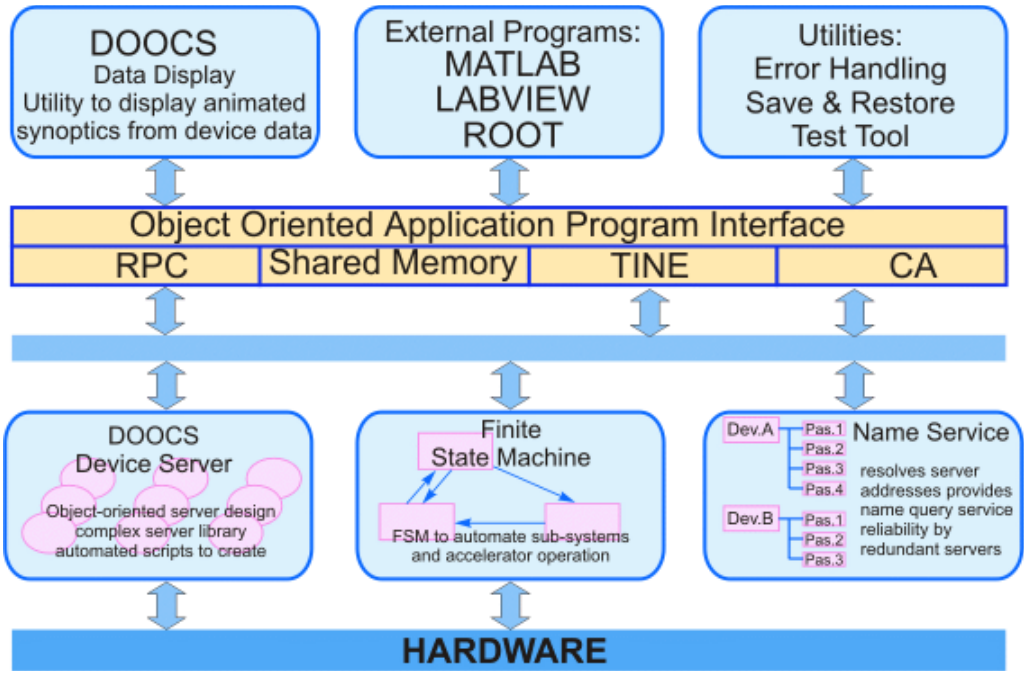

Fig. 6.2.4. Distributed object oriented control system.

Database

All the machine design information, hardware configuration and calibration data, machine optics parameters will be stored in databases. In order to achieve a better understanding of the machine, it is desirable to have a more complete archive of the accelerator data. For this purpose we will use an object oriented data analysis system ROOT from CERN.

Concluding remarks

Looking through an existing accelerator control systems we choose products that are able to achieve maximal portability and longevity and which will remain alive even after a new generation of PCs or a change of operating systems. Such an approach reduces cost and also saves the development time.
The user-friendly configuration and reliability of DOOCS has been demonstrated during several years of operation at TESLA Test Facility and the HERA vacuum system.

The object-oriented method is not just a good programming practice but it also leads to transparent system design in the whole field of control system, i.e. device hardware, device servers, user interfaces and program libraries.
The great experience in operating an accelerator with object oriented technologies at DESY is a good basis for creation of reliable control system for the CANDLE Light Source.

References

1. V. M. Tsakanov, V. Avagyan, V. Ayvazyan, G. Amatuni, B. Grigoryan, M. Ivanyan, E. Laziev, Y. Martirosyan, R. Mikaelyan, S. Tatikian and A. Vardanyan, ``Center for the Advancement of Natural discoveries using Light Emission: A New project for 3 GeV Intermediate Energy Light Source in the Republic of Armenia", Review of Scientific Instruments, V. 73, N3, 2002;
2. G. Grygiel, O. Hensler, K. Rehlich, ``DOOCS: a Distributed Object Oriented Control System on PC's and Workstations", ICALEPCS 97, Beijing, (1997); see also http://tesla.desy.de/doocs/
3. L. R. Dalesio, M. R. Kraimer, A. J. Kozubal, "EPICS Architecture", ICALEPCS 91, Tsukuba, Japan, 278-282 (1991).
4. V. Ayvazyan, K. Rehlich, S. N. Simrock, N. Sturm, ``Finite State Machine Implementation to Automate RF Operation at the TESLA Test Facility", Proc. PAC'2001, Chicago, 286-288 (2001)
5. V. Ayvazyan, K. Rehlich, S.N. Simrock, G. Amatuni, A. Yayloyan, “The Design of the Control System”, Proc. of the 8th EPAC, La Villete, France (to be published).