FIRST EXPERIENCES WITH THE CONTROL SYSTEM FOR THE ACCELERATOR OF ANKA

H. Schieler*, A. Weindl,
Forschungszentrum Karlsruhe, PEA, Postfach 3640, D-76021 Karlsruhe, Germany
B. Jeram, M. Juras, K. Kenda, I. Kriznar, B. Lesjak, K. Mele, T. Milharcic, M. Perko, M. Peternel, U. Platise, M. Plesko, M. Smolej, R. Sabjan, G. Tkacik, I. Verstovsek, B. Zorko, K. Zagar,
J. Stefan Institute, Ljubljana, Slovenia

Abstract

ANKA [1] is a 2.5 GeV synchrotron radiation light source being built in Karlsruhe, Germany. The control system for the accelerator is based on the three-tier standard model architecture. However, modern products based on standards in distributed objects and networking are applied in addition to low-cost hardware including PCs. We use the LonWorks field bus network with intelligent nodes and standard I/O modules to connect the individual devices directly to PCs that run device servers under Windows NT. Those PCs act as WWW servers for data transmission, application distribution and documentation retrieval. Applications in the control room run also on Windows NT hosts as WWW clients. However, they could run in any Web-browser on any platform, because all operator control is performed through a Web-browser with Java applets/applications. The communication with the control system data servers is done through CORBA. CORBA objects are wrapped in JavaBeans which are simply connected with commercial data-manipulation and visualization Beans into full-fledged applications or applets. First experiences with this control system during the operation of the ANKA microtron are presented.

1 INTRODUCTION

The control system (CS) of the accelerator is based on client and server PC's running under WinNT and the LonWorks field bus with intelligent nodes and standard I/O modules to connect the individual devices directly to the server PC's. These server PC's communicate via CORBA with client PC's in the control room. All operator control is performed through Java applets/applications.

The first real-world test of the system was on the 53 MeV microtron of ANKA during the period from October 98 to March 99, controlling it's vacuum system and power supplies successfully.

* Email: schieler@anka.fzk.de

2 CONTROL SYSTEM ARCHITECTURE

The control system architecture is planed to be as homogeneous as possible from the operator's point of view. The CS is designed to use existing intranet/internet infrastructure and web technologies such as HTML/HTTP, web browsers/servers with Java and CORBA/IIOP. This decision was made because nowadays a large proportion of people are familiar with web browsers and because the WWW standards provide equal user interface to any information regardless of its type.

Technically, the control system follows the three-tier standard model architecture:

1. the field bus layer with devices;
2. the process control layer with accelerator objects;
3. the visualization layer with control GUIs.

Conceptually, it is composed of two layers, connected through the device servers:

1. the field bus layer with asynchronous event-driven data acquisition/control;
2. the object oriented layer with a model of devices where the client talks to devices as if they were there.

Following, an overview of the different layers is given. Further details can be obtained from [2],[3],[4],[5].

2.1 The Field Bus Layer

The LonWorks field bus is a powerful data acquisition and networking system that connects up to 32000 intelligent nodes with I/O modules directly to a PC that runs under Windows 98/NT. The PC interface is connected to all device interfaces through a twisted pair cable. On the Server PC, the LonWorks Network Service (LNS) management tool is used to register each node. LonWorks technology offers a complete network system in hardware and software in a single micro-controller (the Neuron chip) and eliminates any need for network programming. Many LonWorks boards are commercially available, however it is also relatively straightforward to interface own designs to the Neuron chip. After a careful

analysis of the ANKA CS I/O requirements, three I/O boards were designed that cover most cases:

- **Ariadne** is a serial interface board, which supports EIA-232, EIA-422 and EIA-485 standards at maximum baud-rate of 115kbps. It has 16k bytes long buffers on receive and transmit lines and an on-board power supply unit that can source current from 230 V AC line, unregulated 7-12 V DC and regulated 5 V DC.
- **Zeus** is a high precision I/O card with a 16-bit ADC and DAC, DAC trigger input, and optically isolated digital channels (8 inputs and 8 outputs). Four analog channels with a nominal sampling frequency of 1 kHz are multiplexed to the ADC, which is oversampled at 4 kHz to ensure 0.3 LSB precision. The DAC operates at a maximum rate of 10 kHz. An additional on-board peripheral micro-controller, specially designed to control booster and storage ring power supplies includes: a) function generator synchronized with DAC trigger input, b) 32 kb of memory to buffer DAC function and ADC data and c) peripheral self test.
- **Hera** is a digital I/O card with 24 inputs (50 mA), 8 in-/outputs (50 mA) and 8 outputs (solid state relays, 1 A). All I/O's are optically isolated. Two operating modes are provided for inputs and in-/outputs. There is also a 16-bit frequency counter with range of 0-100 kHz (absolute error 1.53 Hz).

The software written for the Neuron chips implements quite complex functions such as state machine and alarms, synchronous ramping in 0.1 millisecond steps and others. The communication to the PC is done by the device servers using the LCA (LonWorks Component Architecture) and LNS library through network variables and remote command invocation, which allows also network management. On top of this, additional functionality was written, such as a template compiler and a file transfer protocol, which loads all run-time constants at start-up from a centralized database. Thus, each constant that is used by the Neuron and the PC clients and servers is stored in one place only, which is the static database.

2.2 The Process Control Layer

Controlled devices are modeled as objects residing on device servers that run on the process control layer. The objects are exposed to remote clients through their interface only. The Accelerator Control Interface (ACI), a language independent collection of interfaces based on network distributed objects is using the CORBA standard. All common accelerator components such as power supplies, vacuum, RF, position and current monitors are defined by means of functions and parameters. The devices are described according to CORBA with the Interface Definition Language (IDL), which presents a language-independent way of defining object interfaces. Each controlled parameter, called device property, is an object by itself, implementing atomic actions such as get/set, increment/decrement, etc. All constants related to a property such as min/max, name, description, etc. are obtained from the property directly by means of remote methods - no direct database access is necessary. Values of the properties are updated asynchronously by means of monitor objects. The ACI is meant to be a standardised interface so that applications and pieces of control systems can be hooked to it from either side. The ACI does not replace existing control system architectures and frameworks but rather tries to use their features in order to be as compatible as possible to those systems. CORBA automatically generates the appropriate communication libraries. There is no need to write other API libraries. CORBA is chosen for this reason and due to its platform and language independence. The need for speed and the necessity to communicate with external drivers require the servers to be written in C++.

The communication between clients and devices is completely asynchronous. Server's responses to client's requests are made via callbacks. There is also a possibility of using "repeated callbacks" – called monitors. The idea is that clients are able to register with servers about which data they require and how frequently it has to be obtained.

2.3 The Visualization Layer

Every operator's interaction with the control system goes through the control GUIs, written as Java applets/ applications. These applications are build around the Java bean model. A Java bean is a component that can be manipulated in a visual builder environment; beans can be graphically arranged and connections between them established. The latter include, for example, event-to-method connections, where the event in one bean triggers the method in the other; property-to-method connections, where a change in property triggers the method; property-to-property connections and so on. Such environments enable the programmer to build an application without typing a single line of code.

Any accelerator application is composed of two types of beans:

- visual beans (GUI objects like buttons, gauges, charts, ...);
- accelerator beans (called Abeans; each Abean represents a real accelerator device).

An Abean encapsulates all remote calls from the client to a device server of the process control layer. Thus the network is invisible to the user of Abeans. Tasks of an Abean include opening the connection and performing the function calls on remote objects; report and manage all errors/exceptions/timeouts arising from network communication, provide handles for asynchronous messages, etc.

Visual beans are mostly commercial products. Therefore the work done in building a control panel for a device consists mostly of connecting the appropriate Abean and commercial visual beans in a visual builder. The developing time is low, of the order of hours for a panel, or one or two days for a full-fledged application that instantiates and interconnects many Abeans - even of different devices.

3 THE CONTROL SYSTEM SETUP FOR THE ANKA MICROTRON

The microtron is the first part of the ANKA accelerator available. In fact, this is a less extensive setup for the CS than the complete storage ring will be, but it is the first real-world use of the CS and the first possibility to collect experience. The CS for the microtron consists of the following hardware and software components:

- Two PCs, a Client and a Server PC, both running under WinNT 4.0 with PII/266MHz CPUs and 128 MB of RAM, connected via TCP/IP.
- The Client PC runs all client GUIs, written as Java applets/applications. They are designed as tables, including all devices of one type (e.g.: all power supplies) or as panels, one for each single device.
- All device server C++ programs are running on the Server PC. For every kind of device type there is one server application, acting as software interface between client GUI and device, delivering values to clients and commands to devices.
- Server programs and client applications communicate using the CORBA implementation of Visibroker (Inprise).
- One LonWorks field bus interface card in the Server PC is connected to all I/O interfaces through one twisted pair cable. This is the field bus branch the device server programs use to communicate with the following I/O interfaces and their corresponding devices:
  a) 5 Ariadne serial interfaces for power supplies,
  b) 30 Zeus I/O boards for power supplies and vacuum gauges,
  c) 2 Hera digital I/O boards for vacuum valves, -pumps and -pump power supplies.

4 FIRST EXPERIENCES

During the first test for the CS with the microtron setup, no principle problems ocurred, and so it was possible to run the microtron successfully in this environment.

The field bus and device part implements most work, because all nodes must be registered and database values for every device have to be set. Once done, only device server executables and clients must be started for running the configurated CS.

Control System GUIs are clearly designed with logical, mostly self-explaining functionality and displays. Representing one single device, a panel shows virtually all necessary values and properties one may expect from the real device. Saving desktop place and preventing confusion, a table gives a comprehensive overview, listing all devices of one type and offering the possibility to control several devices at once.

Network communication raised no serious problems, neither the quantity of LonWorks nodes on the field bus branch, nor CORBA. Operators could work with clients, not caring about network details or problems.

Using Abeans in the Java Bean concept allows an easy extension of new client applications. Only Java knowledge is necessary, while client-server communication details are hidden in the Abeans. Connecting Abeans with beans using visual programming can deliver a lot of values and funcionality. So much time is saved because of less code typing.

Fundamental changes/extensions in the software are sophisticated, because much knowledge is necessary to manage all layers of the CS.

The three different types of I/O interface boards are enough for running the microtron, but for future purposes additional interface types (e.g.: GPIB to TCP/IP converter) may be required.

PC requirements are high, because the CS needs a lot of resources. Each actual server or client application uses about 20 MB of RAM, some even more.

5 CONCLUSION

In general, the first real-world test of the Control System was successful. There were no principle problems controlling the microtrons vacuum system and power supplies, so that the commissioning of the microtron was possible in this environment.

Of course, the present microtron version of the CS software is not perfect, therefore some changes and extensions have to be made. The CS has to be optimized concerning performance and user-friendliness and the control of RF systems and the power supply ramping has to be added for further use in the booster synchrotron and the storage ring of the ANKA light source.

6 REFERENCES

[1] D. Einfeld et al.,"Status of the accelerator for the 2.7 GeV light source ANKA in Karlsruhe", Proc. PAC '99, New York , (1999)

[2] M. Dach et al., "A Control System Based on Web, Java, CORBA, and Fieldbus Technologies", PCaPAC99 workshop, Tsukuba, (January 1999)

[3] K. Kenda et al., "I/O Control with PC and Fieldbus", PCaPAC99 workshop, Tsukuba, (January 1999)

[4] M. Plesko, "Implementing Distributed Controlled Objects with CORBA", PCaPAC99 workshop, Tsukuba, (January 1999)

[5] G. Tkacik et al., "Java Beans of Accelerator Devices for Rapid Application Development", PCaPAC99 workshop, Tsukuba, (January 1999)