THE CONTROL SYSTEM FOR THE ACCELERATOR OF ANKA

B. Jeram, M. Juras, K. Kenda, T. Milharcic, G. Mavric, M. Peternel, U. Platise, M. Plesko, R. Sabjan, M. Smolej, G. Tkacik,
J. Stefan Institute, Ljubljana, Slovenia, e-mail: mark.plesko@ijs.si
H. Schieler, Forschungszentrum Karlsruhe, Germany, e-mail schieler@anka.fzk.de

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. This keeps development costs at a minimum. Instead of employing VME, 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.

1 INTRODUCTION

We want to have a control system that will 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 [2]. 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.

The architecture of the CS is based on the Standard model. Essentially the same thing is called *a three-tier architecture* in the world of databases:

1. the visualization layer with control GUI;
2. the process control layer with accelerator objects;
3. the fieldbus layer with devices.

2 THE VISUALIZATION LAYER

Every operator’s interaction with the control system will go through a web browser:

- control GUI (Java / CORBA)
- logbook forms (Java / JavaScript / CGI)
- help, documentation (HTML)
- notification (e-mail)

We have chosen Java because it is a modern object oriented programming language, it has well defined data types and API (Application Programming Interface), it allows easy use of graphic widgets, threads and other system tools without having to know the specifics of a given platform. Java is also an interpreted language, so it is a little slower than compiled languages like C++, but we found out that by using JIT (Just-In-Time) compilers it is fast enough for our needs.

The applications will be build around Sun's JavaBeans[3] model. A JavaBean 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 windows, buttons, gauges, charts and the like);
- device Beans.

A device Bean 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 device Beans. Each device (which is presented in the control system as a remote object - see next section) has a corresponding device Bean. Tasks of a device Bean 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 will mostly be commercial products. Therefore the work done in building a control panel for a device will consist mostly of connecting the appropriate device Bean 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 interconnect many device Beans - even of different devices**.**

3 THE PROCESS CONTROL LAYER

Controlled devices are modeled as objects[4] residing on device servers that run on the process control layer. The device server are implemented along the guidelines of the

TACO system[5]. The objects are exposed to remote clients through their interface only, using the CORBA architecture[6]. CORBA automatically generates the appropriate communication libraries. There is no need to write other API libraries. We have chosen CORBA for this reason and due to its platform and language independence[7].

The need for speed and the necessity to communicate with external drivers require the servers to be written in C++. In the future, when we find appropriate Java development tools, we might write device servers in Java using JNI (Java Native Interface) and a native code compiler. The result will be only one platform and development tool, which will greatly simplify the maintenance involved.

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.

The server is independent of the underlying fieldbus. Therefore our Web-based concept can be used in conjunction with any other control system, e.g. EPICS, etc.

Since the device Bean encapsulates all the communication details with the server, it can easily be rewritten should we want to use a different interface instead of CORBA, with no modifications to the rest of the client code. In that way that we can replace our current ORB with any other ORB or even with an entirely different architecture, like Sun's RMI[8], Microsoft's DCOM[9] or pure sockets.

4 DATABASES

Our design of the control system involves three databases:

- static database that stores configuration parameters like names, constants, calibration coefficients, attributes, alarm levels, fieldbus addresses, etc.
- snap-shot database that stores the state (i.e. all settings) of the machine
- historic database that logs data over long periods of time

The main idea of three three-tier architecture is that clients don't access the database directly but through the CORBA server. Each object is responsible to provide data that belongs to it. The CORBA objects therefore implement commands that return all data from the static database.

Long-term history data. are stored into a relational database and retrieved off-line through dedicated applications.

Also the snap-shot database is used and managed through a dedicated application. We are still investigating whether we should use a relational or an object oriented database for the snap-shot data.

5 THE FIELDBUS LAYER

Due to the relatively modest requirements of ANKA, we have found[10] that instead of having dumb devices and a strong middle layer, we bring intelligence to the devices and avoid the complexity of VME. This is possible also because most equipment has already some intelligent control. It is just necessary to combine them with a proper fieldbus.

Since a light source is a relatively static machine, there is no need for hard real-time control. Therefore we have decided to opt for the LonWorks[11] fieldbus, because it offers a complete network system in hardware and software in a single micro-controller (the Neuron chip) and eliminates any need for network programming.

The software on the Neuron chips implements quite complex functions such as complete magnet power supply control including state machine and alarms and synchronous ramping of several power supplies where their current is increased bit-by-bit in 1 ms steps.

After a careful analysis of all I/O requirements of ANKA, three I/O boards were designed that cover all cases. All boards are made in SMD technology, size 160mm x 100mm (Europa format), 5V supply voltage, with I/O connectors at the back and the LonWorks connector at the front.

One board, called Ariadne, is just a serial interface that is used to connect commercial instruments to the fieldbus. The other two boards are more specific:

<u>Zeus</u>

The board contains one DAC, four multiplexed ADC inputs, eight digital inputs and eight digital outputs, a function generator and a trigger input. The interface to the power supply is pin-compatible to the BESSY ADA16 board. The DAC and the ADC have a resolution of 16 bits over the interval of 0-10V or ±10V and have differential outputs and inputs, respectively. The function generator can set the DAC after a predefined curve with a speed of 0.1 ms per step. The ADC measures up to 1000 measurements per second per channel.

<u>Hera</u>

The board has 24 digital inputs 8 digital input/outputs (a channel can be either an input or an output) and 8 solid state relay digital outputs. The solid state relays can switch currents up to 1.5 Amperes. All channels are circuited as open collectors. The inputs have over-voltage protection with a Zener diode and a varistor. The inputs are read out directly and multiplexed into a 100 kHz counter with a resolution of 16 bits. The selection of channels and the control of the multiplexer is done by the Neuron chip.

6 PERFORMANCE

Old analog control systems allowed to control parameters smoothly by the use of knobs from the control room. Most of the current control systems don't support this, because of slow network communications. This is changing now as even the object oriented CORBA takes only few milliseconds to execute a remote method. For the operator to get an analog feel, about 25 set/read pairs are necessary, just like on a TV screen, i.e. a turn-around time of 50 ms or less.

Our measurements on the complete control system[2,10], which starts with a Java application at the operator console, communicates through CORBA with the server, which in turn uses LonWorks to communicate with the Neuron chip, show a turn-around time of 22 to 40 ms. If we compare this result with the result for Java - CORBA communication, approx. 4ms, we find that Java is indeed fast enough for our control system.

7 CONCLUSIONS

The analysis of the subject and the working prototype[12] clearly demonstrate that the design of a control system based on Web-technology and commercial fieldbus solutions is a good one and ripe for deployment. More important, though, is the global perspective: such technologies are (or at least will be) widely supported in the near future, and the investment in them seems like a worthwhile idea. The important aspects of communication, such as security, database access, transparent method invocations etc. are already precisely defined by the existing standards and have their implementations provided by the software vendors; the only remaining thing that needs to be done is to pick the correct pieces of software and integrate it in a reliable, reusable and easy-to-maintain manner.

8 ACKNOWLEDGMENTS

We sincerely thank Steve Hunt for his on-going help and for believing in our ideas.

9. REFERENCES

[1] D. Einfeld et al., ANKA - Status of the 2.5 GeV Synchrotron Light Source at Forschungszentrum Karlsruhe, Proc. PAC 97, Vancouver 1997.

[2] B. Jeram et al., A Control System Based on WWW-Technologies, Proc. ICALEPS97, Beijing 1997

[3] http://splash.javasoft.com/beans/WhitePaper.html

[4] M.Plesko, The CORBA IDL Interface for Accelerator Control, this conference.

[5] A. Götz et al., TACO: An object oriented system for PC's running Linux, Windows/NT, OS-9, LynxOS or VxWorks,,PCs and Particle Accelerator Control (PCaPAC) Workshop, DESY Hamburg, 1996.

[6] http://www.omg.org/

[7] S. Hunt, B. Jeram, M. Plesko, The Implementation of the OO Control System API with CORBA, Proc. ICALEPS97, Beijing 1997

[8] http://www.microsoft.com/activex/dcom-f.htm

[9] http://www.javasoft.com/products/jdk/rmi/index.html

[10] B. Jeram, G. Mavric, M. Plesko, M. Smolej, Experience with LonWorks as a Fieldbus for the Light Source ANKA, Proc. ICALEPS97, Beijing 1997

[11] The ‘95-’96 Echelon LonWorks Product Databook, Echelon Corporation, 1995.

[12] see http://kgb.ijs.si