A CONTROL SYSTEM BASED ON WEB, JAVA, CORBA AND FIELDBUS TECHNOLOGIES

M. Dach*, S. Hunt*, B. Jeram, M. Juras, K. Kenda, I. Kriznar, B. Lesjak, K. Mele, T. Milharcic, M. Perko, M. Peternel, U. Platise, M. Plesko, R. Sabjan, H. Schieler[+], M. Smolej, G. Tkacik, I. Verstovsek, B. Zorko, K. Zagar, J. Stefan Institute, Ljubljana[·]
e-mail mark.plesko@ijs.si, http://kgb.ijs.si

Abstract

We present the control system for the light source ANKA, which builds on the three-tier standard model architecture. Modern products based on standards in distributed objects and networking are applied in addition to low-cost hardware including PCs. The LonWorks field bus network with intelligent nodes and standard I/O modules connect the individual devices directly to PCs. Those PCs act as Web servers for data transmission, application distribution and documentation retrieval. Other PCs on the net run Web browsers with Java clients. The communication with the control system data servers is done through CORBA. CORBA objects are wrapped into JavaBeans, which are connected with commercial data-manipulation and visualisation Beans using visual tools or programmatically. The CORBA objects and JavaBeans are generic models of controlled data that can be used at any other control system. The Java applications are based on those objects only and can thus be run on any other accelerator.

1 INTRODUCTION

The control system uses existing intranet/internet infrastructure and web technologies such as HTML/HTTP, web browsers/servers with Java and CORBA to the maximum extent. This decision was made not only because nowadays a large proportion of people are familiar with web browsers, but because there are many Web-based commercial tools and libraries available that can be used as building blocks of the control system, thus reducing the development time.

As it is – as yet – too costly to have one Internet node per controlled device, the LonWorks fieldbus is used to network controlled devices. The gateway between the fieldbus and the Internet Protocol (IP) is done via software (named device server) running on PCs.

Owing to the fact that many commercial components exist, great care has been taken in using them as often as possible, while keeping the number of different products small. Table 1 gives an overview of products used. It can be noted that only those components have been developed in-house, for which no commercial alternative exist. This comprises components, which are specific to accelerators, such as API, data server and the I/O boards.

Table: Products used in the control system.

usage	**Product**	**Source**
consoles	*PCs	Many
operating system	Windows NT	Microsoft
panels	*Java	Sun
visual programming	VisualAgeJava	IBM
widgets, plot/trend	JClass Beans	KLGroup
accelerator API	*Abeans*	*in-house*
process computer	*PCs	many
operating system	Windows NT	Microsoft
Internet communication	*CORBA	Inprise
fieldbus management	LCA/LNS	Echelon
archive database	*SQL	many
config. database	*text-files*	*in-house*
data server	*device server*	*in-house*
fieldbus	*LonWorks	Echelon
microcontroller	*Neuron	Echelon
programming language	*Neuron C	Echelon
crossdeveloping tool	LonBuilder	Echelon
I/O boards	*Zeus,Hera*	*in-house*

2 ARCHITECTURE

2.1 Design Considerations

Some important design decisions are based on the particularities of the ANKA project

- small machine: 100 m circumference, 2.5 GeV, medium emittance
- low cost (15 M US$), short time-scale (3 years)
- outsourced subsystems (injector, RF-system)
- small staff, also for operation

The consequences for the control system are:

- must avoid extensive development

[·] *PSI, Villigen, Switzerland, [+]FZK, Karlsruhe, Germany

- needs to connect seamlessly to externally developed subsystems
- clean and simple concept for later maintenance

It was decided that performance can be sacrificed and interoperability among components is sufficiently important to tolerate overheads. Furthermore, generality and flexibility, often requested because of missing or insufficient advance specifications, have been limited. Based on experiences from similar machines, a list of software and hardware interfaces has been established, which the designers of equipment had to follow. Should a custom implementation be inevitable, then it must be hidden from the rest of the control system at the lowest possible level, i.e. on the fieldbus node, be it in hardware or software.

2.2 Design Simplifications

These directions worked very well so that a careful analysis revealed that only three different types of fieldbus boards are necessary: DAC/ADC combo, 40 channel I/O and serial interface. More complex I/O, such as image processing, PLC drivers and GPIB-Ethernet converters, is handled directly by the PC.

Although the control system is designed to support all primitive data types for controlled values, only double and unsigned integer did suffice. In turn, the number of necessary GUI components is small. Not only were the types of values limited, also their properties were defined and fixed in advance. This means that a controlled value is either read-write or read-only and that it has a name, a minimum, a maximum, etc. The list of properties was made large and configurable enough to cover .all cases, yet the sole fact that it is fixed in advanced simplified the design of clients and servers and databases.

The only disadvantage of this approach is that the fieldbus programs are quite complex and different for each device, i.e. not one program per board type but one per device type. Also the device servers and the API is then device specific, however due to the pre-defined value types and structures, all those components can be built easily and fast. In fact, a wizard has been written which generated device servers based on device interface definitions. The Java API libraries, JavaBeans, need less than one page of extra code per device. So in total, the concept proved to be advantageous.

This approach is of course by no means novel. EPICS for example provides similar basic value types, yet is open for user defined value types and properties. The main differences of our approach with respect to EPICS are:

- EPICS in principle assumes that nodes are not intelligent – EPICS does all the state processing and event generation, while in our system this is done by the fieldbus nodes
- EPICS is based on channels, i.e. individual control values, not on devices, while in our system devices are implemented already on the fieldbus level.

2.3 General Layout

Technically, the control system follows the standard model three-layer architecture. Conceptually, it is composed of two layers, connected through the device server:

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

The device server adds little functionality apart from serving as a gateway and multicasting events to many clients. But due to its existence it hides all details of the fieldbus system to the client, although the event sources reside actually on the fieldbus nodes.

The object-oriented layer, which is the only one that is seen by the user of the control system, is based on the Web, Java and CORBA. The goal is that all user actions can be done from a Web browser. This includes, apart from device control, also writing and reading the logbook, checking alarms, searching the data log, reading help and documentation, etc. Tasks that deal with the management of the control system, however, will generally not be made available through a Web browser. The security implications are severe, not so much because of external security breaches, which are unlikely, but because management should be kept in the hands of a small group of people.

3 COMPONENTS OF THE CONTROL SYSTEM

Separate contributions to this workshop describe the use of the fundamental components LonWorks [1], CORBA [2], and Java [3] in great detail. Following are short descriptions of control system components that provide the framework and services to the fundamental components.

3.1 Databases

Our design of the control system involves three databases, which accomplish completely different tasks.

The **static database** stores configuration parameters like names, constants, calibration coefficients, attributes, alarm levels, fieldbus addresses, etc.. 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. Since the properties of the controlled values have been determined in advance, each property is obtained conveniently through a call to a CORBA method with the same name as the property. The static data used by the LonWorks nodes are stored in the static database, too. They are downloaded to the nodes at start-up of the device server with a generic procedure involving a template compiler and a LonWorks ftp-like protocol. Data that are used both by the device server and the nodes are located only in one place of the database to ensure consistency. Normal users of the control system should not modify the static database. The configuration client runs therefore on the device server level only. As it does not have to run in a Web browser, it has been written in C++ for efficiency reasons. It provides handy actions such as versioning, rollback, locking. The database is hierarchically organised with directories and text-files, benefiting from the Windows NT file system. The client therefore uses MFC libraries of the Windows Explorer and Notepad to display and edit the database content.

The **historic database** logs data over long periods of time. Long-term history data are stored into a relational database and retrieved off-line. The device servers keep a short-term history for all controlled values in a ring buffer during run-time. The length of the history is set in the static database and comprises usually about 1000 seconds. The buffer is periodically sent to the historic database. As the type of queries to the historic database can not be foreseen, a general SQL tool will be provided to allow the user to construct any imaginable query.

The **snapshot database** stores the state (i.e. all settings) of the machine. The snapshot database is used and managed through a dedicated application. This database uses a CORBA-based location service to discover all devices during run-time. It then uses Java introspection to determine all settable properties of a device. Actions of the application include save/restore, view/edit/merge, etc. Due to the small amount of data at ANKA, each entry in the snapshot database will be stored as a text file. The application will provide means to search the files in different ways. At a later stage a more powerful relational database could be used.

3.2 System Management

The management of the fieldbus is done partially with tools that form part of the developing environment of LonWorks and partially through the device server.

The device server and the clients are managed with a special CORBA client which can start/stop a server, update static data on a running server disable/enable one or several devices, disable/enable or stop monitors, monitor device server events (client connect/disconnect, errors, etc.). As in the case of the static database, the management client is written in C++.

The security of transactions will be very simple but effective. Each client that wants to connect to a device must first register with the management server. Only clients with the correct password and host IP number will be granted access to the device.

3.3 Using the Web

The Web acts as glue that helps the user see the control system from a single application, namely a Web browser. In addition to standard Java/CORBA clients of the human machine interface, which include a Java visualisation tool for AutoCAD files, standard Web applications will be used for the help system and for the logbook.

The help is simply a group of html files, with embedded PDF documents for technical drawings.

The logbook will be just another newsgroup, where only the operators can post but anybody within the laboratory can read. The current Web browser allow so conveniently to attach any electronic document type, including machine status printouts, that only a simple Excel template is necessary to simplify operator input. The machine status can be written to a file with a simple Java application and then attached, too. As the law requires keeping the logbook on hardcopy, the latest mailings to the newsgroup will be printed once per shift.

An information service page with a small Java client displaying the actual status of the light source, its energy, current and lifetime history will be available without security restrictions to everyone.

CONCLUSIONS

It is important that the concepts used will last over the lifetime of the ANKA project, which is long compared to a typical lifecycle in the computer business. It is risky to make predictions, but we believe that products that are based on the technologies described in this paper can experience a longer lifetime than usual.

Experiences with the running system for the ANKA microtron show that the concept lives to its promises. A successful port of the Java Beans to a test case at the Swiss Light Source, where the calls to CORBA were replaced with calls to CDEV, is a proof that the strict object oriented model of devices can be used to effectively hide other types of communication channels. This means that there is no limit to the reusability of the Java applets developed for ANKA at other accelerators.

REFERENCES

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

[2] M. Plesko, Implementing Distributed Controlled Objects with CORBA, PCaPAC99 workshop, Tsukuba, January 1999.

[3] G. Tkacik, Java Beans of Accelerator Devices for Rapid Application Development, PCaPAC99 workshop, Tsukuba, January 1999.