THE UPGRADE OF THE ANKA CONTROL SYSTEM TO ACS (ADVANCED CONTROL SYSTEM)

I. Kriznar, G. Pajor, M. Plesko, M. Sekoranja, G. Tkacik, D. Vitas (JSI and Cosylab),
K. Cerff, W. Mexner (ANKA-ISS)

Abstract

The entire CORBA communication layer at the synchrotron light source ANKA was efficiently upgraded to the next generation device servers, based on Advanced Control System (ACS). The old system, which was running since 2000, required an upgrade to a new version of commercial CORBA libraries. Instead of purchasing a costly license, a decision was made to upgrade to open source TAO CORBA and ACS. The design of ACS is the next step in the evolution of the idea, which lays behind the old ANKA CS, a vision of a control system with distributed CORBA objects for remotely controllable physical devices. The ACS, developed in cooperation with ESO, extended the old device servers with improved management and debugging capabilities and greatly improved the long time stability of servers on Windows NT machines. Adjustment to the new server interface was smooth and without major changes at the fieldbus and client side of the CS. This was possible due to modular object-oriented architecture at all control system layers and good design of interfaces for devices. The code of complex Java clients stayed practically the same, we only had to change the communication plug of Abeans, our core Java libraries. This has significantly reduced the upgrade time because all end-user Java applications remained visually and functionally unchanged. This is also of great value to the operators because no additional training was required. Interfaces to LonWorks device drivers, which are based on functionality of devices, also remained unchanged. In a system with a large number of various devices we were able to keep the whole LonWorks fieldbus intact, we even used the same configuration database with the same set of installed drivers. The smoothness of the upgrade confirms that our developments were focused into the right direction. Now even those users are 100% satisfied with the control system, who always had something to complain about.

INTRODUCTION

Control system for ANKA accelerator storage ring was designed and produced by KGB (Kontroll Gruppe fur Beschleuniger) group at Josef Stefan Institute in Ljubljana, Slovenia. Essentially the same group of people also successfully upgraded part of the control system to ACS in 2002.

ANKA is a 2.5 GeV synchrotron radiation light source located in Karlsruhe, Germany. It contains about 500 physical devices (power supplies, vacuum pumps, beam position monitors, RF generators, etc.) that are managed by the control system. The device I/O is handled by self-sufficient microcontroller boards, which connect to a standard LonWorks field bus network. Each branch of the network is attached to a PC with simple device servers running on it. The DeviceServers map the devices and their properties onto approximately 2000 objects and make them remotely available using CORBA (ACS - Advanced Control System). On the client side, Abeans completely wrap the CORBA client-side objects. Abeans provide a rich application framework which allows even non-experts to easily build powerful applications.

The main development objective of the control system for ANKA was to make it user(operator)-friendly and easy to maintain. ANKA was built with minimal cost; we had to minimize in-house development and maximize the use of commercial-quality products. 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. Note that only those components have been developed in-house, for which no commercial alternative existed. This category includes components, which are specific to accelerators, such as API, data server and the I/O boards.

Table 1: Products used in the control system

Usage	**Product**	**Source**
consoles	PCs	Many
operating system	Windows NT	Microsoft
panels	Java	Sun
accelerator API	Abeans	Cosylab
process computer	PCs	many
operating system	Windows NT	Microsoft
Internet communication	ACS	ESO & Cosylab
fieldbus management	LCA/LNS	Echelon
archive database	SQL	many
config. database	text-files	Cosylab
data server	ACS server	Cosylab
fieldbus	LonWorks	Echelon
microcontroller	Neuron	Echelon
programming language	Neuron C	Echelon
cross-developing tool	LonBuilder	Echelon
I/O boards	Zeus, Hera, Ariadne	Cosylab

OBJECT ORIENTED DESIGN WITH DEVICE/PROPERTY ARCHITECTURE

ANKA control system design is based on object-oriented (OO) technologies from its lowest layer up to the client software. The essence of this design lies in the description of physical devices in OO terms, for which Basic Control Interface (*BACI*) was used (the very first design was called Accelerator CORBA Interface (ACI) [1]). In accordance to CORBA specification, device descriptions were phrased in terms of Interface Definition Language (IDL), which presents a programming language-independent way of defining object interfaces. The BACI has been meant to be a standardized interface so that applications and pieces of control systems can be hooked to it from either side. The intent of BACI IDL definitions was to create standardized, functionally-oriented interfaces to which clients and servers hook from each side. In addition, we were hoping for wider acceptance of these definitions in the control system community, because they are independent of the underlying HW details of the control system.

A *device* is a CORBA object that corresponds to a physical device, e.g. power supply, vacuum pump, current monitor, etc. The device is the basic entity of the BACI, because it is the most natural concept for modelling physical entities in an accelerator. Actions that are executed on a device, like on, off or reset correspond to methods of the device. Each device has a number of *device properties* that are controlled, e.g. electric current, status, position, etc.

Device properties, which are also defined as objects in the BACI, are referenced by the devices that contain them as IDL attributes. Properties are distinguished by type (pattern = unsigned integer, double, etc.) and by being read-only (RO) or read-write (RW) objects. Each such property object has specific characteristics, e.g. the value, the minimum, its description, units, etc. The methods of the property interface allow the user to retrieve the characteristics and optionally (in case of RW properties) also set the value: `get()`, `set()`, `minVal()`, etc.

A well known example of a definition of device is IDL for a simple power supply:

```
interface PowerSupply : Device {
  // properties
  readonly attribute RWdouble current;
  readonly attribute ROdouble readback;
  readonly attribute ROpattern status;
  // commands
  void on(CBvoid);
  void off(CBvoid);
  void reset(CBvoid);
}
```

DEVICE SERVERS

Device servers were precursors to ACS servers. A *DeviceServer* is a CORBA sever written in C++ that implements one specific device IDL interface and exports devices with this type to the network. Device servers were running on Windows NT computers with LonWorks fieldbus interface card in each.

In ANKA, we control about 500 distinct physical devices through 29 (initially 26) different IDL device types. They were exported by 35 (41 at the moment) device servers on 9 Windows NT machines with one LonWorks fieldbus interface card per machine. Device servers were using a well-known CORBA implementation of a major vendor. After a number of problems with CORBA stability and purchase of license for libraries, the decision was finally taken to upgrade device servers to a different CORBA implementation. At that time, a stable release of ACS, built on TAO open source CORBA implementation, was already available. With an upgrade to ACS, we were able to move to new CORBA and whole new control system, which was created with experience accumulated at similar projects in the astronomical and particle accelerator communities. In addition, ACS is a professional-grade control system with far better support and documentation base than the original device servers.

ACS

ACS is a CORBA-based control system framework with all features expected of a modern control system ([4] and [5]). ACS provides a powerful XML-based configuration database, synchronous and asynchronous communication, configurable monitors and alarms that automatically reconnect after a server crash, run-time name/location resolution, archiving, error system and logging system. Furthermore, ACS has built-in management, which enables a centralized control over processes with commands such as start/stop/reload, send message, disconnect client, etc. and is fine-grained to the level of single devices. ACS comes with all necessary generic GUI applications and tools for management, display of logs and alarms and a generic object explorer, which discovers all CORBA objects, their attributes and commands at run-time and allows the user to invoke any command. ACS employs several standard CORBA services such as notification service, naming service, interface repository and implementation repository. It hides all details of the underlying mechanisms, which use many complex features of CORBA, queuing, asynchronous communication, thread pooling, life-cycle management, etc. Written in C++ and using the free ORB TAO, which is based on the operating system abstraction platform ACE, ACS has been ported to Windows, Linux, Solaris and VxWorks. ANKA port of ACS runs on Windows NT and Windows 2000.

ACS at ANKA

By drawing on our experience with old device server deployment, we utilized ACS framework to create a new ANKA CS installation very similar to the old one. New ACS based control system shares the same Device/Property

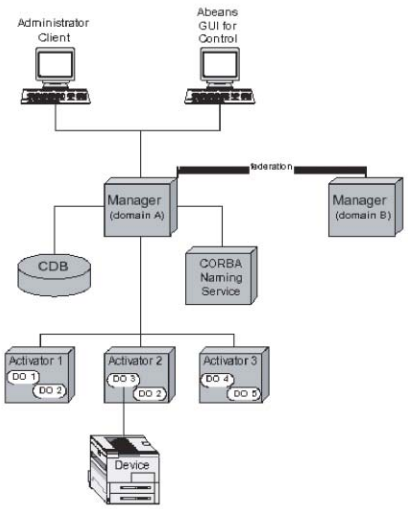

Figure 1: ACS deployment diagram.

data model (called Component/Property at ACS) as described by BACI Basic Control Interface (*BACI*). BACI IDL interfaces in ACS are generalization of the initial BACI (or ACI) definitions from ANKA control system. BACI defines a set of base classes and interfaces, which prescribe how Component (Device at ANKA) and Property implementators provide data flow from remote distributed objects. In effect, BACI itself does not define any specific control system, but is a formal design pattern specification of how its basic interfaces are to be combined to create a set of Device/Property IDLs. When BACI was incorporated into ACS, surprisingly very little of basic BACI definitions were changed. This proved how solid and well thought was OO design of ANKA control system.

Transition to ACS was performed very smoothly. It took one man-month of work to customize ACS servers for ANKA (replacing the functionality of DeviceServers) and two man-weeks of work squeezed in one week visit at ANKA for on-site debugging in order to deploy a stable ACS installation. Much of this success is owed to the fact that only one device server layer of existing control system was changed.

The following control system components were replaced or changed during transition to ACS:

DeviceServers	replaced with ACS containers (Activators) and distributed objects (DOs)
Abeans	minor changes in Plug due to BACI update
StaticDB	minor changes in configurational database, added additional ACS management related entries

THe following control system components remained intact during transition to ACS:

high-level Java client software	machine operators was able to operate the machine with a familiar application
LonWorks device drivers	key to smooth transition, a lot of work was saved by keeping already integrated devices in the control system as they are

FUTURE PLANS

The development of Abeans/ACS control system at ANAK currently proceeds in two directions. An idea emerged, add the ACS interface to beamline devices. Firstly, we are preparing an idea of how to model beamline devices with BACI and deploy them on ACS platform. Beamlines would benefit from ACS services and Java clients. Even greater benefit would be the possibility to write storage ring or beamline Java clients that would make automatic optimization based on data from both parts of the machine.

Secondly, the total number of computers will be reduced at least by half by putting three LonWorks fieldbus interface cards onto a new single computer. The nine PII server PCs will be replaced by three new Pentium IV PCs and one LonWorks ISA card per PC with three PCI standard cards. This will reduce the maintenance effort for the server side of the ANKA control system.

CONCLUSION

Device servers at ANKA were smoothly replaced with ACS activators in very short time. Transition with minimal effort was possible due to the fact, that only single layer of existing ANKA control system was totally replaced. Besides this ACS was designed on experience with old device servers deployment, so both share the same data model. Even though, the ACS was a generalization of initial ANKA control system design, the old set of IDL interfaces which prescribe remote distributed objects was changed surprisingly little. This proved how solid and well thought was the initial object-oriented design of ANKA control system.

REFERENCES

[1] M. Plesko, “The CORBA IDL Interface for Accelerator Control”, EPAC’98, Stokholm, June 1998

[2] M. Plesko et al, “A Control System Based on Web, Java, CORBA and Fieldbus Technologies”, PCaPAC’99, Tsukuba, January 1999

[3] I. Verstovsek, M. Plesko et al, “ANKA Control System Takes Control”, PCaPAC’00, Hamburg, October 2000

[4] M.Plesko et al, “ACS the Advanced Control System”, PCaPAC’02, Frascati, October 2002

[5] http://www.eso.org/ gchiozzi/AlmaAcs/

[6] http://www.cosylab.com/