MIGRATION FROM ACS 1.1 TO ACS 4 AT ANKA

I. Križnar, M. Pleško, M. Šekoranja, P. Kolarič, (JSI and Cosylab),
W. Mexner (ANKA-ISS)

ABSTRACT

The control system at ANKA, the 2.5 GeV synchrotron light source in Karlsruhe, has been designed and built with CORBA distributed remote objects. The whole control system was divided in three major layers: device drivers and fieldbus, CORBA device servers and finally Java GUI libraries. In the year 2002 the layer with device servers was successfully replaced with ACS version 1.1 (ANKA Advanced Control System, ICALEPCS'03). The smooth transition proved that the design of ANKA's control system with object-oriented architecture was well chosen. The ACS is CORBA based middleware developed in cooperation with ESO. The ACS framework is build on top of CORBA which hides the complexity of the CORBA middleware and other libraries. In addition ACS provides an implementation of a coherent set of design patterns and services that makes the whole control system software uniform and maintainable. ANKA was one of the first real-life installation of ACS. Since then ACS has experienced a fast and substantial evolution, which was driven by the requirements coming from users. In the year 2005 ANKA has decided to upgrade their ACS 1.1 installation to the latest stable ACS release. With this transition ANKA will benefit from the improved performance and stability and also the integration of new hardware into ANKA control system will be easier. Since ACS version 1.1 it has been introduced a new DevIO abstract interface which simplifies writing device drivers and servers. The upgrade work is in progress.

1 INTRODUCTION

The control system for the 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 is successfully maintaining and upgrading the control system ever since. At the moment ACS installation at ANKA is being upgraded to latest stable ACS version which is at the moment 4.1.3.

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.) which are managed by the control system. The device I/O is handled by self-sufficient microcontroller boards, which connect to a standard LonWorks fieldbus network. Each branch of the network is attached to a PC with simple device servers running on it. The device servers map the devices and their properties onto approximately 2000 objects and make them remotely available using ACS (CORBA). 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.

Usage	**Product**	**Source**
consoles	PCs	many
operating system	Windows 2000	Microsoft
panels	Java 1.4	Sun
accelerator API	Abeans 2	Cosylab
process computer	PCs	many
operating system	Windows 2000	Microsoft
Internet communication	ACS 1.1	ESO & Cosylab
fieldbus management	LCA/LNS	Echelon
archive database	SQL	Microsoft SQL Server
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 (analog I/O), Hera (digital I/O), Ariadne (serial I/O)	Cosylab

Table 1: Products used in the ANKA control system

2 OBJECT ORIENTED DESING OF DEVICES

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. 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);
}
```

3 ACS

ACS is a CORBA-based control system framework with all features expected of a modern control system ([3] 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. ACS components are written in C++ and Java. ACS has been ported to Windows, Linux, Solaris and VxWorks. ANKA port of ACS runs on Windows 2000 computers.

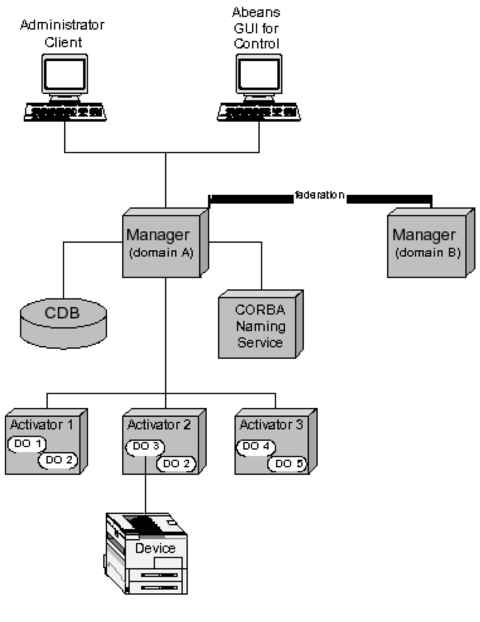

Figure 1: ACS deployment diagram.

3.1 CURRENT STATUS

ACS is now at half of development life and very much matured since version 1.1 which is used at ANKA. The core of ACS is now very stable in terms of design and implementation. ACS has been extensively used for development of Atacama Large Millimeter Array (ALMA), the project for which is being designed and built, and as well for 20 other development sites on four continents. ACS is available under LGPL license [5] to encourage wider community of users to provide feedback and help. ACS is besides ANKA and ALMA used as base system for several other projects: the Atacama Pathfinder Experiment, the Spanish OAN 40 meters radio telescope, the Sardinian Radio Telescope in Italy and the Hexapod Telescope in Chile. ACS provides common software framework, the Component/Container model allows different project to share Components and solutions [1].

3.2 ACS AT ANKA

Device servers at ANKA are written in C++ and runs in a Windows 2000 environment. A device server is essentially ACS component that implements one specific device IDL interface and exports devices with this type to the network. Device server computers are constrained to Windows operating system because the LonWorks fieldbus interface card is only supported on this platform.

At ANKA, we control about 500 distinct physical devices through 29 (initially 26) with different IDL device types. They were exported by 35 (41 at the moment) device servers on 9 Windows 2000 machines with one LonWorks fieldbus interface card per machine.

ANKA was one of first ACS installations and is successfully running on one of the first ACS versions (1.1) since 2002. ACS was very successfull from the beginning but version 1.1 still did not solve some issues with stability and performance. Since then ACS has steadily progressed with new features and the stability was improved. Major improvements since the version 1.1 were:

- Improved stability
- Manager and some of central services are now written in Java (jManager) and can run on any platform.
- Reorganized ACS support for components (DevIO interface) which makes writing new device server easier.
- Improved Error handling
- ACS device servers can be now completely written in Java (jBACI).

With the old ACS 1.1 system running quite well there were additional reasons needed for the decision to upgrade.

- ANKA want to move back to main ACS branch. It is more expensive to fix and maintain ACS 1.0 than to use the latest ACS, because ACS 1.1 version is supported by ANKA alone. Any work done on ACS 1.0 at ANKA will not be included in next ACS releases thus makes later upgrade more problematic.
- Once ANKA has joined the main ACS branch, it will be easier to stay up to date with the latest improvements.
- ANKA needs to be prepared to replace the LonWorks fieldbus. LonWorks communication boards are now relatively old and the production costs for replacing parts are increasing.
- The new ACS device server support is more flexible. It makes possible to integrate new devices to the control system without using the LonWorks fieldbus. With an intelligent integration platform, such as microIOC, there will be even no need for any fieldbus at all.

4 THE UPGRADE

The ACS device servers needs to communicate with LonWorks fieldbus PC cards, which are supported only for the Windows platform. Unfortunately our search for Linux drivers for LonWorks has given no usable results. For this reason the new ACS device servers have to to run under Windows. Because the latest C++ ACS libraries are not supported on Windows decision was made to take the Java ACS environment and to connect to the C++ native LonWorks driver with JNI (Java Native Interface) technology.

The JNI framework allows Java device server to call C++ native methods. The native C++ part of device server will be a transparent wrapper around the LonWorks communication library. Calls from Java will be transfered to the LonWorks communication layer. A native device server will be able to update the Java device server code trough the JNI interface, thus transport callbacks and events to the Java device driver are possible. The native part will become very simple and thus more robust and reliable. All complexity, multithreading and handling CORBA communication will be done in Java code. Since Java has proven to be very suitable for programming server applications this will further improve stability and performance. Calling methods through JNI interface brings some cost in performance. To minimize this costs of communication between native and Java code the code will be simplified and minimized. Java device server will talk to a native device one-to-one. The serving of many clients and dispatching values will be done complete in Java.

The upgrade will be performed in two steps to make sure all initial problems will be smoothed out before new system will be ready to take over. In a first step only one device driver for power supplies will be ported. This device server is one of most complex devices and is heavily used in daily operations. When the power supply device server have been well tested and accepted as mature, in a second step all other 29 device servers will be ported and tested during a 2 weeks shutdown.

5 CONCLUSION

ANKA will upgrade ACS 1.1 control system to the more stable ACS 4 release. Even if these one of first ACS releases is reasonably adequate, there are strong reasons to upgrade the ACS installation at ANKA to the latest version. When the special port of ACS at ANKA has been replaced with the head version, it will be easier to follow latest improvements if required. At the moment the upgrade is progressing with first step and the first version of new system will be tested during ICALEPCS conference.

References

[1] G. Chiozzi et al, “The ALMA Common Software, ACS Status and Development.”, ICALEPCS’05, Geneva, October 2005

[2] I. Križnar, M. Pleško, W. Mexner et al, “The Upgrade of the ANKA Control System to ACS”, ICALEPCS’03, Gyeongju, October 2003

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

[4] I. Verstovšek, M. Pleško et al, “ANKA Control System Takes Control”, PCaPAC’00, Hamburg, October 2000

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

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