# INTEGRATION OF ABEANS AND ACS IN THE GSI CONTROLS ENVIRONMENT

K. Höppner, G. Fröhlich, L. Hechler, U. Krause*, V. RW Schaa, GSI, Darmstadt, Germany
I. Kriznar, M. Plesko, A. Pucelj, M. Sekoranja, I. Verstovsek, Cosylab, Ljubljana, Slovenia

*Abstract*

GSI will upgrade the control system in the near future and is therefore examining new technologies, such as the possibility to use CORBA middleware and Java on the client level. GSI has implemented UFC, the interface to the device server level, primarily on the OpenVMS operating system. For other platforms, a subset of UFC is available by a thin server that runs on an OpenVMS machine. Based on the Java interface to this server a proof-of-principle integration of Abeans and UFC libraries was demonstrated – the UFC Abeans plug. Abeans is a library that provides simple Java beans for connection with the control system. At the same time, it provides several useful services: logging, exception handling, configuration and data resource loaders, authentication, and policy management. In this article we will describe the details of the upgrade to Abeans and discuss possible further steps, including migration to a CORBA based middle layer, the Advanced Control System (ACS). ACS is a platform independent control system nucleus for writing servers based on an object-oriented model of distributed devices. Like Abeans, ACS hides all details of the underlying mechanisms, which use many complex features of CORBA, queuing, asynchronous communication, thread pooling, life-cycle management, etc.

## INTRODUCTION

The GSI accelerators are operated by a control system using VME computers on the device server level and OpenVMS workstations for the operator interface. Both layers communicate by a proprietary in-house protocol which is, on the client side, implemented for OpenVMS only. In addition, there is also one gateway server which translates UFC protocol to TCP/IP based communication.

Devices are modeled in a unique way. This allows access to all devices by one common narrow interface.. Access to the device server level is by the name of the device and the name of the property to be executed. In the GSI control system about 65 different types of devices are supported. Each of it implements, besides several common ones, about 40 individual properties. A total of 2870 properties are implemented in the GSI system.

After 15 years of operation, GSI is in the process of modernizing its control system in hard and software [1]. Renovation includes the VME boards of the device server level too. The boards are no longer supported by the manufacturer and are not produced on the market any more. Replacement will be by a Linux based system (e.g. PowerPC). Switching to a powerful hardware, in combination with a comfortable operating system, allows upgrading from the in-house network protocol to CORBA communication.

*U.Krause@gsi.de

## ABEANS ON UFC

Abeans is the next generation of Cosylab's Java based client framework for building control system applications. It has been ported, plugged to such different CS as those of DESY (TINE), SNS (EPICS) and GSI. Abeans consist of two parts, one are the connection and service beans and the other are GUI beans, called CosyBeans. Connection and service beans provide application services and the mechanisms that allow simple implementation of data flow between the local application and the remote control system. This task is realized in two layers. Firstly, Abeans define a model that is a layer of Java Beans components that represent controlled objects. Secondly, Abeans define a plug which is a driver layer, specific to a given model and an underlying communication system. In addition, Abeans provide several useful services: logging, exception handling, configuration and data resource loaders, authentication, and policy management. CosyBeans provide clear and consistent visualization of dynamic data with standardized presentation of alarms, monitors and connection status [2].

In November 2002, a proof of principle Abeans plug was created. The plug connected Abeans to the UFC layer. Abeans plug was connected to the gateway server via the TCP/IP protocol. At the moment, the plug only provides for read-only functionality. The next step is to extend the implementation of the UFC Abeans plug to provide all the functionality offered by the gateway server.

## GOING TO CORBA

The process of upgrading UFC has now gone far enough that it is possible to go to the next step: For every device, a CORBA-object on the device server level will be created. This article discusses a possible approach by using ACS – a CORBA based system developed by Cosylab and ESO [3], currently used in ANKA light source and for the ALMA project. In the next section this approach is presented on a specific example of a GSI power supply.

## OO VIEW OF A POWER SUPPLY

To illustrate the OO concepts in more detail, a GSI power supply for magnets was taken for prototyping as a representative example of a current GSI device.

One of the major principles of object oriented (OO) design is abstraction: Each object can be described as a set of attributes (properties and characteristics) and actions.

A more detailed analysis of object's attributes reveals that attributes can be divided into three groups:

- **static attributes** - attributes that do not change, e.g. minimum and maximum of the power supply's current (MIN/MAX CURRENT),
- **attributes which change very rarely**, e.g. version of the power supply (VERSION),
- **dynamic attributes** – e.g. the current of a power supply (CURRENT).

Besides attributes, objects also have **actions** that change object's states, e.g. ON, OFF, RESET, INIT, ABORT.

The existing integer property POWER was mapped to two actions, ON and OFF. Having POWER property and writing 0 and 1 does not follow OO design which should be as close as possible to the natural perception of the world – power supply magnet can simply be turned on and off.

In addition, the execution of some actions can take quite a long time and this will make remote call to block which is often not the desired behavior. The problem can be solved using asynchronous communication (using Asynchronous Completion Token design pattern – an object behavioral design pattern for efficient asynchronous event handling). In this way, remote call is not blocked and the caller is notified about the completion of an action using a callback.

**Properties.** All properties (dynamic attributes) are mapped to objects, so a component, i.e. modeled device, is a container for property objects.

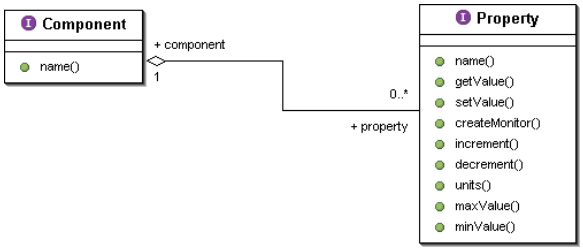


Figure 1: Component - property diagram.

Properties itself contain characteristics, actions and other methods. Properties are strongly typed, e.g. `double` / `long` / `string` property, which enables compile-time checks and thus prevents errors.

This approach is very powerful. For example, given a property object, e.g. current, you basically know everything about the property – obtaining information about the property and its manipulation is very trivial and simple.

**Characteristics.** A characteristic is a data item that is considered static. It is represented by a name - value pair, where value parameter is typed. A characteristic is accessed through a single accessor method. Characteristics are contained both by components and by properties.

A remote call that accesses a characteristic is synchronous, i.e. it blocks until the value is returned to the caller, because it is assumed that static values will be stored in a quickly accessible repository, e.g. in a static configuration database.

MODEL_NAME or SERIAL_NUMBER of a power supply magnet can be considered as component characteristics and maxValue, minValue, units, resolution, alarmHighOn, alarmHighOff, etc. as property characteristics.

Once the OO abstraction is completed, the mapping can be written in the Interface Definition Language (IDL), a meta language used to describe objects in CORBA. The next step is too looking at how to deploy these objects and how to provide all the services required by the control system, e.g. configuration database, naming service, logging, authentication, etc.

## PUTTING IT ALL TOGETHER

In abstract terms, a control system can be viewed as a collection of services that enable the interaction between controlled entities (components) and the clients. Containers provide the environment for components to run in, with support for basic services like logging system, configuration database, persistency and security. ACS provides all the implementations of these services.

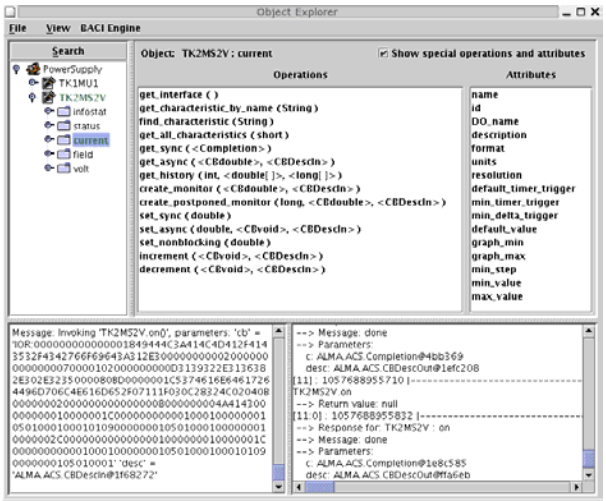


Figure 2: Object explorer, a generic ACS application can be used to display and manipulate a GSI power supply.

ACS has a so called Manager, which is set up at one central location that is known across the entire system. The Manager is acquainted with all the components and containers in the system, as well as other resources, such as configuration database and other services. It is a client's entry point into the system and manages security and components lifecycle (instructing containers to activate / deactivate components).

On top of the ACS device server layer run Abeans through the Abeans plug for ACS, used at ANKA and ESO. Figure 2 shows how a generic ACS application, the

Object Explorer, can be used to control a GSI power supply, the device chosen for prototyping.

## FURTHER STEPS

The existing GSI control system cannot be skipped in short time, to make a smooth transition, the GSI control system and ACS have to be supported in parallel, both on the device server and the client application level – UFC and ACS applications must know how to connect both to new and to old devices.

This section summarizes the discussions about possible further steps. The new control system will be CORBA based object oriented. Two possible approaches based on ACS seem feasible, the *gateway* and *transitional IDL solution,* both of them will be described in brief. It is beyond the scope of this article to decide for any of the solutions.

**Gateway solution.** According to this scenario, the existing UFC TCP/IP server is enhanced to work as a translation server. UFC devices are exported as ACS servers and at the same the new devices are put on the UFC. Neither new nor old applications have to be changed in any way to communicate to all devices, regardless of the implementation.

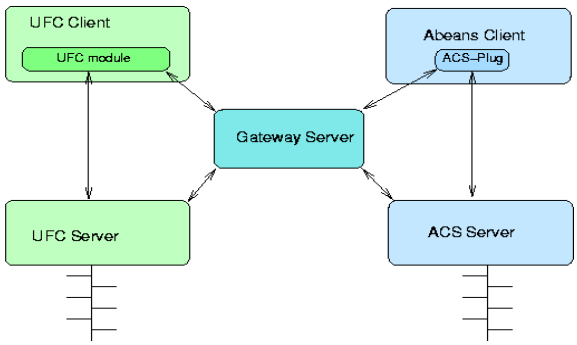


Figure 3: Transition solution with a gateway computer.

Benefits: Old and new system do not have direct knowledge of each other, therefore no recompilation of any part of old system is needed. Yet all devices are available everywhere. ACS management tools and services (Object Explorer, Administration Client, logging and Logging Client) could be used on all devices.

Drawbacks: A single gateway computer is a potential performance bottleneck therefore stress testing has to be performed. No automatic mapping will be possible in general, the majority of the 2870 implemented properties has to be handled individually.

Supporting a set of nearly 3000 properties in centralized gateways can be avoided by integrating the mapping into the device servers. A narrow interface on the device objects which provides access by the old property names will allow a direct mapping of the old UFC interface to the new CORBA-UFC communication. The new interface can be implemented with limited effort since the majority of the device specific software on the actual VME boards can be re-used. During modernization of the GSI control system all devices will be equipped with the proposed CORBA-UFC interface. This can be the starting point of another approach:

**Transitional IDL.** New Java clients use the Abeans interface to the control system and implement two types of connection: CORBA-UFC plug for old devices and ACS plug for new devices. To access newly installed ACS devices the CORBA-UFC interface has to be installed additionally. This can be done step by step, whenever a new ACS-device type is added to the GSI control environment.

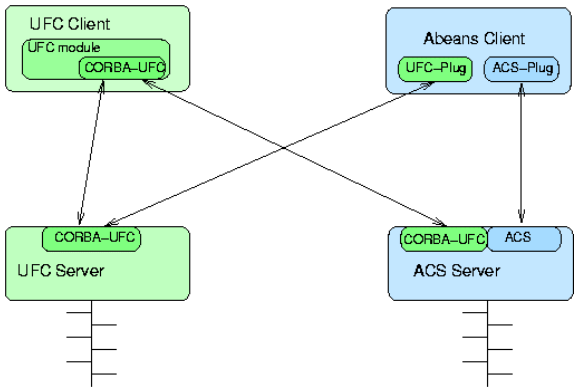


Figure 4: Transition solution with transitional IDL.

Benefits: No connection overhead, clients and servers are connected directly. Additional functionality of the UFC plug in Abeans can be implemented quickly.

Drawbacks: Old clients have to be at least partially recompiled. UFC protocol has to be implemented and maintained in many places: in the new ACS device server and in the Java clients.

## CONCLUSION

In the article we have discussed a possible approach to the upgrade of the GSI control system, the main emphasis was on how to represent the existing devices in a OO way. In addition, some prototype solutions on an example of a GSI power supply are already implemented. The next step is to focus on one of the possible approaches outlined in the previous paragraph and start porting the entire CS to CORBA.

## REFERENCES

[1] U. Krause, V. Schaa, L. Hechler, Re-engineering of the GSI control system, ICALEPCS 2001, San Jose, California, USA, November 2001.

[2] Igor Verstovsek et al, Abeans: Application Development Framework for Java, ICALEPCS 2003, Gyeongju, Korea, October 2003.

[3] Gianluca Gchiozzi et al, The ALMA Common Software (ACS): Status and Developments, ICALEPCS 2003, Gyeongju, Korea, October 2003.