INTEGRATION OF RENOVATED NETWORKING MIDDLEWARE INTO A RUNNING CONTROL SYSTEM ENVIRONMENT

U. Krause, L. Hechler, K. Herlo, K. Höppner, P. Kainberger, S. Matthies, G. Schwarz,
GSI, Darmstadt, Germany.

Abstract

Currently the proprietary networking middleware in the GSI control system is replaced by a CORBA based re-design. Rebuilding all controls components is out of scope, so existing applications as well as device specific front-end software still have to be used. The renovated middle layer has to fit between the former application's device access interface and the front-end framework. Providing similar functionality as before was a major design aspect therefore. Compatibility requested some extensions to the original approach, while the new outline, targeting more flexibility and clarity, lead to additional capabilities for future usage of the system.

INTRODUCTION

Accelerator installations are operated for long time, which may span decades. During this period, progress in hardware and software platforms for the control system is evident. Basic components some day are no longer available or are no longer sufficient for the growing demands. From time to time a new generation of controls platforms have to be introduced. While the primary task then is to assure the established functionality, adaptation to a more recent environment gives the change to upgrade the system significantly and to enhance its functionality.

CONTROLS RENOVATION

Starting Point

The GSI control system originates from the mid 80s. It uses VME computers for the front-end equipment control and OpenVMS workstations on the application level.

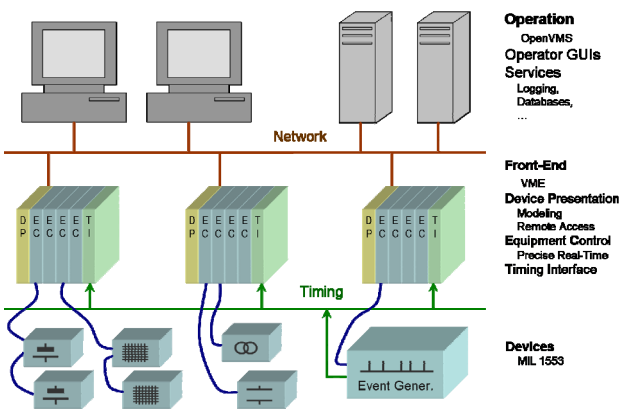

Figure 1: GSI Control System

Communication is by an in-house network protocol, implemented on raw Ethernet packets. A characteristic of the GSI system is to split the front-end layer: An equipment control layer, providing real-time reaction with short delay times, and a non real-time presentation layer to model all equipment in a unique manor and to handle the remote access. Both are implemented on different boards which operate tightly coupled.

Not unusual at the time when it was designed, the components are closely related to specifics of the hard- and software platforms. Even worse, multiple interconnections lead to confusing dependencies between front-end and application layer. Modifications in one area often request adaptation of modules all over the system. Substantial upgrades of the core were avoided therefore.

As a result, the system depended on the platforms originally chosen. Only special 68k VME boards on the front-end and OpenVMS on the application level are supported. The device presentation boards are no longer available for years and have to be replaced urgently. Additionally, the limitation to VME and OpenVMS is a severe handicap for integration of new controls demands. A more open structure is badly needed.

Strategy

A project was started to replace the outdated VME boards. However, instead of simply porting the existing structure to a new type of hardware, a more substantial renovation was aimed for: Opening of the rigid structures.

Structural modifications must not disturb handling of the machines. Since commissioning, accelerator operation was more and more refined, now regularly serving, on a pulse to pulse base, typically five experiments in parallel from three simultaneously operated ion sources.

Controls software, as well on application level as on the front-ends, was extended continuously to achieve such flexible operation. Re-implementing this functionality from scratch is out of scope – the effort would be far too high. Prerequisite in the renovation therefore is to keep the existing specific software.

The outline of the GSI control system shows a way to do so. Accelerator equipment is modelled, on the front-ends, in the nowadays well established object oriented view as devices with properties. Remote access from the operation layer software is via a common call interface. On the front-ends, each property is implemented as a separate function, called **U**ser **S**ervice **R**outine (USR). USRs have a common interface to the system core. The USRs, together with the software on the equipment control layer, implement the specifics of device equipment.

Basic idea of the renovation is to keep the remote access interface on one end and the existing USRs, and the equipment control boards with their software on the other side. Within these boundaries, in a Gordian knot approach, the software was rebuilt completely.

For the renovated system, PowerPC VME boards running Linux were chosen. CORBA as well established middleware was preferred to in-house developments.

Outline of the Renovated System

In the new front-end structure, the devices are implemented as C++ objects. Properties too are modelled by objects which are attached to the devices. With some adaptations the existing USRs are integrated in the properties as their execute method.

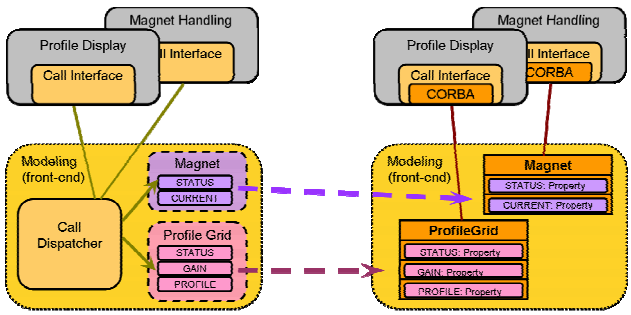

Figure 2: Migration to new Controls Outline

Management of the device objects, like instantiation and monitoring, is done by a new device manager service.

The CORBA based remote access supports synchronous as well as asynchronous commands and subscriptions. Access is by a new object oriented client interface. It is available for Java, and on Linux and Windows for C++ and Python. On OpenVMS, the former procedural call interface is still available, implemented as a wrapper on top of the new one.

Status of Renovation

The renovated system was introduced in the facility several months ago. Installed now in 6 out of a total of 43 VME crates, it showed its suitability during regular machine operation. No modification of the application's software was needed; re-linking with the new remote access library was sufficient.

Step by step, after adapting the USRs for each of the remaining 50 equipment types, the new system will replace the old system in all front-ends.

BENEFITS OF RENOVATION

Primary focus for the renovation was, as a must, to provide the established functionality of the former system. Using nowadays techniques for development led to several enhancements, compared to the original implementation.

Further improvements were addressed explicitly when limitations of the old system showed up clearly during the project. Opportunity was taken to smooth and clean the system. Additionally, some specifics of the existing system were not considered sufficiently when work started, and had to be handled additionally.

Both increased the originally estimated effort significantly, but on the other hand provided a more solid implementation and additional options for the future.

Portability

The manifold interconnections between front-end and application layer have been the major obstacle to advancements of the control system core. In the renovated system the new modules are far more decoupled.

This was seen when the system, developed for the VME environment, depending on functionality of the real-time equipment control layer, was adapted to standard desktop computers to have a more comfortable testing environment. With some shortcuts, a demonstration could be made operational in short time.

Encouraged by this, a solid implementation was build by re-arranging modules and adding functionality which is, in the VME environment, provided by the real-time layer. This stand-alone system then could be ported to the Windows operating system too in short time. While Linux is the preferred system for the controls, at least the Windows clients interface is requested in some locations.

While the GSI system for long time was limited to VME-Systems only, it is now well prepared for other platforms. A first usage of the new capability was to integrate Cosylab's MBox stepping motor driver in the controls environment [1].

Encapsulated Networking

Good modularization is essential for flexibility. Care should be taken that networking does not introduce unintended coupling between front-end and operation's applications. Therefore the communication modules are encapsulated. Both applications and device specific front-end software are clearly separated from any networking implementation, no CORBA specifics are seen.

Such encapsulation will allow modifying the networking without affecting the applications. Switching to another protocol, like SOAP, can be handled internally. Even several protocols in parallel can be supported then, an option for future extensions of the system.

Access Control

After thoughtless access to real accelerator devices an access control system had to be added to the originally open GSI control system. Users may access devices only when specific rights are granted.

The access control was implemented in the applications access interface and was based on internals of the remote communications. With the renovated middle layers, a new implementation was needed. In the new system access control is handled in the front-end devices. Rights information is provided by a key mechanism [2].

The new implementation provides several rights levels, according to the criticality of properties. Handling is simplified by declaring groups of users, equipment types and single devices. Rights can be defined for such groups now instead of administrating each user and each equipment type individually.

Data Container

The former procedural remote access interface originally supported single data or arrays of base

numerical integer and floating point types. Different byte orders and float representations on client and server side are converted automatically.

The need to combine integers and floats in one access led to introduction of a raw data format in which the bitwise data representation is exchanged. Conversion of data representation had to be left to the applications – a very error prone procedure.

The new implementation uses general data container objects. They may hold any sequence of the basic data types. Type information is stored additionally, which allows type-save extraction at the receiver side. Automatic conversion to any of the supported data types can be requested when data are extracted

Code Generator

Properties often exchange not only single values, but multiple data. In many cases these data are no arrays but structures, in which each element has its special meaning. For clarity descriptive names then should be used. So the wish came up to use names to access the data container elements instead of array indices only.

A formal XML description of the properties and their data was developed. From this description code frames and adapter classes are generated.

The code frames provide the connection to the property implementations. All formal code, needed to integrate the specific code into the new general property objects is generated automatically.

Adapter classes handle property data. Type safe setter and getter methods for the data elements of the container are generated. The method's names correspond to the names of the data elements, given in the formal description. This allows handling a data element only by its name, not bothering about its absolute position in the data container.

An additional benefit is consistent documentation. A textual description of property and data, to be used by the application developers, is generated automatically. Separately written descriptions, always likely to be outdated, will no longer be needed.

Subscription Service

The existing system allowed grouping of commands for different devices, if connected on the same VME node, in one remote access call. All such single requests are executed in the front-ends one after the other as one block. This ensures a kind of data synchronization.

In the new approach, device objects are independent. Each device executes requests separately, loosing any synchronization. This takes effect especially for subscription to event triggered execution. While synchronization is assured by the timing system, start of subscription now may happen in different cycles, resulting in data shift by one cycle.

Data synchronization by grouping commands used only a side effect in the existing control system, barely fulfilling the real needs. A since long requested more profound solution could no longer be postponed.

A subscription service, as an add-on on the client interface, correlates data from any set of devices in one of the timing areas. It groups, for different devices, responses from event subscription according to their system wide cycle identifier, distributed by the timing generator. Missing responses are identified and the requesting process is informed.

While the original correlation by combined commands was restricted to devices connected to the same VME crate, the new subscription service can handle any combination of devices in one timing area. This will simplify applications development, which is faced with more demands for data correlation, like for true transmission measurement following single bunches on their way through the beam lines.

CONCLUSION

Control systems operate in a moving environment. Underlying soft- and hardware platforms develop constantly, new technologies rise, and well established components some day become obsolete. New equipment is installed in the accelerators, more complex operation modes have to be supported, and more advanced machine control and better diagnostics is requested.

Reaction to the growing needs became more and more difficult in the GSI control system. The huge base of specific hand- and software installations however forces to stay with the system: Effort for re-implementing the specifics again in another system would be too high.

The communication layer renovation opened a way out of the dilemma. The solid general architecture allowed to cut out a severe hindrance to modernization, and to replace it by a more flexible state of the art element. Even more, not only preserving the established features, new functionality and more options could be added. Step by step exchange in short shutdown periods, keeping the majority of the controls unaffected at each single step, holds risk for machine operation low.

The renovated communication shows that evolutionary modernization of a control system, developed long time ago, is possible. Migration steps don't need to be small, whole core layers may be exchanged without requesting long interruptions in accelerator operation.

With the new system further upgrades like migrating operation's applications to Linux can start. The new core now opens ways of interoperation with other systems, preparing the integration of the existing accelerators as injector into GSI's planned FAIR facility.

REFERENCES

[1] K. Herlo, "Stepping Motor Control for Septum Plate Positioning", this workshop

[2] S. Matthies et al, "Access Control in the Renovated GSI Control System: A Combined Name- and Rights-Server", this workshop