Control of Distributed Data Acquisition Systems Using Object Oriented Methods

G. Kemmerling, M. Korten
Institut für Plasmaphysik, Forschungszentrum Jülich
EURATOM Association, 52425 Jülich, Germany
M. Drochner, P. Wüstner, K. Zwoll
Zentrallabor für Elektronik, Forschungszentrum Jülich, 52425 Jülich, Germany

Abstract

EMS (Experiment Message Specification) is a flexible software package, designed to build scalable distributed data acquisition systems for experiments at the Cooler Synchrotron (COSY) at the Forschungszentrum Jülich. It uses an object oriented hardware mapping for control and monitoring, which is logically based on MMS and FMS standards. Access from the experiment workstation is realized through TCP/IP socket communication, managed by a dedicated process on the workstation. The system is now 4 years in operation and an adaption to PC technologies is currently under development. Within the context of this upgrade, also the special communication process should be replaced by a standardized tool. The object oriented Common Object Request Broker Architecture (CORBA) is seen to address the problem in an adequate way.

1 Introduction

The Cooler Synchrotron COSY at the Forschungszentrum Jülich is a proton accelerator and storage ring, designed to investigate in medium energy nuclear and particle physics (40 to 2500 MeV). There are several experiments, located at internal beam and external target positions. Depending on the physical program, various hardware configurations are used and eventrates up to some thousand events per second are achieved.

In order to cope with these requirements, a flexible distributed architecture was choosen for the development of the data acquisition (DAQ) [1]. The system consists of a scalable amount of digitizing crates in CAMAC, FASTBUS and VME standards, each equipped with its own intelligent Readout Controller (ROC), and an Eventbuilder (EB) crate in VME with several processors to combine and store experiment data as an event on tape. All intelligent devices use Motorola processors of the 680xx family, running OS-9 as operating system. While the transfer of experiment data is established by VICbus interconnections between front-end and event-building controllers, control and monitoring of the system can be performed from an experiment workstation via an Ethernet-based LAN.

The system software is logically based on MMS (ISO 9506) [2] and FMS (DIN 19245) [3], powerful application layer protocols for distributed automation systems in industry. Because it is related to physical experiments, it is called EMS (Experiment Message Specification). EMS provides an object oriented client/server interface to the DAQ. All front-end devices and their functionality are mapped to abstract objects, which offer a set of services for control and configuration. The communication between clients and servers is managed by a dedicated process on the workstation and is based on TCP/IP sockets, using a proprietary protocol for exchange of data.

The system is now 4 years in operation at various physics experiments, but the increasing amount of data asks for a major revision [4]. Because of the very limited market size of readout controllers, especially for CAMAC and FASTBUS, and their long CPU lifetimes, a migration of the system to PC-technologies is currently under development. Instead of expensive embedded intelligent ROCs and EBs, transparent controllers with a connection to PCI/PC based CPU boards will be used. This approach allows on a very economical way to comply with the new demands by following the rapidly improving CPU performance on the PC market. As operating system for the PC-boards, the freely available NetBSD distribution was choosen, providing a homogeneous UNIX environment in the whole DAQ system.

Within the context of this upgrade, the system should also be opened for a standardized access method. By retaining the large investments in software development of the EMS interface, the communication process on the workstation will be replaced by the object oriented Common Object Request Broker Architecture (CORBA) [5]. Thus, the implied particular platform and protocol issues and therefore higher manpower requirements can be avoided.

The following sections describe the advantages of EMS and CORBA and the benefits for a combination of both systems. Work is influenced by another project at the Forschungszentrum Jülich, where the integration of a DAQ-system with CORBA has already been shown promising results [6].

2 Experiment Message Specification (EMS)

The central object in EMS is the Virtual Experiment Device (VED). It represents a ROC or an EB and acts as server for clients on the experiment workstation. A VED models the externally visible behaviour of the server and offers the resources and functionality associated with a real device for control, monitoring and event data transfer. In such a way, the client process on the workstation communicates not directly with different application programs in the distributed data acquisition system, but always with VEDs, providing a uniform view and access, independent of the type of the specific server.

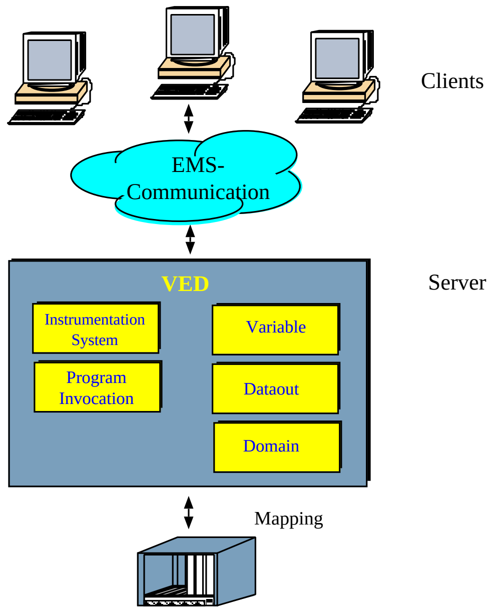

Figure. 1. Object oriented hardware mapping in EMS

As shown in fig. 1, a VED contains a set of generic objects, that help to coordinate the access to the functionality represented by the specific object. A short description of these objects is given below:

- **Variable**, either a single integer or an array of integers in VED's memory,
- **Domain**, a loadable image consisting of data,
- **Program Invocation**, an executable program related to one or more domains,
- **Instrumentation System**, the main building block for the server, a complete functional unit with a logical or physical meaning, consisting of one or a group of hardware modules, a set of procedures e.g. for event readout, initialisation and set-up including necessary lists,
- **Data Output**, dedicated to a sequential data recording device, a buffer or a communication link.

Whereas the Domain, Variable and Program Invocation are well known objects of MMS and FMS, the Instrumentation System and the Data Output are dedicated to the experiment's needs. The definition of the instrumentation system reflects the fact that an experiment should normally offer a higher flexibility to the user than a manufacturing device. The VED corresponds to the Virtual Manufacturing Device (VMD) in MMS.

The servers consist of the EMS-interface and hardware dependent code. The connecting link are local procedures, which must be programmed according to the application needs. A complete program is executed by a sequential interpretation of a list of local procedure calls, downloaded from a workstation client. A global list of the hardware modules and their types is stored in corresponding data structures.

Client programs on a workstation are developed by the use of C and C++ libraries, which allows access to EMS-services via the communication process. The communication process can manage several client/server connections at a time. In particular, it has the following tasks:

- queueing of requests from several clients for one server, so that the server only works on one request at a time,
- assignment of logical names to server addresses,
- handling of fault conditions, debugging and logging - some transport protocols (e.g. OS-9/net) are not able to recognize unexpected breaks of connections,
- management of asynchronous messages (so called *unsolicited messages*.

There are currently several client applications used for experiment control, based on X-Motif, InterViews or Tcl/Tk. For smaller applications and test systems, an interactive script interface is also available.

3 Upgrade of communication architecture

The key boundary condition for the migration to PC-technologies was to gain improvements with preserving as much as possible the existing instrumentation and software developments. However, the proprietary communication process on the workstation, which was originally introduced to handle OS-9 shortcomings, will be replaced by a standardized and platform independent software tool, that is able to tie the client/software applications together. The CORBA architecture was selected for this purpose.

CORBA is an specification of a object-oriented architecture for distributed applications, which is defined by the Object Management Group (OMG) [7]. It combines the benefits of object-orientation and distributed computing and allows clients to invoke methods on server objects transparently, without knowing the object's location or the programming language used for the implementation of the object.

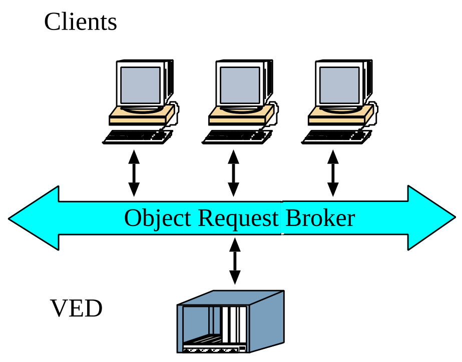

Figure. 2. Communication between clients and server, managed by the Object Request Broker

The core of the whole architecture is the Object Request Broker (ORB). The ORB is responsible for the delivery of client-requests from one computer to objects possibly on another computer (e.g. a front-end controller) as well as for the return of resulting information (see figure 2). By hiding all details of data transmission from the client, it facilitates the communication between clients and server. Therefore, the user can concentrate on the implementation of the objects services and their usage in client programs, avoiding all pecularities of communication. To target an object, the client only has to specify the object reference, which is created when the object is created.

By usage of CORBA, the EMS-objects and data structures are described by interfaces in the standardized Interface Definition Language (IDL), independent of any programming language. The mapping of such interfaces to client/server source code of a specific programming language is performed by the IDL-compiler of the CORBA implementation. There are many CORBA implementations on the market, available on almost all popular computer platforms. As language mappings of IDL, vendors usually provide C++, C or Java. Thus, the introduction of CORBA not only extends EMS to platforms and operating systems not supported so far, but gives the user also the possibility to choose between some programming languages for the client implementation. The usage of Java, for instance, has the advantage of building client applications for EMS, which directly can be inserted in Web-browsers. Furthermore, with the IDL-definition of EMS a strict division of client and server side is achieved, resulting in a better maintenance of software. To guarantee the interoperability between ORB's from different vendors, CORBA specifies the Internet Inter-ORB Protocol (IIOP). IIOP defines the transfer syntax and a set of message formats for ORB interoperation just as an IDL interface defines the protocol between an object and its clients.

4 Conclusion

In order to validate the new concept for communication, based on the choice of CORBA, a pilot implementation is currently under development. It is to test the techniques of integrating EMS and will have reduced but basic functionality of a VED. In a later stage this interface will be enlarged to enable full control and set-up from the experiment-workstation, whereas for the transfer of experiment data high bandwidth connections between servers are further on used.

Most implementations of CORBA do not consider security mechanisms. Thus, servers are usually open to general access without additional authorization checks, which is not acceptable in experiment control. An additional server, a socalled 'System' server, has been introduced for this purpose. It manages users access to the system and keeps track of their individual authorities. Furthermore, it provides the necessary adresses of VEDs in the system.

As CORBA implementation, the freely available omniORB-distribution [8] is used. omniORB is a multithreaded Object Request Broker, which is compliant to the IIOP protocol. It supports a simple C++ mapping of IDL interfaces, which allowed us to implement first versions of servers in a rather short time.

References

[1] K.Zwoll, M.Drochner, W.Erven, J.Holzer, H.Kopp, H.-W.Loevenich, P.Wüstner, K.H.Watzlawik, N.Brummund, M.Karnadi, R.Nellen, J.Stock, S.Dienel, K.H.Leege, W.Oehme, "Flexible Data Acquisition System for Experiments at COSY", IEEE Trans.Nucl.Sci., Vol 41, No. 1, pp 37-44, 1994

[2] ISO9506 - Manufacturing Message Specification, International Organisation for Standardization

[3] Profibus Application Layer (FMS, FM7, and LLI), DIN19245

[4] M.Drochner, W.Erven, P.Wüstner, K.Zwoll, "The Second Generation of DAQ-Systems at COSY", IEEE Real Time Conference 97, Beaune, France, September 1997

[5] CORBA 2.0/IIOP Specification, document 97-02-25, http://www.omg.org/corba/corbiiop.htm

[6] G.Kemmerling, M.Korten, H.Kleines, K.Zwoll, C.Balke, M.Ephraïm, M.Koopmans, C.T.A.M.de Laat, W.Lourens, E.van der Meer, J.Venema, W.Kooijman, A.A.M.Oomens, F.Wijnoltz, "Remote Experiment Monitoring and Control at the TEXTOR 94 experiment", IEEE Real Time Conference 97, Beaune, France, September 1997

[7] homepage, http://www.omg.org

[8] ORL homepage, http://www.orl.co.uk/front.html