GUI AND I/O INTERFACE FOR COACK-II

K. Nigorikawa, I. Abe, J. Kishiro, S. Kurokawa, T. Kosuge
High Energy Accelerator Research Organization(KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan
M. Mutoh
LNS Tohoku Univ. Taihaku, Sendai, Miyagi, Japan

Abstract

COACK-II[1,5] (Component-ware Oriented Accelerator Control Kernel) is a project currently under development at KEK[1]. It is a generalized control kernel in an accelerator domain based on COACK phase-I, which was developed by the KEK PF Linac group. The conceptual system configuration of COACK-II and various test results are described in this paper.

1 INTRODUCTION

The control software for accelerators or large-scale experimental equipment was occasionally developed independently and sometimes dedicated to specific equipment. Based on an object-oriented analysis of control system fundamentals, it was concluded that in spite of the difference in size or architecture of the control software, there exists a common flow of commands or data. The control commands to the accelerator magnet, for instance, come from the control client to the I/O interface module through a command exchanger, in which command security, examination and assignment of the I/O module address and logging into the database etc. are performed. The magnet current or voltage data is returned to the client GUI, and also recorded into the database. In this example, except for the realization method, the fundamentals are security examination, addressing, and logging and recording of the command and data. In the object-oriented model, these are private procedures and the contents of the command and data are public properties. The magnet control is easily classified as a "Magnet object".

2 COACK-II

The COACK-II project is a new accelerator control kernel development based on Object Oriented Modeling and an extension of that to the COM (Component Oriented Model). In COACK-II, the fundamentals of the accelerator control software are constructed using Component ware, and entire control programs are assembled in the Component Oriented architecture. The control components are implemented not only in the kernel itself, but also in the communication between the client GUI and the kernel, and between the kernel and the I/O device. In the control network the many distributed I/O devices are easily implemented in a single kernel by employing DCOM[2] (Distributed Component Object Model).

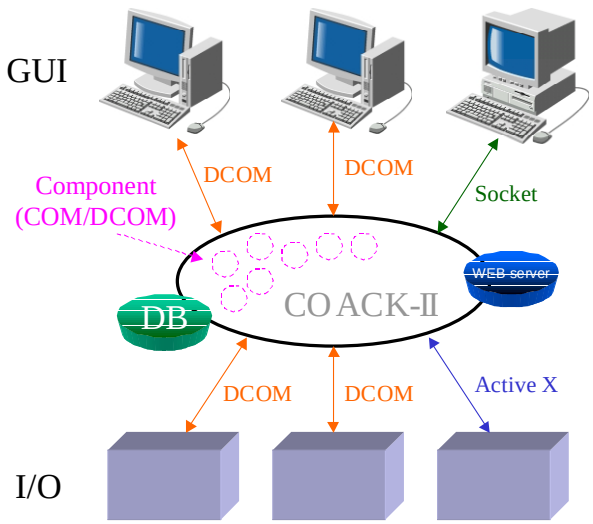


Figure 1: COACK-II

3 GUI AND I/O IN THE COACK-II

The control software usually implements a character-based code according to specially developed I/O access commands. The software developers need to know at least the command provided by the system designer or the hardware developers. By contrast, COACK-II is designed to assemble all of the control commands based on an object-oriented method. For instance the objects are classified as "Magnet" and "Vacuum" etc., each of which contains the data and set-up procedures for the concerned devices. The objects are also implemented in COM/DCOM as a communication procedure. The COACK-II kernel is mainly implemented by employing the component-oriented method. This allows the main flow of data and commands to the accelerator object through the kernel to be implemented via a COM/DCOM channel. The fundamental components in the kernel communicate with each other through this channel. Transmission of the control command is an equivalent action to change the property of the concerned object. The COM/DCOM architecture provides automatic

[1] This project supported by JST.

communication between the kernel components and the user GUI presentation frames. The component-ware architecture makes it possible to build up a complicated control GUI by dragging and dropping the components that are already implemented in the system.

As a result of the many types of I/O devices in an accelerator control system, the software developer must implement many device handlers. To avoid this frustration, COACK-II components are designed to include the communication channel to the currently available SCADA software.

4 GUI AND I/O CONNECTION TO COACK-II

4.1 Connection Protocol

The connection between the user GUI and COACK-II is implemented using COM/DCOM. COACK-II also incorporates a TCP/IP Socket, thus allowing the implementation of multiple platforms.

The string-command goes through the network from the user GUI and COAKC-II, in which a class, object and property name are formatted specifying the components in the control network. COACK-II incorporates a Message-exchanger to decode this command and inheritance for the component, which corresponds to this command. The Message-exchanger examines the registered component in the system, and transmits the command to the instance, or relays a message to the user GUI if the component is not available. Because these communications are encapsulated by DCOM, the user or developers need not know the command. Figure 2 depicts the command flow in COACK-II.

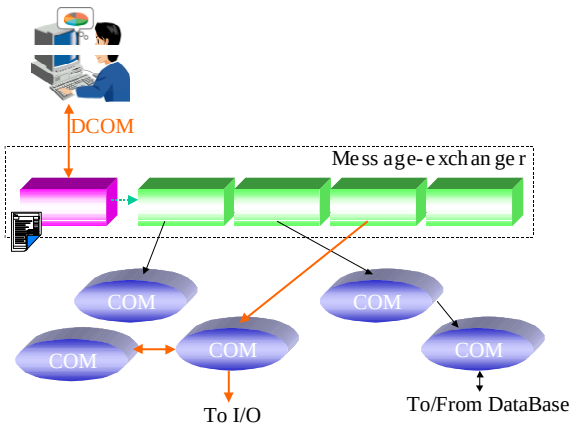


Figure2: Message-exchanger

4.2 Implementation of the ACOP

For the GUI layer, we examined the ACOP[3] (Accelerator Component Oriented Programming) system developed at DESY/CERN. ACOP demonstrates excellent performance on our COACK-II project. ACOP also has the advantage of being able to be downloaded from the Internet. ACOP is the pioneer of component-ware for accelerator control systems.

4.3 Implementation of SCADA

SCADA (Supervisory Control and Data Acquisition) software is strongly supported in COACK-II, because this software is commercially available to every control developer. SCADA not only supports direct access to the I/O devices, but is also a means of GUI. SCADA is used in COACK-II as local intelligence in the I/O layer, which provides the possibility to develop local control independently of the whole system. SCADA local control is implemented as a local component using ActiveX. This configuration explicitly smears out the varieties of I/O devices and accelerator objects in COACK-II.

4.4 DCOM performance

Using DCOM as the protocol for network communication enables the components to be distributed in the network. DCOM is not a simple communication protocol, but it can be superior to the network, which contains various security checks of the distributed components, and so on. We have measured the communication speed on the network compared to a simple TCP/IP Socket using WindowsNT (Pentium III 450MHz, 100Base-T). A formatted command, as described above, was transmitted 10,000 times to the server computer through Fast Ethernet, and the transfer time was measured. The results are given in table 1.

Table1: Connection speed

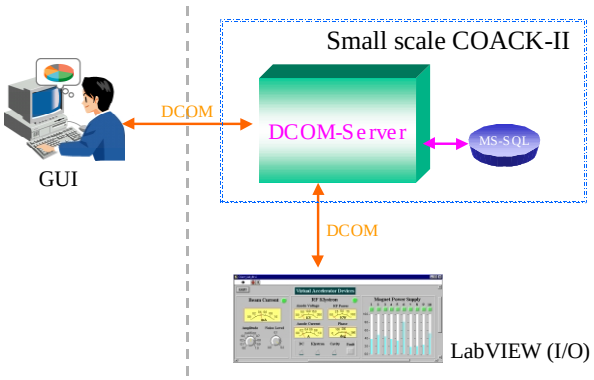
Protocol	Speed (mSec)
TCP/IP Socket	0.49
DCOM	3.58

Using DCOM, a single command can be transmitted in 3.6ms, approximately seven times longer than the same command transmitted via a TCP/IP Socket. The communication speed is fast enough to transfer the command to COACK-II. DCOM is itself a network-oriented communication, which includes some types of checks that DCOM is reliable enough to be used in the COACK-II project.

4.5 Communication test

The program shown in figure 3 was created for a communication test between the GUI, the I/O and COACK-II. This program also tested the DataBase. The GUI was written in Visual Basic and implemented using ACOP. Figure 4 shows the GUI. This software and the DCOM-Server communicate using DCOM; the I/O and DataBase communicate through the DCOM-Server.

LabVIEW was introduced for the I/O layer to control many devices. LabVIEW and COACK-II also



Figer3: Test program

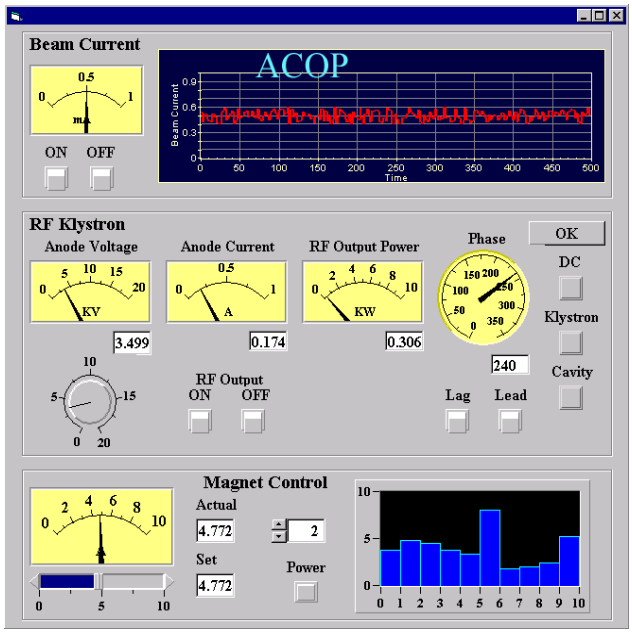


Figure4: Window of GUI

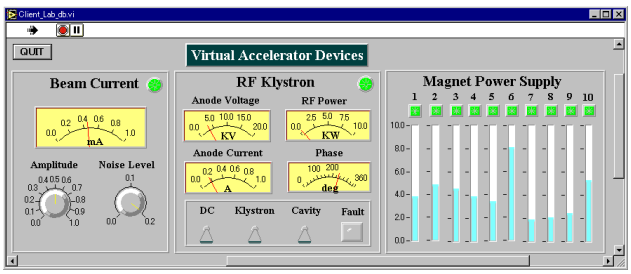


Figure5: Panel of I/O

communicate using DCOM. Figure 5 is an example of I/O control using DCOM.

Through the use of ActiveX, DCOM communication can be implemented into LabVIEW.

The DCOM-Server in this test program is a small-scale COACK-II.

The test result showed a fast response and high level of reliability of device control over the network.

5 THE NEXT PLAN

Through the use of standardized OPC[4] (OLE for Process Control), COM and ActiveX technology, I/O can be connected.

Therefore, it can be said that the conditions under which various I/Os connect to COACK-II will be widened.

6 SUMMARY

A new from of connection between the GUI and I/O has been proposed. The actual connection between COACK-II and I/O was demonstrated, and the statistical results presented. Remote control using the network was easily realized and satisfactorily performed.

Because COACK-II was created from component-ware, it is easy to add new functions dynamically. Furthermore, multiple connections of SCADA software and expansion by OPC can be expected.

REFERENCES

[1] I.Abe, et al., “COACK-II Project on Accelerator Control Kernel Development”, this conference.
[2] Guy Eddon, et al., “Inside DCOM”, Microsoft Press
[3] P.Duval, “Using ACOP in HERA Control Applications”, PCaPAC’99, Tsukuba (1999)
[4] http://www.opcfoundation.org/
[5] http://ninja.kek.jp/COACK2/