INTEGRATING ACOP WITH UNIVERSAL GUI FOR ACCELERATORS

S.Dasgupta, R.B. Bhole & Sarbajit Pal
Variable Energy Cyclotron Centre
1/ AF Bidhan Nagar, Calcutta 700 064

Abstract

The Java-based universal sharable GUI development work[4] proceeded with incorporation of MSChart ActiveX so far, for graph displays of accelerator parameters. To gain from the accelerator control oriented features of the 'Graph Control' devised at ACOP (Accelerator Component Oriented Programming) workgroup [3], acop.ocx has been integrated in the "Universal MMI Applet" container. The 'device name' property is resolved at a level just below, and a corresponding GUI is activated. The selected GUI object has properties and methods to make transactions with the devices through a compliant dll. The dll in our case, is being developed to facilitate distributed data access by communicating with an Active Server Page running on the data base server, containing the control system dynamic data[2].

1 INTRODUCTION

An effort to integrate ACOP ActiveX control in our previously made applet prepared for the development of Universal MMI, has been undertaken.

Full incorporation of ACOP into a Microsoft J++ container appeared to be very restricted as yet. The major part of the work reported here, was to incorporate connectivity between our Universal MMI and the control database. In the ACOP methodology, an ACOP.DLL is necessary to interface the control database or data source to the application client. The data acquisition DLL's to access the device data from a remote data server machine have been implemented by using UDP sockets. Dynamic updating of data at the client site is also provided in this process. While ACOP compliance in the interfaces of these DLLs are being prepared, these have been used in a VB document and data access from an Universal MMI based on this VB document, instead of a Java applet, are carried out.

Many of the method calls in the rendition part of ACOP need variant type parameters. The Java VARIANT class of Microsoft in VJ++ could not be successfully used for passing arrays of any kind. The Universal MMI based on the Java language, for platform independence, could not therefore be integrated with ACOP, for graph plotting, as planned. Instead the data rendition part of the ACOP has been also used in this VB document and accessed from a browser client.

2 DATA TRANSACTION SCHEME

Our database is kept in an Oracle server, residing in a dedicated NT Machine as shown in Figure 1.

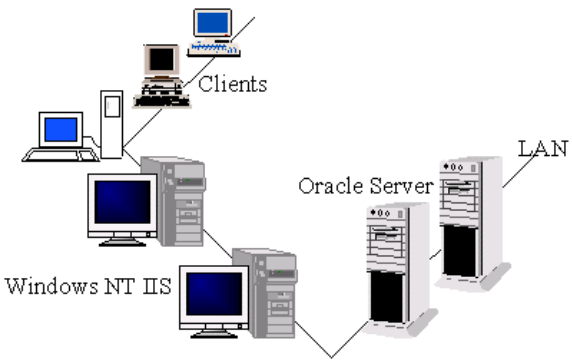

Figure 1: Client Server NW Scheme

The client browser accesses the data from the server database through a UDP transaction. For the client sever communication, a WinSock control is used to connect to a remote machine and exchange data using User Datagram Protocol (UDP) and this control doesn't have a visible interface at run time. Creating a UDP application is simpler because the protocol doesn't require an explicit connection. To send data between two controls, the RemoteHost property is set to the name of the other computer. The RemotePort property to the LocalPort property of the second control is set and Bind method is invoked to specifying the LocalPort to be used. Both computers can be considered "equal" in the relationship, but the server broadcasts only the parameter values which have changed recently on a particular port.

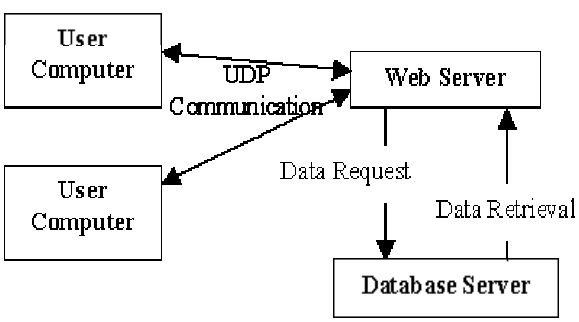

Figure 2: Data Transaction Scheme

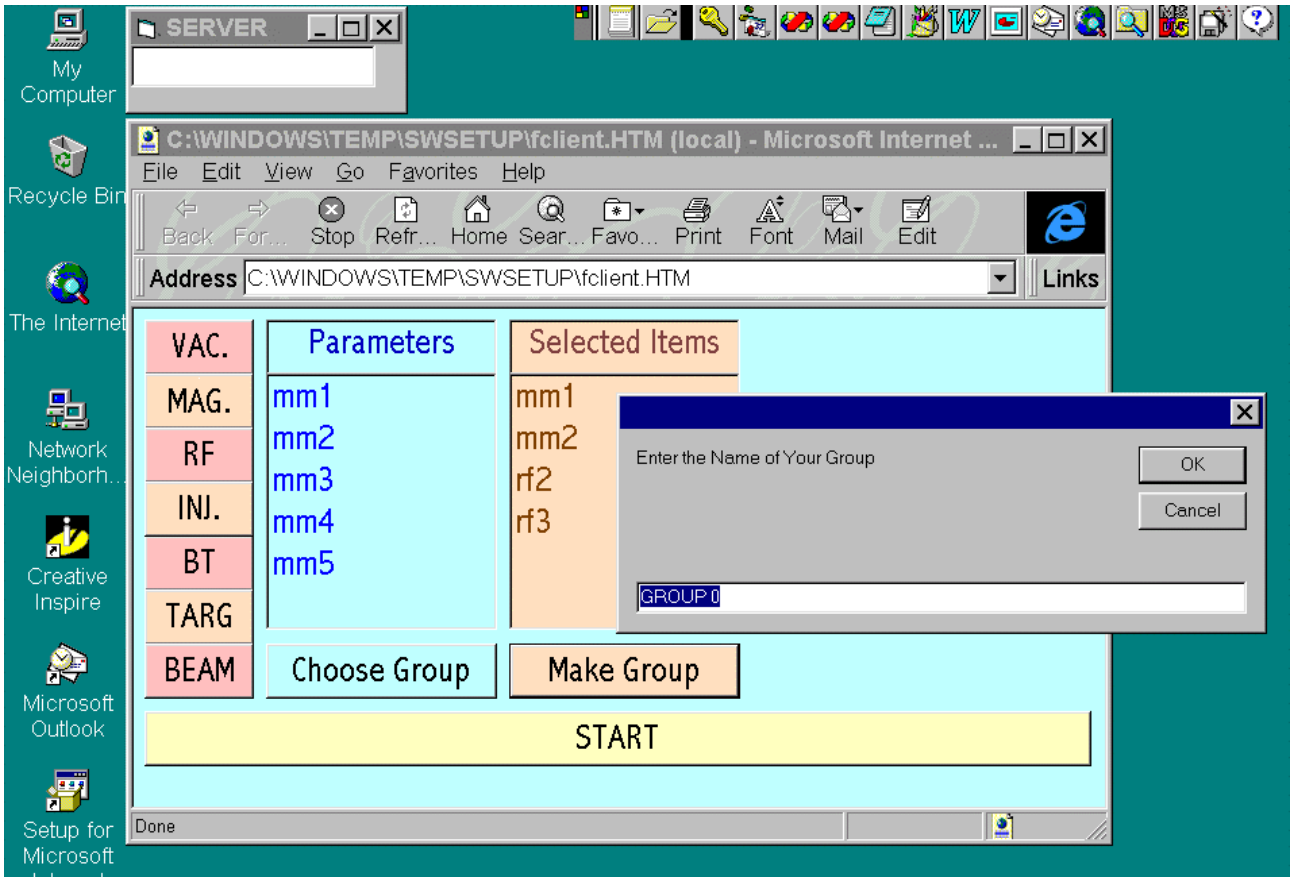

Figure 3: VB Document based DB access MMI

The scheme (shown in Figure 2) implemented presently uses the 'ADO' (Active Data Object) component which is the most recent addition. The previous 'DAO' (Data Access Object) allowed programmers to access the Microsoft Access database only, whereas 'ADO' can access all major databases. With 'ADO' incorporated, the application sees three types of objects i.e. connection, command and Recordset objects. These objects establish connection, execute commands against the database and hold the record retrieved from the database or the records to be updated on the databases. The ODBC data source manager in the Windows Control Panel sets up ODBC data source and Data Source Name (DSN). In Visual Basic code, an ODBC connect string is used to establish a connection to the data source. To access a data source in the code, 'connect' string is required by the ODBC driver to locate and connect to a data source. For accessing ODBC data, the data source is to be registered using the ODBC data source manager. The data source in the Windows Registry makes the information available to applications. The setup for each ODBC data source varies because each data source driver requires a different set of information. The Connect string is used to create a linked TableDef object that points to data within an ODBC data source. A Recordset object on the linked database table contains ODBC data that can be manipulated using the properties and methods of a Recordset object.

3 ACOP INTEGRATION

ACOP ActiveX component, has been used as a server component which has enabled using even a smaller size client. It is found that until the release of Visual Studio 6, the Java and ActiveX communication is limited to "string" parameters only. Therefore, an ActiveX document created in Visual Basic that can run on a container like Internet Browser was implemented.

The ActiveX document is easy to create and distribute. The ActiveX Document has been developed in the Visual Basic Environment from where all parameter of ACOP could be set and various methods supported by ACOP could be readily used for on line graphical display of acquired data. Figure 3 and Figure 4 depict higher level device parameter group formation and displaying on the ACOP graph after acquiring the current value from the central database server.

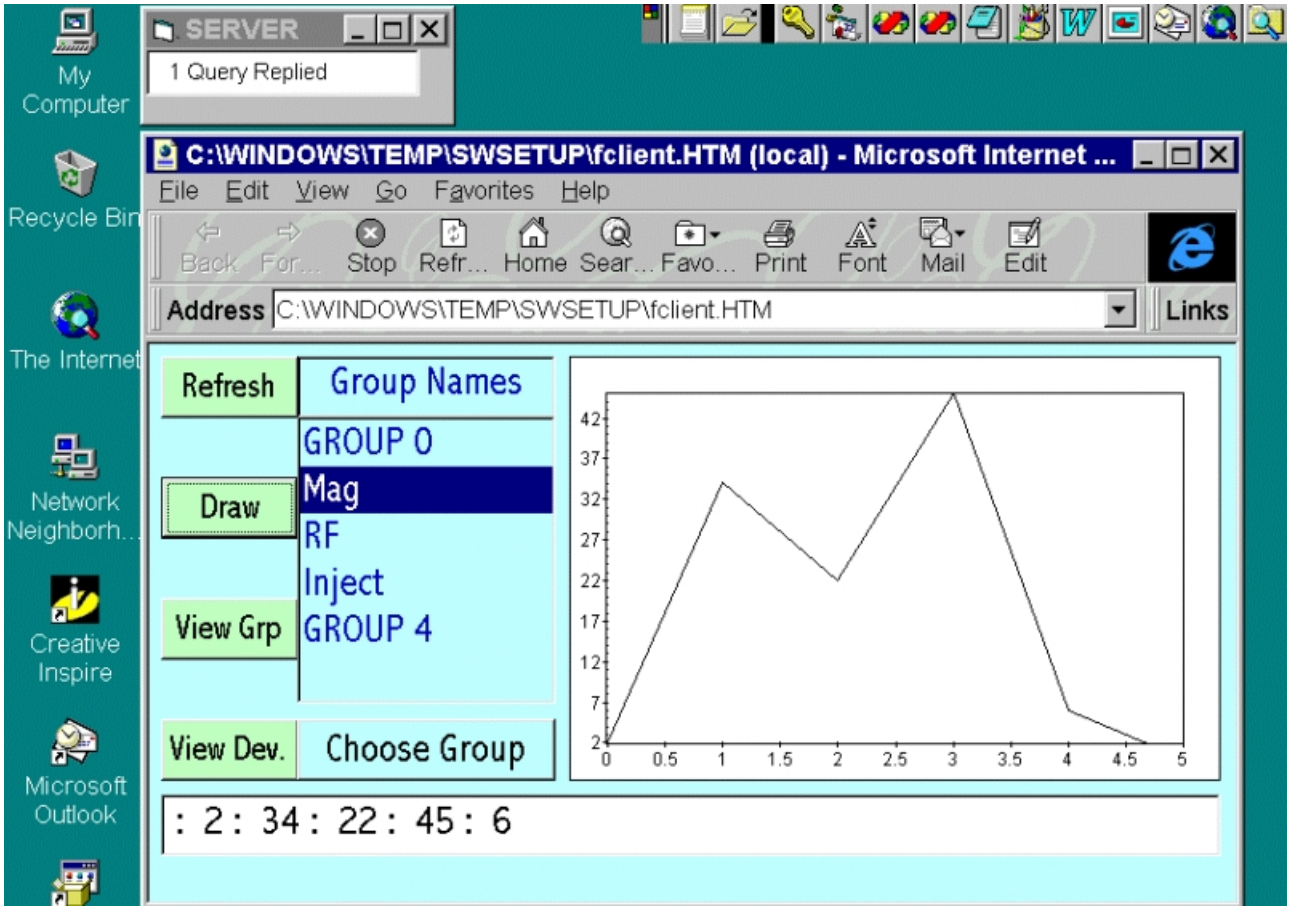

Figure 4: ACOP rendition in Universal MMI

4 CONCLUDING COMMENTS

At the time of this writing, one can not reliably use any type for a property other than string class. Even though the Microsoft documentation shows that one can use other classes, the reality is that only the String class will allow one to actually use the properties in the environment that normally supports Java applets and ActiveX controls.

The client server transaction between client browser and server application is more versatile and fast. The quickest response in the local data transaction is achieved by using 'ADO'. Whereas, the dynamic graphical display is ensured by the ACOP control. The used WINSOC and ACOP control in the web-page if not available at the client machine can be downloaded dynamically from the server. Once downloaded on the client site, it is installed and does not require further downloading unless the source at the server is modified which provides easy distribution of the updated application.

This scheme can also be implemented by Client-side scripting using Visual Basic or Visual Java Script. Even lighter clients can be implemented by HTML at the client side and using server side scripting where the internet server does all the work.

The advantage of platform independence of a Java applet MMI must wait until full workability of ActiveX on Java is easily attainable. Until that time, ACOP ActiveX in a VB document is conveniently used with the niceties of both. When using ACOP ActiveX, one forgoes cross platform compatibility because ActiveX is strictly usable in the Microsoft environment. One can as well use ACOP in VB environment, where it integrates seamlessly, if the above constraint is acceptable.

REFERENCES

[1] I.Deloose, P.Duval, H.Wu, "The Use of ACOP Tools in Writing Control System Software", Proceeding ICALEPS'97.

[2] Sarbajit Pal, S. Dasgupta, "Advancement Towards Development of Shareable Accelerator MMI", Proceeding PCaPAC'99.

[3] Philip Duval, H. Wu, "Using ACOP in HERA Control Applications", Proceeding PCaPAC'99.

[4] S. Dasgupta, Isamu Abe, "Sharable GUI Objects for the Operator's Console", Proceeding ICALEPS'97