EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
Laboratory for Particle Physics

Divisional Report **CERN LHC/99-6 (IAS)**

Integrating the Latest Technologies into a Java Process Control MMI

F. Momal

The LHC/IAS (equipment control) Group is developing supervisory systems by means of industrial SCADA packages. For the past four years, we have provided Web remote access to the data coming from our supervisory and control systems. Combining our findings, an architecture and a strategy have been set-up for a generic Java interface which offers a remote and unique access to all kinds of control data. Using the object-oriented technology, the architecture dissociates the data access layer from the presentation one. Thus, the interface may be used to access different types of data. The data are stored in the interface together with a set of related information (acquisition date, unit, etc.). The graphical interface is based on components which may be stored independently and which can be accessed on demand. Attention has been given to easing the integration of commercial components. To help non-specialists in creating components, a graphical scripting language has been developed.

PCAPAC'99 - 12-15 January 1999 - Tsukuba, Japan

Administrative Secretariat
LHC Division
CERN
CH - 1211 Geneva 23

Geneva, Switzerland
10 November 1999

Integrating the latest Technologies into a Java Process Control MMI

F.Momal, CERN, Geneva, Switzerland

Abstract

The LHC/IAS (equipment control) Group is developing supervisory systems by means of industrial SCADA packages. For the past four years, we have provided Web remote access to the data coming from our supervisory and control systems. Combining our findings, an architecture and a strategy have been set-up for a generic Java interface which offers a remote and unique access to all kinds of control data. Using the object-oriented technology, the architecture dissociates the data access layer from the presentation one. Thus, the interface may be used to access different types of data. The data are stored in the interface together with a set of related information (acquisition date, unit, etc.). The graphical interface is based on components which may be stored independently and which can be accessed on demand. Attention has been given to easing the integration of commercial components. To help non-specialists in creating components, a graphical scripting language has been developed.

1 TOWARDS A NEW INTERFACE

Process Control Man Machine Interfaces (PCMMI) whether they be commercial SCADA (Supervisory Control and Data Acquisition) packages or a home-made development, do not often stand the comparison with standard office software in terms of user-friendliness and programming facilities. We see generally two kinds of programs: either graphical interface generators without a strong model for fetching or storing the process variables or complete SCADA systems which are hard to extend.

It seems a good time, nowadays that everyone is testing the usefulness of Java® in this area, to try to make Java supervisory and control packages evolve with the latest trends in software technologies.

Making a Modern MMI

We would not accept on our office computer what we still see on control operator consoles. Why can we not cut and paste a dial or a led from one SCADA window to another? Why can we not save a user configuration by clicking on the floppy icon of a toolbar? The first examples of Java interfaces to control systems are contained in fixed applets without even the possibility to have two windows open at the same time.

Microsoft® has popularised the notion of dividing an application between a document and a view. The document holds everything linked to the intrinsic meaning of the data whether the view contains the information on how the data is to be seen; a PCMMI has exactly the same construction. On one side there is the process variable with its value and some related data and on the other side, the graphical representation of the process variable. While the handling of process variables has specific requirements (see "real-time database" below), all the evolutions in terms of user-friendliness and interactivity that come nowadays with the office applications, may be applied to the views. For example, the operator should be able to create a new resizeable view (seen as a new window), and by clicking around, copy and paste parts of other views. A good way of seeing a tree-like structure is the explorer view. It could be useful to apply this to views which are made of subcomponents like, for example, dials, text fields, etc.

New concepts coming from the Internet

To use Java means using the Internet facilities. Not only the PCMMI itself may be downloaded from a remote site, but it should be possible to open views which are stored on some Web servers and to remotely save operator made configurations. Hyperlinks must, of course, be implemented so that the user may navigate, with a single click, between views that are stored on different Web servers. It should be possible to access, with some kind of URLs, not only a view but also a simple graphical component.

The success of the Web is largely due to the simplicity of creating a Web page. The same simplicity could be brought to PCMMI views by the definition of a well-adapted scripting language[1].

Java approach

Java is a powerful object-oriented language whose facilities may be efficiently used in the design of a PCMMI. Java interfaces are a good way of specifying how the different parts of the system are to work together, without fixing how they are implemented and thus easing cooperative development. We see at least three main interfaces: one describing the real-time database which is at the heart of the system, one describing the part which fetches the values of the process variables from the network and one describing how the graphical views behave.

The Java event system is well suited for the communication between the real-time database and the views and between the views' components. These software components should implement the JavaBean

[1] The existing graphical description languages are almost never usable with a simple text editor.

standard. All classes can benefit from the Java serialisation scheme to implement some persistency.
These different points led us to set-up a software structure for a PCMMI in Java.

2 IMPORTANT PARTS OF A JAVA MMI

Real-time Database

In order to realise, in practice, the division between the process variables and the views, a PCMMI should be constructed around some kind of real-time database (RTDB) communicating with graphical views. This would also ensure the coherence between the different representations of the same process variable. The real-time database should have some knowledge about the kind of data which it handles. Together with the value of the process variable, a set of information known as "metadata" is to be stored. Typically, this will include the possible range of the variable, the abnormal values, the physical unit in which it is expressed and a status telling whether the value is considered valid or not[2].
A short-term history may also be saved in the real-time database. It can then be accessed without fetching it from the network.
Java is an object-oriented language; therefore the real-time database should offer an object approach of the variable.

PCMMI Graphical Components

The graphical part of a process control MMI may be seen as a set of representations of process variables and a set of user actuators (buttons, etc.). These representations can be symbolic and linked to the physically controlled element, such as the standard representation of a valve, or can be more generic, such as a trend curve. In either case, the representation may often be applied to different processes. By standardising the software structure and the behaviour of an MMI atomic representation, one should be able to reuse others developments and not rewrite indefinitely the code needed to represent a led or a pump. Component's technology, as defined by Sun® or by Microsoft®, is well suited to MMI visual components. Any modern process control MMI should use this technology to easily integrate other developments and to gain modularity. Sun has defined a way of developing software components (JavaBeans[3]) and since a MMI component shares the graphical behaviour of a JavaBean, any Java PCMMI integrating the notion of software components should stick with that definition. This would ensure the durability of the developments. It will also ease the use of off-the-shelf components which may be of interest, such as a trending component.

Communication between components is normally made with Java events. This is also perfectly suitable for our kind of needs. Events, as they are described in the Java 1.1 specification, may be used to update the graphical representation of a component whenever the process variable to which it is linked to changes. Events are also a good way of transmitting user actions such as a mouse click.
However, MMI components have specific constraints if they must be automatically linked to data in the real-time database and they must behave in a homogeneous way. Therefore, the definition of a PCMMI component is more restrictive than simply following the JavaBean standard. The interface we defined for our system comes with the description of a standard behaviour. Specific events are used and a way of saving and loading the configuration files is recommended. PCMMI components are also asked to offer some specialised introspection mechanism so that they may be inserted into other PCMMI components. By following these recommendations, a component is certain to be smoothly inserted into the system.
A very powerful, yet simple to use, mechanism exists in Java which enables the loading of a class (which could be a software component) at run-time and automatically bind it to the application. This can be used to load, on demand, the components asked for by the user from a remote repository. While lightening the application, it ensures all the users that they use the latest version of the components[3]. It also allows integrating new components without having to restart the application.

A scripting language

Scripting is a fast and powerful way to assemble components together and to describe how the different software parts of a system should behave together. Most of a process control display may be described easily with a scripting language. While it is important to have an efficient way to program components, a computing knowledge is not required to position these components and to connect them to the real-time database. A WYSIWYG interface is always welcome for that kind of work but it is never as powerful as a good, adapted scripting language. The best example is, I believe, the success of HTML. A graphical oriented and editable scripting language is the best and easy way to describe relations between components and even to create simple PCMMI components. It also has the enormous advantage to be easily generated by programs and ensures the durability of this part of the development. This is very seldom in SCADA systems.

[2] In case of network failure for example, the value can't be considered as valid.

[3] We use the same mechanism to store and load the network drivers.

3 ASSEMBLING ALL TOGETHER

We have tried to implement all the previously described ideas into a system which we developed and which is named "RemoteView". It is based around a set of interfaces which describes how the real-time database, the network drivers and the graphical components are to be accessed. A clear split has been made between the RTDB and the graphical representations. Several implementations of these interfaces have been made. Socket, HTTP, …

The interaction between the components and the RTDB is based on a subscription mechanism. A component subscribes the set of process variables which it displays. Whenever one of the variables changes, an event is sent to the component.

Each component offers one or several ways of representing a set of process variables. We call it a representation type. A component loader has the function of finding, on demand, the Java component that offers a given representation type. It will do so by scanning the component repository. This repository may be a Web server, a database or a simple file system.

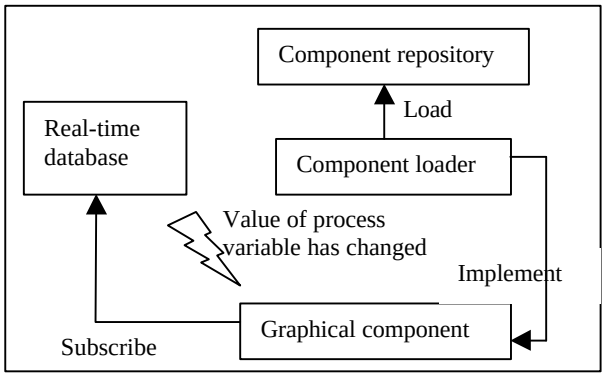

Figure 1: The Graphical Components

A configuration manager, which mimics a file system, handles the remote storage and the retrieval of all the configuration files. It may be used by all the PCMMI parts to save their status.

Events are used to inform the window manager of the user actions. For example, whenever the user clicks on the representation of one or several variables, an event is broadcasted. Using this feature, it is easy to insert, for example a set of variables into a trend chart or to paste a component into a panel of components.

Each view is inserted in a specific window with a menu bar. This resembles the menu bar of any standard application with "new", "open" and "save" buttons.

By using an introspection mechanism, the window manager is able to represent the views as a tree-like structure. This facilitates the manipulation and the recombination of the views.

A graphical scripting language named dgsl (dynamic graphic scripting language) has been defined. The dgsl interpreter, which is a standard graphical component, looks into the script for the RTDB variables it has to subscribe to. Each modification in the value of a subscribed variable launches the execution of the script.

The first release of this Java interface is now running.

REFERENCES

[1] F.Momal, C. Pinto-Pereira, "USING WORLD-WIDE-WEB FOR CONTROL SYSTEMS", ICALEPCS 1995, Chicago (http://wwwlhc.cern.ch/ICALEPCS95/icalep95.htm)

[2] F.Momal, "TOWARDS A NEW GENERIC APPROACH FOR WEB ACCESS TO CONTROL DATA", ICALEPCS 1997, Beijing (http://wwwlhc.cern.ch/Docs/p005.PDF)

[3] Prashant Sridharan, "JavaBeans", Prentice Hall PTR, 1997

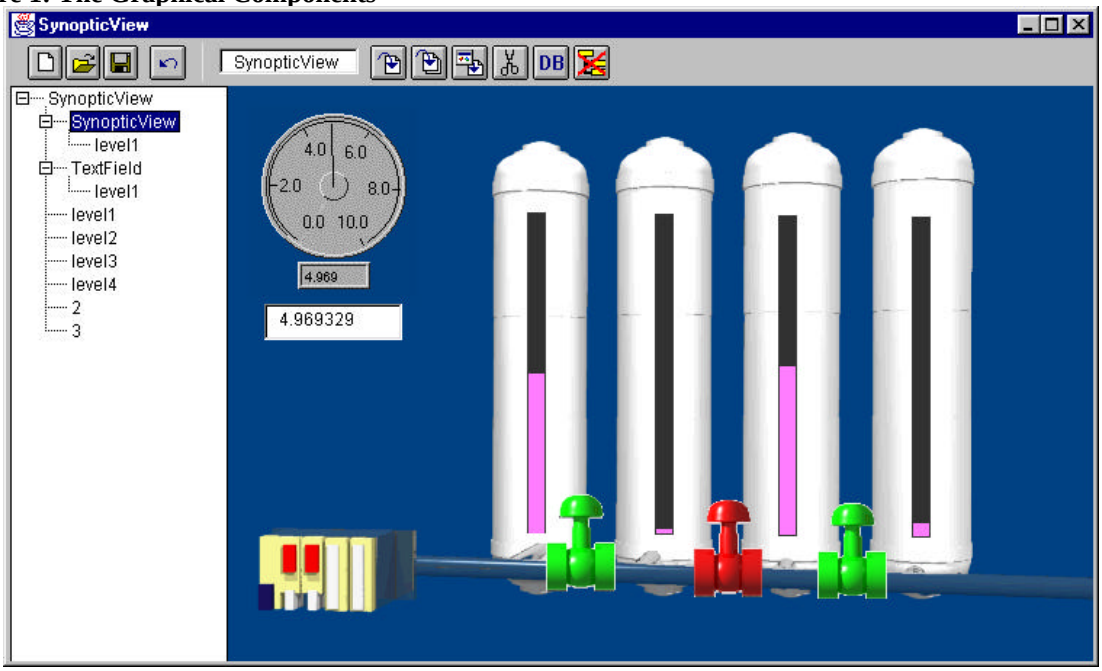

Figure 2: The dgsl interpretor in its window