Graphical User Interfaces for a Data Acquisition System

Ambrosini G1, Bee C.P2, Buono S2, Ferrari R1, Fumagalli1, Ganve P2, Hellman S2, Jones R2, Khodabandeh A2, Mapelli L2, Mornacchi G2, Polesello G1, Prigent D2, Rondot C3, Tamburelli F2

1: Dipartimento di Fisica dell’Universita’ e Sezione INFN di Pavia, Italy
2: CERN, Geneva, Switzerland
3: Centre de Physique des Particules de Marseille, IN2P3, France

RD13 has adopted Motif as its graphical user interface toolkit and XDesigner as a tool for constructing Motif based user-interfaces. Some of the limitations of Motif and tools such as XDesigner are highlighted. We discuss the problems for constructing user interfaces needed for data acquisition systems and suggest possible solutions.

1 Introduction

Today, most computer systems use some graphical display to communicate with the user. The X Window System [1] has become a de-facto standard package for building user interfaces on bit-mapped computer screens. The X Window System offers a relatively low-level of functionality and so toolkits have been built on top of it to simplify the task for the Graphical User Interface (GUI) developer and to ensure a uniform interface across applications. The X Window System based GUI toolkit that is available on the largest number of computer systems is Motif [2] from the Open Software Foundation (OSF). The RD13 [3] project at CERN has adopted Motif to run on its network of heterogeneous UNIX workstations. In this environment we have studied several tools that aid the production of the various user interfaces that are required in a data acquisition system (DAQ).

2 Limitations of Motif

The Motif toolkit simplifies GUI programming by providing ready-to-use components (or widgets) such as pop-up menus, scroll bars, editing windows, buttons and dialogue boxes. With these widgets a developer can construct nearly all the necessary facilities of an interface with a few limitations.

Firstly, among the widgets proposed, there are no specialized facilities to display data in a graphical format such as a table or a histogram. Developers are obliged to either build their own widgets or obtain them from other sources and add them to the set offered by Motif.

Secondly, there is no complete facility for providing on-line help information with applications. Many Motif widgets have a help button and associated callbacks that allow the interface developer to indicate help is available to the user and provide a way of signalling to the application that such help has been requested. But the toolkit does not include a suitable means of displaying the help information itself.

Thirdly, Motif does not provide a desktop manager. The lack of a desktop manager poses a problem since users still have to interact with the operating system through its command lan-
guage. On UNIX workstations, in order manipulate their file directories, start applications and so on, the user must use complex shell commands.

We divide the types of interfaces required in a DAQ system into three types: when a user has to work with a computer system (system interface), to develop an application (programmer interface); and to interact with an application (operator interface). Each type of interface requires tools with specific features and facilities.

3 The system interface

To act as a desktop manager in Motif we have chosen a product called X.desktop[4]. It offers a Macintosh-like interface to users on UNIX workstations (figure 1). The Macintosh approach provides a convenient and intuitive system interface for both programmers and new users. This kind of interface allows the user to perform operations without having to learn a UNIX shell language. X.desktop can be customized by the developer or the system manager to the working environment. One can associate to each type of mouse input action a set of rules written in a language similar to the UNIX Bourne shell. For example, it is possible to write a script so that when a user drags a file on an icon representing a letterbox it is sent as an email message. The user can open several desktops at once giving the impression that many Macintosches are on the same screen.

A problem seen with X.desktop is that the office represented on the screen rapidly becomes overcrowded and may prove confusing to novice users. Secondly, the number of possible combinations a user can perform with the mouse is too high and their use may be inconsistent between the various icons. Thirdly, there are two views of the same file system: the traditional shell interface showing all the files and directories present; and the X.desktop view of the file system where some files may be hidden or relocated as a result of the rules in the scripts. This disparity may be a source of confusion for a user who has to swap between the two interfaces.

4 The developer interface

The Motif toolkit offers a higher level of abstraction to the GUI developer than the XWindow libraries and so simplifies the programming task. Nevertheless, programming an interface with Motif remains a complicated task and so we have adopted a GUI builder tool, called XDesigner[5], which can automate some of the code production.

Using XDesigner, the developer can interactively design the user interface of an application by selecting the appropriate Motif widgets from a palette (figure 2). When the interface satisfies the developer, he can ask XDesigner to generate C code with embedded Motif calls needed to build the interface. This code can then be integrated with the application to produce an executable program. We have developed interfaces to a number of components of our data acquisition system, including the run-control system (figure 3), with the help of this tool.

5 The operator interface

The DAQ operator must be able to monitor the behaviour of the system and to send commands to the various components. This implies a need for a clear view of the data to be controlled and access to some on-line help facilities that provide information on the system and its components.
To provide the on-line help to the operator, and because no such facilities are provided by Motif, we have chosen a commercial product called FrameMaker[5]. FrameMaker is a very powerful WYSIWYG editor that also offers facilities for creating hypertext links between documents. The information to be provided in the on-line help for an application is written using the FrameMaker editor then saved as a document. When the operator presses a help button on the application a callback routine is activated that starts a read-only version of FrameMaker, called FrameViewer, giving it the name of the document. FrameViewer displays the document on the screen allowing the operator to navigate through the document page by page or by following the hypertext links.

We have also investigated various data visualization packages for displaying the actual data acquired and its characteristics. DataViews [6] is a toolkit for the acquisition and display of data that includes an interactive graphical editor which allows the developer to select the source of the data, be it from a process, a file or the result of some calculation, as well as the manner in which it is to be displayed such as a histogram or pie-chart. With this tool we have developed an interface to our DAQ that allows the operator to visualize the characteristics of the physics events that are currently in the computer memory. The display shows, amongst other information, the number of occupied data buffers, the relative size of the events, the average event rate and the number of free buffers available.

While we are reasonably satisfied with the interface we have developed, the use of DataViews presents a number of disadvantages. The toolkit itself is relatively expensive and requires a run-time library for the data acquisition. The interactive graphical editor does not cover all the facilities offered by the toolkit and so some programming by-hand is required for sophisticated interfaces. Lastly, the DataViews toolkit and interfaces are proprietary software and are not available on all the platforms used in the RD13 project. We are looking into alternative means of building similar interfaces. One possibility is to supplement the Motif widgets available through the XDesigner tool with widgets capable of displaying data in graphical forms. Another approach could be to use a well established graphics package like PAW [7] or commercial alternative such as PV-WAVE. We are also starting to see the arrival of complete data acquisition software packages, such as labVIEW [8], on UNIX workstations. labVIEW offers a graphical language in which the developer arranges icons, representing program components, on the screen to make a circuit that can capture and display data.

6 Conclusions

We have adopted Motif as the standard graphical user interface toolkit for the RD13 project. Despite a number of limitations with the toolkit, we have built interfaces to various components of the data acquisition system. The GUI builder tool, XDesigner, has proved very useful in building many of these interfaces. The lack of any widgets in the Motif set for displaying data in a graphical form and for displaying on-line help information has forced us to use other tools, such as DataViews and FrameMaker, to overcome these limitations.

References

1. L.Mapelli et al., A Scalable Data Taking System At A Test Beam for LHC, (CERN/DRDC/91-1).
CERN Program Library Entry Q121. October 10, 1989.
FIGURE 1. The system interface: using X.desktop.
FIGURE 2. The developer interface: using XDesigner.
FIGURE 3. The operator interface: using the run-controller and FrameMaker for on-line help.