EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

The Model Human Interface

C.Boissat, R. Jones, G. Mornacchi

DD Division

CERN

CH 1211 Geneva 23

CM-P00059911

Abstract

A human interface package has been developed in the context of a modular software system targeted to data-acquisition and online applications in High Energy Physics experiments.

1. Introduction

We describe the user interface packages developed in the context of the Model [1] project at CERN. A detailed discussion of the design of the Model Human Interface (MHI) can be found in [2]. Detailed information about the functional aspects of the system is available in the MHI documentation set [9].

The target environment is the online system of a modern High Energy Physics experiment, a special emphasis is given to its distributed computing aspects.

We start with a brief description of the design requirements and the architecture of the MHI system. We then overview its implementation and the various packages available in MHI. We conclude by discussing its availability and current usage.

2. Design Requirements and Architectural Description

Taking into account the size and complexity of an online system we have focussed our attention on the following requirements:

- Transparent support of applications in a distributed computer system.
- Allow interaction with any application running in the system.
- Provide a variety of interactive modes.
- Insulate applications from hardware and software evolution.

The design of the system is based on a client server model where the client is an application program and the server provides, to one or more clients, a multiple windows environment. In this sense the architecture is defined as 4 layers of services:

- Communication
- Window System
- User Interface
- Application library

The communication layer provides the path between clients and servers. The window system provides multiple virtual terminals (windows) in which applications may perform input and output. Together these layers provide the basic interactive model of the MHI.

The user interface layer is a set of packages that provide various interactive modes (menus, forms, dialogue windows). The application library includes initialisation and ancillary procedures as well as services to control and manage the user interface environment.

3. Implementation Description

The MHI architecture has been implemented for DEC VAX/VMS machines and 68000 micro computers running OS9 [3]. The VMS implementation supports clients and servers whereas the OS9 implementation only supports clients. The communications layer is based on Remote Procedure Calls (RPC) [4] which use the DECNET protocol suite for VAX to VAX communications and a raw Ethernet protocol for OS9 to VMS communications. In addition, for DEC machines running the DECWindows software [5], an X-Window [6] based version of the window system is also available. Figure 1 highlights the different layers of software on both the client and server sides.

The X-Window version allows us to eliminate our communication layer; X-Windows provides a message passing protocol for client server communication. The task of multiplexing many applications on to one screen is performed by the X-Server instead of the Model window system.

3.1 Window System

We have defined a minimal window system implemented on top of proprietary windowing software, namely SMG [7] on VAXes, UIS [8] and DECWindows on VAXstations. The window manager hides the differences in the underlying hardware and manufacturer's software from the higher layers.

The X-Window version contains separate processes to repaint window contents when they are overwritten and keep information about the windows created by the applications.

In addition to the distributed window system we also support a distributed graphics service for VAXstations, based on the client-server model and implemented with RPC, using the GKS implementation standard at CERN.

3.2 Interactive Packages

On top of the window system we currently provide four interactive packages [9]:

TTY Emulator: redirects input and output requests, e.g. normal I/O statements of a high level language, from the standard input and output devices to one of the window associated with an application. This emulator is limited to a subset of the terminal driver functions.

Dialogue Package: provides a simple and convenient interface to the window system input and output functions, eg a single instruction to input and validate scalar variables.

Menu Package: provides applications with various types of menus. Users may use single key stroke accelerators, mouse selections or commands to interact with menus.

Forms Package: provides applications with forms to modify sets of parameters.

All these packages are implemented directly on top of the window system and are therefore hardware and (manufacturer) software independent, which means they can run, unmodified, on all the systems supported by MHI. Figure 2 shows a typical screen with menus, a form and a graphics window.

We have also implemented a prototype of the menu package on top of the DECWindows toolkit [10]. This has the advantage of greatly reducing the amount of code needed and provides an interface which is consistent with the DECWindows style.

4. Use and Availability

The MHI system is currently in use in several experiments at CERN as well as in some external institutions. It is distributed in a format compatible with the DEC VMSINSTAL utility, the standard for the distribution of VMS software. It has to be combined with the Model Process Control system, see [11], to provide the integrated system including distributed human interface and application control.

5. Conclusions

We have described a system supporting a flexible user interface targeted at the distributed environment typical of today's online systems. The design focussed on the easy migration of the services to packages supplied by the computer manufacturers: the implementation of the system on X-Windows as well as the menu package prototype on the DECWindows toolkit show that this goal has been achieved.

The problems concerned with the differences between terminal devices, as well as their location, have been isolated into a well defined layer. This approach has been successful since very little effort was involved in porting the interactive packages to different environments (UIS, X, OS9).

6. Acknowledgements

The following people have also contributed, at different stages, to the developments described in this paper: T. Berners-Lee, M. Berutti, L. Bessone, W. Bozzoli, B. Wessels.

References

- [1] A. Vascotto, Modular Architectures for Online Systems, in: Proceedings 1986 CERN School of Computing (CERN 87-4).
- [2] G. Mornacchi, Architecture of the Model Project Human Interface, CERN DD/87/14.
- [3] Microware Systems Corporation, OS-9/68000 Operating System Technical Manual, order number PMN-OST68.
- [4] T. J. Berners-Lee, RPC User Manual, (CERN-DD, 1989).
- [5] DEC, VMS DECwindows Technical Summary, order number ZK4727 (DEC 1988).
- [6] R. W. Schifler and J. Gettys, The X Window System, ACM Trans. on Graphics 5 (1987) 79.
- [7] DEC, Screen Management Routines, in: VAX/VMS Programming volume 5B (DEC 1988).
- [8] DEC, Graphics Programming Guide, in: MicroVMS Workstation Software (DEC 1988).
- [9] C. Boissat, R. Jones, G. Mornacchi, MHI Documentation Set, (CERN-DD, 1989).
- [10] C. Boissat, R. Jones and G. Mornacchi, Report on the Field Test of DECWindows, CERN-DD internal note.
- [11] J-P. Matheys, MPC User Guide, (CERN-DD, 1989).

Figure Captions

Figure 1: Layered software modules on the client (top half) and server side (bottom half). Case of a server layered on top of the VAXstation windowing software (UIS).

Figure 2: An example of an application's screen including two menus, a form, the terminal emulator window and a GKS window.

UIS Architecture