EARLY EXPERIENCE WITH DECWINDOWS/MOTIF IN THE TRIUMF CENTRAL CONTROL SYSTEM

S.Kadantsev
Joint Institute for Nuclear Research, Dubna,
Head Post Office, P.O.Box 79, Moscow, 101000, Russia

and

M.Mouat
TRIUMF, 4004 Wesbrook Mall, Vancouver, B.C., V6T 2A3, Canada

ABSTRACT

Early experience in the implementation and porting of several control programs to the DECwindows/Motif environment in the TRIUMF Central Control System (CCS) is presented. A description is included of the separation of form and function under the Motif windowing system, and how applications were easily implemented using the UIL interface description language and the VUIT interface builder. Several operator interface (OPI) issues and different ways of graphical data representation in a control system are discussed. Applications involving two cyclotron sub-systems, a fourth harmonic RF cavity and the laser control system of an optically pumped H^- polarized ion source, have been adapted to the new environment. The evolution included porting applications under VMS from UIS to XUI and then to Motif.

1. INTRODUCTION

In the world of workstations there have been a number of window managers and graphical systems with which software developers have plied their trade. In the past, systems have mostly been proprietary and quite limited in their platform interoperability. A major change occurred in these areas when personnel from the Massachusetts Institute of Technology (MIT) and Digital Equipment Corporation (DEC) developed the X Window System (commonly referred to as X). Currently "It is widely recognized as a standard for network-based window systems and is supported by a consortium of well-known, highly respected companies...".[1)]

The X window system has been commercially available since first release of MIT X11 in 1986 but until approximately the beginning of 1990 there had not been a widely accepted, non-proprietary (open) X based window manager. Motif is just such X based graphical user interface. It is the product of the Open Software Foundation (OSF) which is a non-profit organization founded in 1988.

The experience with Motif windows within TRIUMF's Controls Group has been gained mostly but not exclusively from two projects, development of the RF booster remote OPI and a port of the optically pumped ion source's laser OPI. These two projects presented different perspectives of using Motif given that one project was a development from scratch and the other was a port from an old window manager and graphics environment known as VAX Workstation Software/User Interface Services (VWS/UIS).

2. THE HISTORY OF GRAPHICS PACKAGES AND WINDOW MANAGERS IN THE CCS

VWS/UIS is an early windowing and graphics system from DEC that runs on the VMS operating system. In 1987 within TRIUMF's Cyclotron Division, a project was started to explore using a workstation as an operator control console. The original plan was to provide local control for the laser sub-system of a polarized H^- ion source. A VAXstation II/GPX running VMS and VWS/UIS was acquired and used successfully.

VWS/UIS supports many of the features that are now commonly expected with windowing systems. From the initial release of VWS/UIS there were some major restrictions. The product is not an open system and thus has the constraints of proprietary packages. The basic omission is the ability to redirect input and output between workstations. Graphics from a VWS workstation can only be displayed locally. The efficiency of generating interfaces was also found to be a problem. In describing VWS DEC states "Its procedural interface, the User Interfaces Services (UIS), provides a high-level programming interface to the graphic subsystem"[2)] but this avoids the issue of efficiency in producing graphics applications. Although a high-level programming interface is provided, the interface elements (graphic objects), are quite low level and it requires relatively a lot of effort to construct higher level components such as data plots. This issue of efficiency would be aided by some type of interactive interface editor but none is provided. VWS/UIS seems to have been recognized by DEC at an early time as being a dead end product and from their in-

volvement in the development of the X Window System, a new package called DECwindows/XUI was made available. The migration from VWS/UIS to DECwindows is quite difficult because the systems are very different. A guide to migration and some tools are available to help the process of moving to the new system but our experience has been that that is it best to re-examine the application in view of the X perspective rather than try and port the older VWS/UIS ideas.

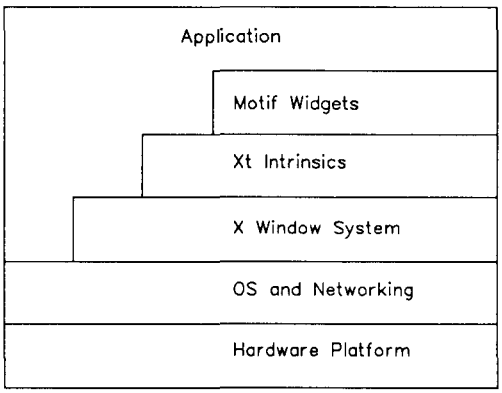

Fig. 1. User Interface Development Model

DECwindows/XUI is a blend of a proprietary window manager and the open X Window System. It addresses some of the open/proprietary issues and takes advantage of the advances of X. One particularly important suppliment to the X Window System is the User Interface Language (UIL). This is a higher level of abstraction than exists in X and provides for the separation of form and function within a graphics application. UIL helps in the topics of efficiency, by allowing a high level of specification of graphics objects (widgets/gadgets), and in organization, by letting you separate the layout and appearance of the application from the functions provided. Again, no interface builder/editor has been included to increase development efficiency. A separate issue is the standardization of the look and feel of applications between various vendors' platforms. The solution to this problem was the need for an open window manager. As mentioned in the Introduction, one answer to this came from OSF with the product Motif. DEC has embraced the Motif window manager with a product called DECwindows/Motif. A migration from DECwindows/XUI to DECwindows/Motif was undertaken on the laser control workstation. Motif is currently available from a number of vendors, for various hardware platforms and operating systems.

The main Motif components are as follows:

1. Style Guide.[3] Describes the principles, philosophy, and components used to build consistent and well-integrated applications.
2. Window Manager. Allows user to manipulate multiple applications on the screen. Plays the principal role of enforcing style guidelines. Includes configurability of appearance and behavior. Provides for consistancy.
3. Interface Toolkit. Provides a library of graphical objects which can be consistently used by all applications.
4. User Interface Language. Allows application developers to describe the presentation characteristics of the interface independently from the application code itself - separate form and function.
5. Documentation. Includes Style Guide, programmer and user documentation.

The migration from DECwindows/XUI to DECwindows/Motif is relatively easy. Typically, procedure and parameter names are straight substitutions and only occasionally are the number of parameters in a call different. Tools are available to make these translations but only some languages are supported, fortran is not one of them. DECwindows/Motif is somewhat faster at drawing graphics than DECwindows/XUI which is somewhat faster than VWS/UIS but in general the hardware platform is a bigger factor in graphics speed than the software being used so although the increased performance is welcome, it is not such a critical issue. A common feature between DECwindows/XUI and Motif is that applications can make use of a multilayer development model (see Fig. 1).

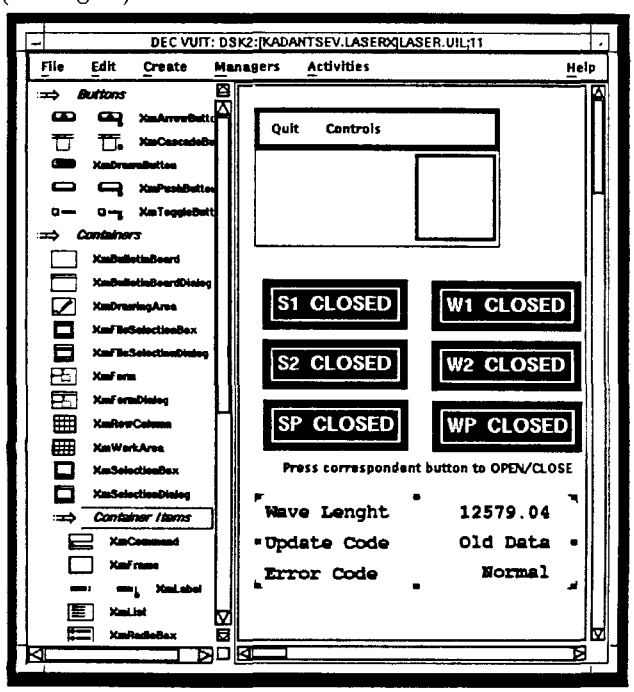

Fig. 2. Using the VUIT Interface Builder

The various levels of abstraction allow developers to accommodate the tradeoffs between performance and

special needs at the low end, and the efficiency of development and maintenance at the high end. In particular, it is worth examining two features at the high end of the abstractions. Concepts within the User Interface Language and the availability of interactive interface builders have been found to be important aids in application development.

UIL is a specification language that describes the characteristics of the graphical interface. This specification describes the graphical objects, known generically as widgets and gadgets, in the interface and specifies the procedures to be called (callbacks) when a change of state occurs as a result of a user action or other event. There is a UIL compiler that translates the UIL source module into a User Interface Description (UID) binary file. When the application executes, the resource manager reads in the UID file and builds the run-time structures necessary to create the user interface. This means that the UID file can be changed to alter the appearance and structure of the interface without altering the executable image and its flow. The result is a separation of form and function. A simple example of this is to have several UID files with text in different languages and only one executable image. The UIL level of abstraction is very useful and has been enhanced by the availability from several sources of interactive interface editors that build UIL. DEC's product which has been used by the TRIUMF Controls Group is called VUIT.

This is a WYSIWYG-style editor that greatly enhances a developers ability to build, test and modify Motif application interfaces. DEC VUIT generates and imports Motif standard UIL which is non-proprietary so the interface's form is portable between environments. Having an interface editor that generates UIL allows the interface to be developed separately as opposed to it being included in the application program. The UIL file which is created contains a static definition of the user interface but at the user's request a simulation of the run-time appearance and behavior of the interface can be obtained using the simulation feature.

Interface builders such as VUIT increase productivity in a number of ways:

1. Laying out an interface with the builder is faster than writing UIL or Xt Intrinsics code and a continual view of the interface is available so the edit/compile/run/adjust cycle is eliminated.
2. Standard UIL and X Window System interfaces mean that applications can be used on a variety of hardware and software platforms without modifications or enhancements.
3. The interface builder checks the legality of resources and relationships and then restricts the user to legal choices, the user does not have to know or look up this information.
4. Portions of existing interfaces can be copied, exported and imported to allow re-use of previous work.
5. Resources are easily configured into the widget definition.
6. Changes made to a single widget in the work area can be propagated to multiple widget definitions throughout an interface via literals and lists.
7. The builder's default parts box can be extended to include the user's own widgets.
8. Extensive, on-line, context-sensitive Help is available.
9. A widget tree browser is available to help manage and navigate the interface and aid in understanding the hierarchy of an unfamiliar interface.

3. LASER APPLICATION

A polarized H^- ion source (I4) has been developed that relies on lasers[4] for optical pumping of a rubidium vapour charge exchange cell. The laser control system continues to evolve from a VAXstation/CAMAC setup using VWS/UIS for local control to a local and remotely operated system using Motif. Figure 3. shows the simplifide block diagram of the I4 laser control system. The local VAXstation connects to CAMAC and can run autonomously and remote X windows servers can connect to that VAXstation via ethernet.

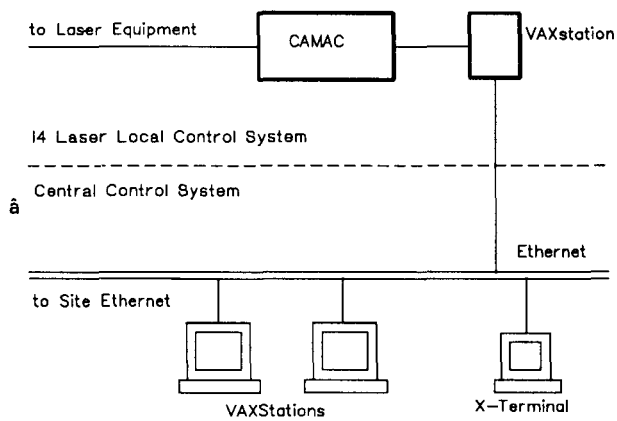

Fig. 3. Simplified Block Diagram of the I4 Laser Remote/Local Control System

3.1. Laser Operator Interface Evolution

Early developments used a VAXstation II/GPX, VWS and UIS. That software is easy to use but low level. It is slow to modify and does not allow remote operation but it allowed the requirements to be fulfilled. The decision was made later to use DECwindows/XUI to provide support for remote operation.

At the end of 1991, DECwindows/Motif superseded DECwindows/XUI and migration was done from XUI to Motif. The early XUI windows had been done at the UIL and Xt Intrinsics level while Motif windows used VUIT. In the near future all VWS/UIS will be phased out and Motif will be used for both local and remote operation.

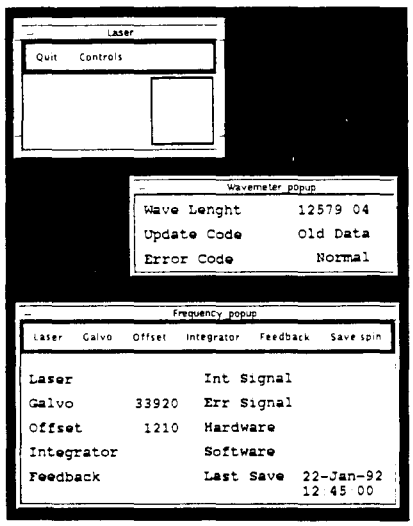

Fig. 4. Parts of the I4 Operator Interface.

3.2. Lessons Learned

When the decision to move from VWS/UIS to Decwindows/XUI was made, a re-evaluation of the old OPI was made by the software developers and the primary user. Several salient points came out of the discussions:

1. Make simplifying the window interface a high priority.
2. Minimize descriptive text and where possible put text on buttons, etc.
3. Remove diagnostic information from the top level interface.
4. Add Help via a pulldown menu.

Running both the VWS/UIS software locally for the end user and simultaneously running X for remote development has allowed the system to be used while the new development was underway. In addition, X allows multiple connections so more than one developer was able to be working on the new interfaces while the old system continued to run. The use of VUIT later on made the prototyping cycle between developer and end user much easier because changing the interface was then simpler than editting UIL code.

4. RF BOOSTER APPLICATION

The 160 KV, 92 MHz booster cavity is amplitude and phase regulated by a VME/PC-AT based local control system.[5] This control system is partitioned into three subsystems: digital, analog, and rf. It also includes automatic power-up sequencing, cavity conditioning and tuning functions. These and other parameters can be controlled locally through soft buttons and knobs in conjunction with a plasma discharge display. Some functions can be controlled remotely from Central Control System using a DECwindows/Motif based OPI.

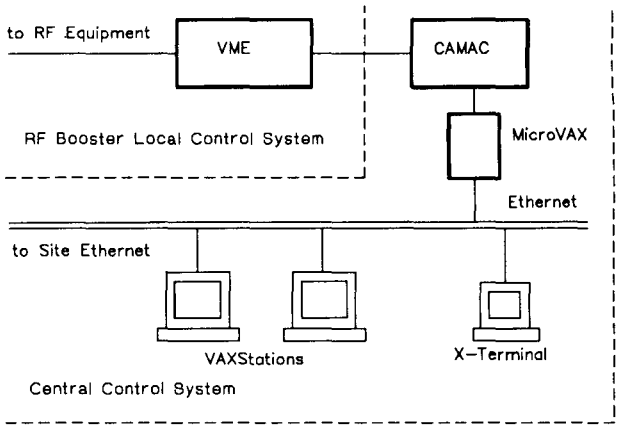

Fig. 5. Simplified Block Diagram of the RF Booster Local/Remote Control System.

Figure 5. shows the simplified block diagram of the RF Booster Control System. VAXes in the CCS communicate with the RF VME crate via a CAMAC-to-VME link.

4.1. RF Booster Operator Interface

The primary goal of the interface design was to create an operator interface that is consistent and easy to use. The main guidelines were as follows:

1. Adopt the operator perspective. Take operator's point of view. Involve operators and RF engineers in the design.
2. Give the operator control. Provide multiple ways for operators to access application functions. Use progressive disclosure.
3. Use real-world metaphors. Allow direct manipulation. Provide rapid response.
4. Keep the interface natural. Provide natural colors and screen elements' positions.
5. Keep the interface consistent. Keep intra-application and inter-application consistency.
6. Communicate application actions to the operator. Provide feedback, anticipate errors, provide warnings.

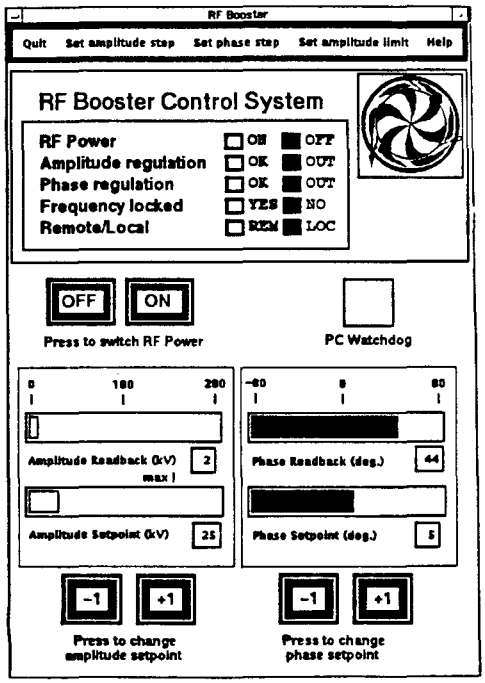

Fig. 6. RF Booster Operator Interface.

4.2. Lessons Learned

During the prototyping, a close liaison was maintained with the RF engineers and the cyclotron operators. A number of interesting requests came out of this work:

1. Push buttons were desired instead of sliders for varying the setpoints.
2. Binary status indicators were setup in groups so that abnormal conditions can be seen at a glance without having to read the specific text.
3. Some indication that the interface is active was included.
4. Setpoints and readbacks were paired as vectors with the same scaling so that abnormalities can be identified without having to read a textual number, just compare the vectors for equal length.

A beneficial point of doing development using an X based system is that most of the work, including the OPI, can be done from the sanctity of the developer's office with their local workstation instead of working in the often unpleasant environment of the laboratory's subsystems. This aspect is another area of increased efficiency.

5. CONCLUSIONS

Our early experience shows that developing an operator interface with a graphical interface builder such as VUIT is more productive than using the toolkit directly. Having some experience in using VUIT and having implemented callback routines, one can build the graphical user interface for a small (2-3 controlled object) application in an hour.

Motif and its environment has some favourable features, among them:

1. Documented standards from many sources.
2. Available from many vendors (it's open).
3. Interface builders available from many sources.
4. It's object oriented.
5. Multilayer implementation (layers of abstraction).
6. Separation of form and function.
7. Interoperability between hardware and software platforms.

Motif's problem areas seem to be the following:

1. Low level software is complex.
2. No specialized widgets for data visualization (graph, histogram, etc.).
3. No suitable means to display help information.
4. Strongly associated to C programming language, so other languages are not a good fit, in particular, the XUI to Motif porting tools do not support applications written in Fortran.

6. REFERENCES

1) McMinds, D.L., **Mastering OSF/Motif Widgets** (Addison-Wesley Publishing Company, Inc., Massachusetts, 1992), p.XXV.
2) **DECwindows/X11 Server for VWS Installation and User's Guide** (Digital Equipment Corporation, 1989), p.1-1.
3) **OSF/Motif Style Guide. Release 1.1** (Prentice Hall, New Jersy, 1991).
4) Levy, C.D.P., Burge, R., Dehaven, R., Mouat, M., Sarkar, S., Schmor, P.W., Wilkinson, N. and Zelenskii, A.N., "The laser system of the optically pumped ion source at TRIUMF," in **Proceedings of the International Workshop on Polarized Ion Sources and Polarized Gas Jets** (KEK Report 90-15, November 1990) pp.180-186.
5) Fong, K., Laverty, M., "RF control system for the TRIUMF booster cavity," presented at the EPAC Conference, Berlin, March 1992.