Migrating From X-Window Workstations To Windows NT PCs

David E. EISERT
Synchrotron Radiation Center, University of Wisconsin-Madison, 3731 Schneider Drive, Stoughton, WI 53589-3097, USA

The operator interface for the synchrotron storage ring Aladdin has evolved many times over the years to keep pace with new technology. Recently we have started a migration from VaxStation 3100s running VAX/VMS to PC clones running Windows NT. Although we could upgrade the existing workstations with the latest model, there are other reasons to change operating systems. The most important reason is the ability to integrate commercial PC applications such as LabView from National Instruments into the existing control system. In addition, the software development packages available under Windows NT are integrated more closely with the windowing environment than their VAX/VMS counterparts. Some of the advantages and disadvantages we have uncovered in porting existing code and in integrating commercial PC applications are discussed.

1. INTRODUCTION

The first version of the operator interface for the Aladdin control systems was implemented on color graphics terminals and a minicomputer [1]. As we acquired graphics workstations the interface remained largely unchanged for several reasons. Firstly, the operation staff were familiar with the existing interface and were reluctant to lose some of the features of our non-standard graphical user interface (GUI). Another problem developed due to the evolution of GUI environments; it was easier not to use any of the new features than to keep up with the fast changes in these environments. In addition, when we made our last major revision of the operator interface code, the workstations lacked graphical software development tools.

Over the years we have made some improvements to our operator interface software to increase efficiency when porting to a new GUI environment. The software has been consolidated from many medium sized applications to one large application and five small applications. Only two applications utilize the graphical display, the other applications perform various monitoring and file recording tasks in the background. The original software was written in FORTRAN and it was difficult to interface with GUI environments written in C. During the migration to X Windows the original FORTRAN code was rewritten in C. In addition, the parts of the software that use the display have been divided from sections that perform calculations and I/O operations.

There are several reasons to move from X Windows to Windows NT. Our staff has almost entirely converted over from the VAX/VMS systems to the PC platform. They wanted the ability to take work home and use the same computers at home as they do at work. They also wanted access to the many user-friendly PC applications. Afterwards they noticed that some of the PC applications would be very useful for machine studies and data manipulation. Another reason to switch platforms is the cost of color workstations compared to PC clones. The differences in the initial purchase price are narrowing rapidly but workstations will always be more expensive. The workstations also require a service contract since we can not afford to purchase spare workstations. Spare PC clones are available in emergency situations from almost every desktop and spare parts are available from local retailers.

2. DEVELOPMENT TOOLS

VAX/VMS has been a leading platform for software development for many years. The compilers have always been regarded as generating extremely efficient code. Unfortunately the development tools available under VMS never made the transition to the GUI environment. Software development tools always use a command line environment with each tool acting independently of the others. Compiler vendors created the Integrated Development Environment (IDE) under Windows/Windows NT. Many Windows compilers do not require the use of the IDE environment and at first people familiar with a command line environment will probably avoid using it. Eventually it becomes apparent that there are many advantages to working in an environment where all the development tools work together in one package.

2.1 Compilers

C++ compilers have been available for a long time on the PC platform but have only recently been offered by DEC. VAX C did not implement very strict type checking and allowed many practices that generate warning messages in current C++ compilers. More recently we started using the DEC C++ compiler but very few of the warning conditions were corrected since it has an option to implement the VAX C style type checking. When moving to a PC C++ compiler much of the type checking had to be turned off to avoid hundreds of warning messages.

Fortunately the PC C++ compilers have many options when it comes to warning messages, errors and optimizations. These options could be controlled by command line style switches but the IDE makes setting these switches easy through the use of a dialog box. Due to the IDE the whole development environment has many more options than is typically available with command line environments. In addition, GUI code normally has very large include files but PC C++ compilers precompile these include files. The next time a module is compiled, the include files do not have to be processed.

2.2 Debuggers

There are basically two stages of development for debugging software. During the first stage of debugging one would build the code with all the information needed for a source code debugger. The debugger is normally invoked and the code will run normally unless an error is encountered. When an error is encountered the debugger halts the execution of the code and enters the debugger. After the code is initially debugged a second release version of the code is ordinarily produced. The release version does not have all the information needed for a source level debugger. When a bug is encountered the system may produce one of several results. The worst case is that the system crashes without returning any information about the problem. In the best case only the program terminates and enough information is returned before it terminates to identify and correct the problem.

Both systems are fairly comparable when running under a debugger. The VAX/VMS Debugger supports a GUI-based debugger but lacks many features available in Windows NT. The VAX/VMS Debugger is primarily a command line debugger with an add on GUI interface. As a result the Windows NT version is a better match with the GUI interface. For example, the executing code is highlighted in the source code editor rather than just displayed in a special debugger window. The Windows NT version also allows the debugger to be invoked after an error is encountered. Normally the debugger has to be invoked before the program is started but under NT if an error is encountered the debugger can be started at the point the error occurred.

The early MS-DOS/Windows operating system lacked any memory protection from the released version of an application program. Normally every programming error encountered would produce an entire system crash with little information about the cause of the error. Windows NT has an extensive number of protection mechanisms for the operating system. Earlier windows code that took advantage of the lack of memory protection will fail under Windows NT. When a program running under Windows NT encounters a programming error a dialog box pops up explaining there was a fatal error. Some information on the location of the error can be obtained from the information reported in the dialog box. Unfortunately the Microsoft Visual C++ compiler normally does not include procedure names in the executable image, therefore only the address of the error can be reported. It should be noted that we have never crashed the Windows NT operating system due to an application programming error. Windows NT is able to recover all resources used by the errant program including any networking resources. VAX/VMS systems produce a program stack dump on fatal program errors with the procedure names and specific address information. VAX/VMS provides a slight advantage over Windows NT by providing the names of the procedures where the error occurred.

2.3 Object Editors

Early efforts to develop GUI code encountered difficulties when laying out the elements of a dialog box. The origin pixel coordinates and pixel dimensions of the elements had to be entered by hand in a text file. Later development tools were supplied that allowed graphical editing of the elements in dialog boxes. Both X Windows and Windows NT have similar utilities for creating and editing the GUI visual elements. Although the Windows NT GUI editor is built into the development package IDE interface, the X Windows editor is available only as a separate tool.

2.3 Other Tools

Some other tools that are useful during the software development process include intelligent program editors, software version control and code performance profilers. The early editors supplied basic text editing features such as text entry, navigation through text files, text searching and replacement. Later versions like the VAX/VMS Language Sensitive Editor included enhancements for checking code syntax prior to compilation. Editors available under Windows NT improves upon syntax checking with syntax highlighting. Code components like constants, code statements and comments are highlighted in different colors for easy identification. Under VAX/VMS we used the inherent software version control supplied by the VMS file system. Since VMS maintained revisions of source modules until they are explicitly purged, older versions could be easily recovered by deleting the more recent versions. At present the Windows NT file system does not support multiple file versions so we had to purchase a software version control utility. The version control software offered many more features than we required but some of these features prove to be quite useful. Rather than just keeping the last few revisions, the software version control keeps all version and can restore any previous version upon request. Although we had access to the VAX/VMS code profiler, we never used it because of the extra effort required. The Windows NT code profiler is enabled with a simple dialog check box. The ease of use makes the tool much more valuable in determining potential efficiency problems.

3. PORTING CODE

Most of our code compiled under Windows NT without problems but there were a few areas that required some effort to port. These include cursor control, graphical drawing and the network interface. At present we have ported approximately half of the operator interface code to Windows NT. Since our code relies on some non-standard GUI features we had to investigate ways to bypass some of the Windows NT GUI features. Fortunately Windows NT has available all the hooks needed by our software and after a little experimenting we were able to accomplish our tasks. In addition, we had previously used our own network protocol for communication but it was simpler to use UDP/IP that is supported by Windows NT than to add our network protocol to Windows NT.

3.1 Existing Code

Many facilities use external knob boxes or GUI-based sliders for virtual analog knob support under their software. We have used another method that I have not seen at any other facility. Our implementation uses the track ball that is normally used for cursor control in GUI environments. This is accomplish by confining the cursor to the main window, hiding the cursor from view and then placing the hidden cursor in the center of the screen. Every tenth of a second the cursor position is checked and changes in positions are translated to increments in the analog control. If the hidden cursor gets too close to a screen edge, the cursor is repositioned to the center of the screen. Standard GUI style guidelines discourage the “grabbing” of the cursor but all GUIs seem to provide hooks for these functions. Both X Windows and Windows NT allow the cursor to be bound to a particular window and to be hidden from view. One potential problem with this method involves child windows gaining control of the cursor when the hidden cursor is positioned over them. Our software does not use any child windows so the problem is avoided.

Our user interface appears much as it did when we used color graphic terminals. We use a blank graphic window to draw all the text with only a few lines highlighting the areas around the text. This simple format allows for very fast drawing requiring only a fraction of a second to draw a whole screen. Cursor handling becomes a little more complicated because all mouse events inside the window have to be interpreted by the program without the normal assistance provided by the GUI environment. This format also allows for very efficient updates of the device readings due to the minimal overhead imposed by directly drawing the text. Many of these speed enhancements were needed for the older, slower workstations but they have become less important for newer systems.

Under VAX/VMS we developed our own network protocol because it was simpler than writing a complete implementation of a standard network protocol on our VME systems. It is not as easy to implement a new network protocol under Windows NT. It appears that the protocols have to be installed as part of the protocol layer of the driver for the ethernet adapter. We did not want to write any code for Windows NT that would have to be installed as part of the operating system. So we had to approach the problem from the VME system end and write an implementation of UDP/IP for the VME systems [2]. Once the UDP/IP implementation was completed for the VME systems it was only the simple matter of using WinSock for Windows NT. WinSock is an extension of the standard UNIX Berkeley Sockets implemented under Windows NT.

3.2 Commercial Libraries

Our existing code contains some two-dimensional plotting routines needed to display plots of storage ring devices versus time. This plotting code would draw the plot in a empty window using only the text and line drawing functions of the GUI. In porting the code to Windows NT it became apparent that there are several low cost plotting packages that could be used for this function. Under VAX/VMS the plotting packages were so expensive we implemented our own simple plotting routines. By using these inexpensive packages we can create many styles of plots that we could not produce with our own simple plotting software.

4. COMMERCIAL SOFTWARE

One of the main disadvantages of using VAX/VMS systems is the lack of inexpensive commercial software packages. DEC did develop a licensing program that allowed educational institutions access to their code for a small annual fee. Unfortunately this did not reduce the cost of software from other commercial developers. For Windows NT we can purchase many commercial applications at a reduced educational price.

4.1 LabView

LabView is a commercial graphical data acquisition and control program developed by National Instruments that runs under many operating systems including Windows NT. The program offers the graphical control displays that are now common for GUI environments. It also has a graphical programming environment that does not require knowledge of traditional programming languages. This graphical programming environment appeals to our scientific and engineering staff as a simple tool that they can manipulate to accomplish many control related tasks without the need of a controls programmer. In practice there is a steep learning curve needed to work effectively in this environment. Even though the graphical programming environment hides the underlying programming language, knowledge of standard programming concepts are needed to develop the graphical code. At present our staff is reluctant to attempt to make their own changes to the graphical code but they should be able to make these changes after they have developed a little more experience.

The graphical programming environment has several disadvantages compared with traditional programming languages. The graphical language is not as complete as a traditional programming language. All working variables have to be created on the graphical control display and then marked as not visible. Simple functions that require only three or four lines in a traditional programming language can require a whole screen of the graphical programming language. The graphical programming language is best suited for simple data manipulation and display. Very impressive graphical displays can be created easily but much of the underlying data manipulation is best implemented in a traditional programming language. Fortunately there are several methods to link a traditional programming language with LabView, the simplest is through dynamic link libraries.

4.2 Dynamic Link Library

Dynamic link libraries (DLLs) are code and/or data that is not part of the original executable. The DLLs are loaded at run-time by the original executable. This method allows commercial software developers to let end users add custom software to the original commercial executable. Parts of our original control code can be made into a DLL and added to any Windows NT application that allows this method of extension.

4.3 Other Packages

LabView was the first Windows NT application that we thought would be useful in complementing our existing control software. Other packages that we could link to include spreadsheets, mathematical packages, and databases. Spreadsheets and many mathematical packages allow the end user to add functions through dynamic link libraries. At present we have not started to develop links to these applications but we will attempt to include them in future developments. We have started to use a Windows NT database to store device descriptions. Our VMS version was very outdated and needed extensive revisions. The entire device description database was moved to a Windows NT database in only a few days.

5. CONCLUSION

Changing operating systems and windowing systems involves many modifications to the original software. Normally we have only made extensive changes when the existing systems or software packages have become obsolete. The VMS operating system is not obsolete but it is clearly no longer the primary system used at our facility. Windows NT offers a very robust development environment and makes available many inexpensive commercial applications. The initial effort of porting the control software to Windows NT is justified by allowing us to utilize more commercial software. Future software development efforts will be in integrating and extending commercial packages rather than writing our own software.

This work was supported by the National Science Foundation. The current contract number is DMR-9212658.

References

[1] J. P. Stott and D. E. Eisert, CERN Yellow Report **90-08**, 103 (1990)

[2] D. E. Eisert, Fermilab Report, These Proceedings.