N91-24048

TRANSPORTABLE APPLICATIONS ENVIRONMENT (TAE) PLUS: A NASA USER INTERFACE DEVELOPMENT AND MANAGEMENT SYSTEM

Martha R. Szczur

NASA/Goddard Space Flight Center
Greenbelt, MD 20771 USA
mszczur@postman.gsfc.nasa.gov

ABSTRACT

The Transportable Applications Environment Plus (TAE™ Plus), developed at NASA's Goddard Space Flight Center, is a portable *What You See Is What You Get* (WYSIWYG) user interface development and management system. Its primary objective is to provide an integrated software environment that allows interactive prototyping and development of graphical user interfaces, as well as management of the user interface within the operational domain. TAE Plus is being applied to many types of applications, and this paper discusses what TAE Plus provides, how the implementation has utilized state-of-the-art technologies within graphic workstations, and how it has been used both within and outside NASA.

BACKGROUND

Emergence of graphical user interfaces

With the recent emergence of sophisticated graphic workstations and the subsequent demands for highly interactive systems, designing and developing good user interfaces has become more complex and difficult. Prior to the graphic workstations, the application developer was primarily concerned with developing user interfaces for a single monochrome 80x24 alphanumeric character screen with keyboard user entry. With high resolution bit-mapped workstations, the user interface designer has to be cognizant of multiple window displays, the use of color, graphical objects and icons, and various user selection techniques (e.g., mouse, trackball, tablets).

High resolution graphic workstations also provide system developers with the opportunity to rethink and redesign the user interfaces (UI) of their next generation applications. For instance, in a command and control environment, many processes run simultaneously to monitor a particular operation. With modern graphic workstations, time critical information concerning multiple events can be displayed concurrently on the same screen, organized into different windows in a variety of graphical and textual presentations. As today's workstations inspire more elaborate user interfaces, the applications which utilize their graphics capabilities increase in complexity. Prototyping different user interface designs, thus, becomes an increasingly important method for developing and communicating concepts and requirements for an application.

Role of prototyping at GSFC

Prototypes can be constructed with various levels of sophistication and fidelity. At their simplest they are visual mockups of the user interface. A prototype can also be a dynamic sequence of events, with simulated control between steps. They can even be a working model of a system, which can evolve into an operational system or be used for research purposes.

Within the government environment, prototypes also can play the important role of communicating specifications from government agency to the contractor, as well as to validate contractor interpretation and design approaches by reviews from the targeted government user. Prototyping key concepts and salient

features of proposed user interface standards, applied in typical operations scenarios, greatly enhances the users' ability to respond and have their concerns understood. Thus, including prototyping as a step in the application development cycle can ensure user acceptance of the final operational application.

Requirements for a prototyping-to-operational development environment

To support our development methodology we wanted to establish an integrated environment that allows prototyped user interfaces to evolve into operational applications. This environment would satisfy the following objectives:

- separate the user interface from the application,
- provide tools to allow interactive design/change/save of user interface elements,
- take advantage of the latest hardware technology,
- support rapid prototyping,
- manage the user interface,
- allow integrated management of multiple, asynchronously-active processes,
- develop tools for increasing application development productivity,
- provide the application with runtime services, and
- allow portability to different computing environments.

Building on existing technology

Many of these objectives were addressed in the early 1980's when GSFC recognized that most large-scale space applications, regardless of function, required software to support human-computer interactions and application management. This lead to the design and implementation of the Transportable Applications Executive (now, referred to as TAE *Classic*), which abstracts a common core of system service routines and user dialog techniques used by all applications [Ref. 1]. Over the years, TAE Classic has matured into a powerful tool for quickly and easily building and managing consistent, portable user inter-faces, but only for the standard alphanumeric terminal. When the requirement to support graphical user interfaces emerged, TAE Classic was examined as a potential *building block*. It was determined that it had a sufficiently flexible architecture and data structure to accommodate the extensions that would be needed to support user interface development within the graphic workstation environment.

WHAT DOES TAE PLUS PROVIDE?

To meet the defined goals, services and tools were developed for creating and managing window-oriented user interfaces. It became apparent, due to the flexibility and complexity of graphical user interfaces, that the design of the user interface should be considered a separate activity from the application program design. The interface designer can then incorporate human factors and graphic art techniques into the user interface design. The application programmer needs only to be concerned about what results are returned by the user interaction and not the look of the user interface.

In support of the user interface designer, an interactive *WorkBench* application was implemented for manipulating interaction objects ranging from simple buttons to complex multi-object panels. As illustrated in Figure 1, after designing the screen display, the WorkBench saves the specification of the user interface in resource files, which can then be accessed by application programmers through a set of runtime services, Window Programming Tools (WPTs). Guided by the information in the resource files, the routines handle all user interactions. The WPTs utilize the standard MIT X Window System™ to communicate with the graphic workstations. [Ref. 2] As a further aid to the UI developer, the WorkBench provides an option to generate the source code which will display and manage the designed user interface. This gives the programmer a working template into which application-specific code can be added.

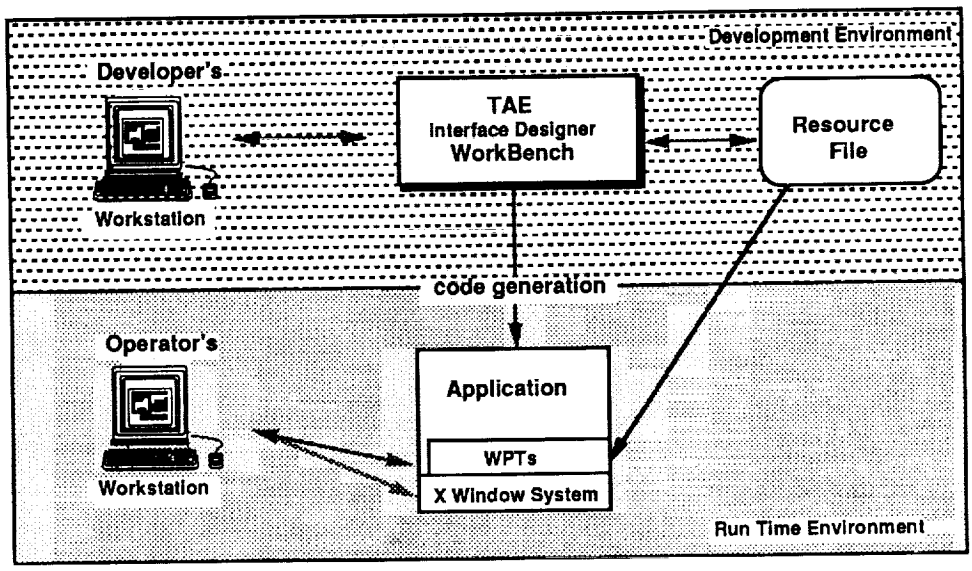

Figure 1. TAE Structure

INTERACTION OBJECTS AS BUILDING BLOCKS

The basic building blocks for developing an application's graphical user interface are a set of interaction objects. All visually distinct elements of a display that are created and managed using TAE Plus are considered to be interaction objects and they fall into three categories: user-entry objects, information objects, and data-driven objects. *User-entry objects* are mechanisms by which an application can acquire information and directives from the end user. They include radio buttons, check boxes, text entry fields, scrolling text lists, pulldown menus and push buttons. *Information objects* are used by an application to instruct or notify the user, such as contextual on-line help information displayed in a scrollable static text object or brief status error messages displayed in a bother box. *Data-driven objects* are vector-drawn graphic objects

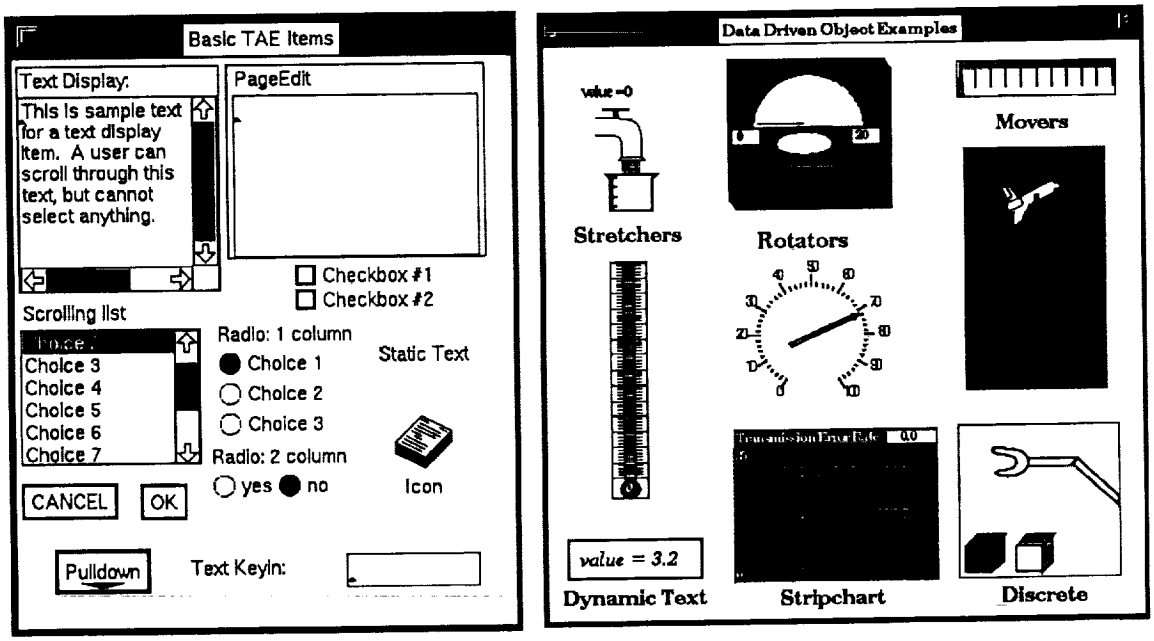

Figure 2. TAE Plus User Interface Interaction Objects

which are linked to an application data variable; elements of their view change as the data values change. Examples are dials, thermometers, and strip charts. When creating user dialogues, these objects are grouped and arranged within *panels* (i.e., windows) in the WorkBench.

The use of interaction objects offers the application designer/programmer a number of benefits with the expected payoff of an increase in programmer productivity. The interaction objects provide a consistent look and feel for the application's user interface, which translates into reduced end-user training time, more attractive screens, and an application which is easier to use. Another key benefit is that since the interaction objects have been thoroughly tested and debugged, the programmer is able to spend more time testing the application and less time verifying that the user interface behaves correctly. This is particularly important considering the complexity of some of the objects, and the programming effort it would take to code them from scratch. Refer to Figure 2 for a sample of the TAE Plus interaction objects.

TAE PLUS WORKBENCH

The WorkBench provides an intuitive environment for defining, testing, and communicating the look and feel of an application system. Functionally, the WorkBench allows an application designer to dynamically lay out an application screen, defining its static and dynamic areas. The tool provides the designer with a choice of pre-designed interaction objects and allows for tailoring, combining and rearranging of the objects. To begin the session, the designer needs to create the base panel (i.e., window) into which interaction objects will be specified. The designer specifies presentation information, such as the title, font, color, and optional on-line help for the panel being created. The designer defines both the presentation information and the context information of all interaction items to reside in the panel by using the item specification window (refer to Figure 3). For icon support, the WorkBench has an icon

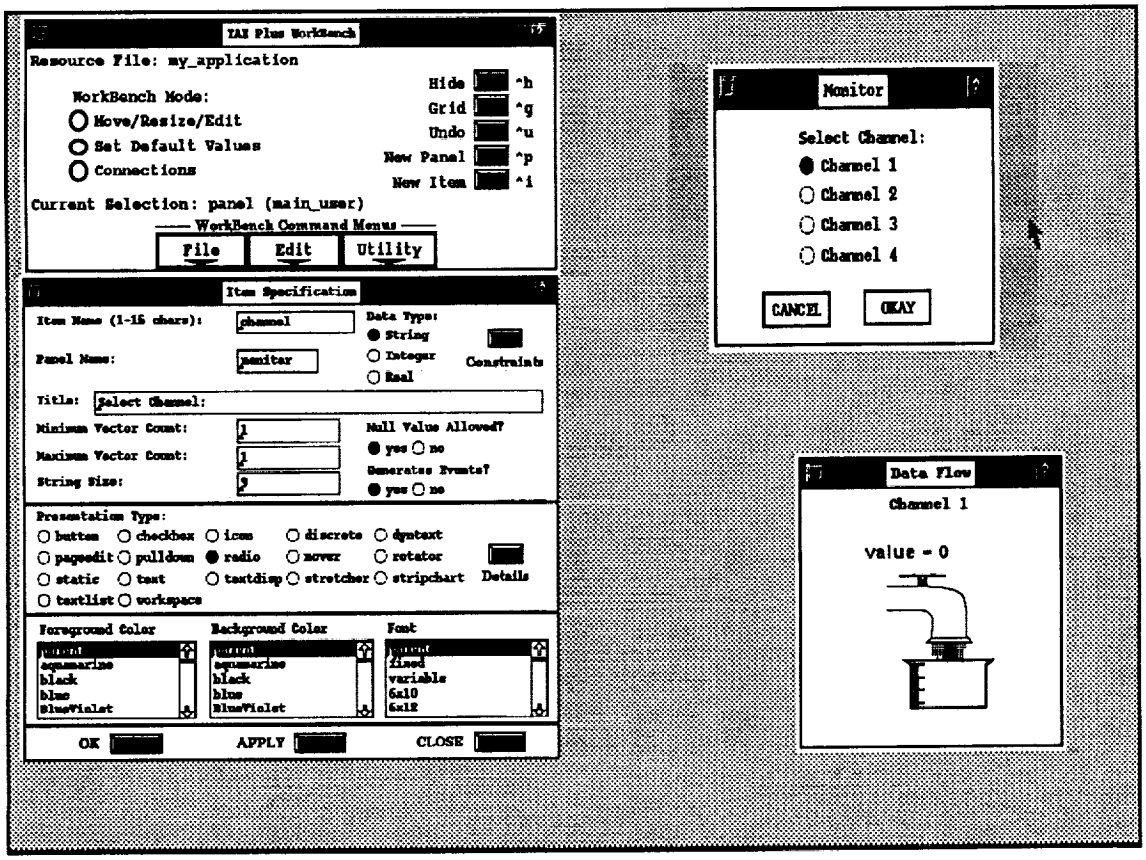

Figure 3. Building a user interface with the WorkBench

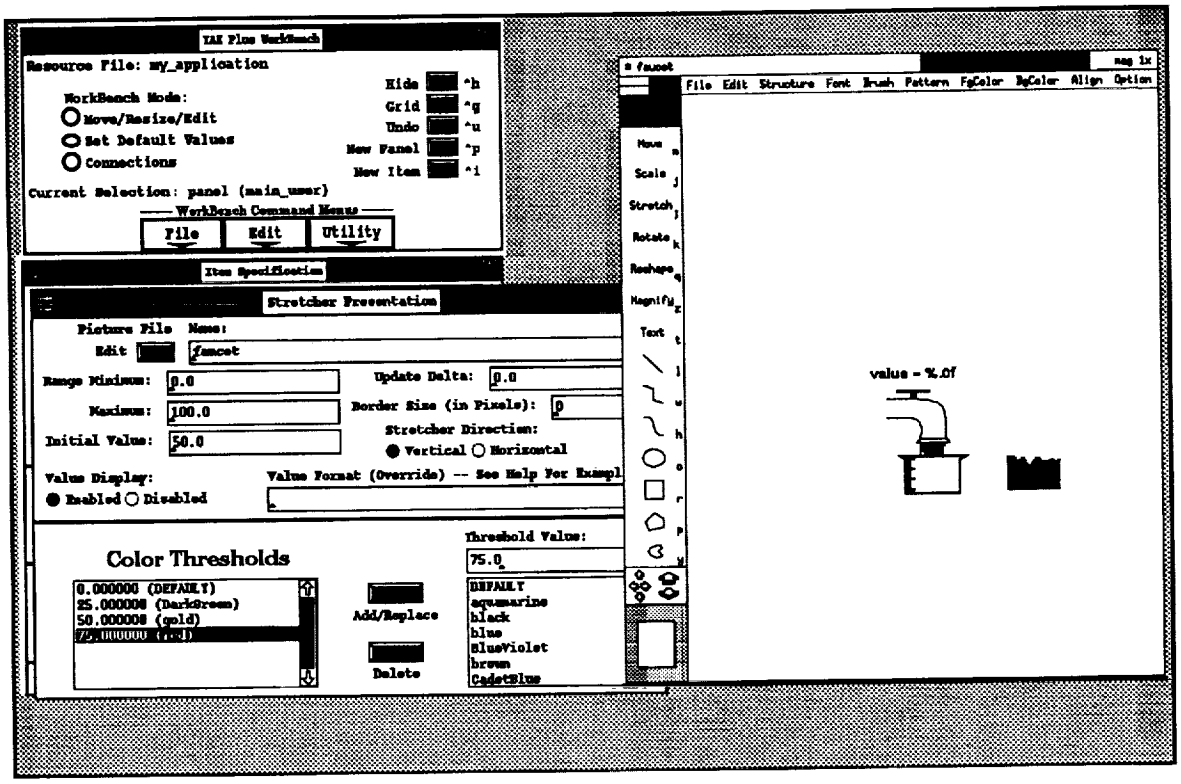

Figure 4. Creating a stretcher data-driven object

editor, within which an icon can be drawn, edited and saved. As the UI designer moves, resizes, and alters any of the item's attributes, the changes are dynamically reflected on the display screen.

The designer also has the option of retrieving *palettes* of previously created items. The ability to reuse interaction objects saves programming time, facilitates experimenting with different combinations of items in the prototyping process, and contributes to standardization of the application's look and feel. If an application system manager wanted to ensure consistency and uniformity across an entire application's UI, all developers could be instructed to use only items from the application's palette of common items.

When creating a data-driven object, the designer goes through a similar process by setting the associated attributes (e.g., color thresholds, maximum, minimum, update delta) in the specification panels. To create the associated graphics drawing, the WorkBench provides a drawing tool within which the static background and dynamic foreground of a data-driven object can be drawn, edited, and saved. Figure 4 shows the drawing tool being used to create a *stretcher* data-driven object.

Most often an application's UI will be made up of a number of related panels, sequenced in a meaningful fashion. Through the WorkBench, the designer defines the interface *connections*. These links determine what happens when the user selects a button or a menu entry. The designer attaches *events* to interaction items and thereby designates what panel appears and/or what program executes when an event is triggered. Events are triggered by user-controlled I/O peripherals (e.g., point and click devices or keyboard input).

TAE Plus also offers an optional help feature which provides a consistent mechanism for supplying application-specific information about a panel and any interaction items within the panel. In a typical session, the designer elects to edit a help file after all the panel items have been designed. Clicking on the edit help option in the Panel Specification Panel brings up a text editor window in which the appropriate information can be entered. The designer can then define any button item or icon item to be the help item for the panel (in this scenario it would be the help icon in the panel "Monitor"). During the application operation, when

the end-user clicks on the question mark item, the cursor changes to a question mark symbol (?). The end-user then clicks on the panel itself or any item in the panel to bring up a help panel containing the associated help text.

Having designed the layout of panels and their attendant items and having threaded the panel and items according to their interaction scenario, the designer is able to preview (i.e., rehearse) the interface's operation from the WorkBench. With this potential to test drive an interface, to make changes, and to test again, iterative design becomes part of the prototyping process. With the rehearsal feature, the designer can evaluate and refine both the functionality and the aesthetics of a proposed interface. After the rehearsal, control is returned to wherever the designer left off in the WorkBench and the designer can either continue with the design process or save the defined UI in a resource file.

Developing software with sophisticated user interfaces is a complex process, mandating the support of varied talents, including human factors experts and application program specialists. Once the UI designer (who may have limited experience with actual code development) has finished the UI, he/she can turn the saved UI resource file over to an experienced programmer. As a further aid to the application programmer, the WorkBench has a "generate" feature, which produces a fully annotated and operational body of code which will display and manage the entire WorkBench-designed UI. Currently, source code generation of C, Ada, FORTRAN and TCL are supported, with bindings for C++ expected in a future release of TAE Plus. The programmer can now add additional code to this template and make a fully functional application. Providing these code stubs helps in establishing uniform programming method and style across large applications or within a family of interrelated software applications.

WINDOW PROGRAMMING TOOLS (WPTs)

The Window Programming Tools (WPTs) are a package of application program callable subroutines used to control an application's user interface. Using these routines, applications can define, display, receive information from, update and/or delete TAE Plus panels and interaction objects. WPTs support a modeless user interface, meaning a user can interact with one of a number of interaction objects within any one of a number of displayed panels. In contrast to sequential mode-oriented programming, modeless programming accepts, at any instance, a number of user inputs, or *events*. Because these multiple events must be handled by the application program, event-driven programming can be more complex than traditional programming. The WorkBench's auto-generation of the WPT event loop reduces the risk of programmer error within the UI portion of an application's implementation.

Wpt_AddEvent	Add other sources for input/output/exception
Wpt_BeginWait	Display busy indicator cursor
Wpt_CloseItems	Close items on a Panel
Wpt_ConvertName	Get the X Id of a named window
Wpt_Endwait	Stop displaying busy indicator cursor
Wpt_Init	Initializes interface to X Window System
Wpt_ItemWindow	Gets the window Id of the window containing a parameter
Wpt_MissingVal	Indicates if any values are missing
Wpt_New Panel	Displays a user interface panel
Wpt_NextEvent	Gets next panel-related event
Wpt_PanelErase	Erases the displayed panel from the screen
Wpt_PanelMessage	Displays message in "Bother Box"
Wpt_PanelReset	Resets object values to initial values
Wpt_PanelTopWindow	Gets panel's parent shell window Id
Wpt_PanelWidgetId	Return the Widget Id of a Wpt Panel Widget
Wpt_PanelWindow	Returns the X Id of a panel
Wpt_ParmReject	Generates a rejection message for a given value
Wpt_ParmUpdate	Updates the displayed values of an object
Wpt_Pending	Check if a WptEvent is pending from X, Parm or file.
Wpt_RemoveEvent	Remove a previously registered event
Wpt_SetTimeOut	Set/Cancel timeout for gathering Wpt events.
Wpt_ViewUpdate	Updates the view of a parameter on a displayed panel

Figure 5. The Window Programming Tools (WPTs)

As mentioned earlier, the WPT package utilizes the MIT X Window System as its base windowing system. One of the strengths of X is the concept of providing a low-level abstraction of windowing support (Xlib), which becomes the base standard, and a high-level abstraction (X toolkits), which has a set of interaction objects (called "widgets" in the X world) that define elements of a UI's look and feel. The current version

of TAE Plus (V4.1) operates with the X11R3 and X11R4 version using the X Toolkit and HP widget set delivered with the X software. Due to the growing acceptance of the Open Software Foundation's Motif ™ user interface style as a defacto industry standard, the next release of TAE Plus (V5.0) will be based on the Motif software.

The WPTs also provide a buffer between the application program and the X Window System services. For instance, to display a WorkBench-designed panel, an application makes a single call to Wpt_NewPanel (using the *panel name* specified in the WorkBench). This single call translates into a function that can make as many as 50 calls to X Window System routines. For the majority of applications, the WPT services and objects supported by the WorkBench provide the necessary user interface tools and save the programmer from having to learn the complexities of programming directly with X. This can be a significant advantage, especially when considering the learning curve differential between 26 WPT routines versus over 400 X Toolkit intrinsics and over 200 Xlib services. Refer to Figure 5 for a sample list of the WPTs.

IMPLEMENTATION

The TAE Plus architecture is based on a separation of the user interaction management from the application-specific software. The current implementation is a result of having gone through several prototyped and beta versions of a WorkBench and user interface support services during the 1986-89 period, as well as building on the TAE Classic structure.

The "Classic" portion of the TAE Plus code (≈ 60,000 LOC) is implemented in the C programming language. In selecting a language for the WorkBench and the WPT runtime services, we felt a "true" object-oriented language would provide us with the optimum environment for implementing the TAE Plus graphical user interface capabilities. (See Chapter 9 of Cox [Ref. 4] for a discussion on the suitability of object-oriented languages for graphical user interfaces.) We selected C++ [Ref. 5] as our implementation language for several reasons [Ref. 6]. For one, C++ is becoming increasingly popular within the object-oriented programming community. Another strong argument for using C++ was the availability of existing, public domain, X-based object class libraries. Utilizing an existing object library is not only a cost saver, but also serves as a learning tool, both for object-oriented programming and for C++. Delivered with the X Window System is the *InterViews* C++ class library and a drawing utility, *idraw*, both of which were developed at Stanford University. [Ref. 7] The *idraw* utility is a sophisticated direct manipulation C++ application, which we integrated into the WorkBench to support creating, editing and saving the graphical data-driven interaction objects.

PORTABILITY ISSUES

Throughout the design and development of TAE Plus, one of our primary goals has been to be "portable" over a wide range of hardware platforms. It is a requirement that TAE Plus operate on various UNIX systems and VAX/VMS. There are three primary software areas identified to be the most nonportable -- file manipulation, process control and interprocess communication. The software modules to support these areas are localized and tailored to the individual operating system of each host environment. With the proper use of tailored *include* files, TAE Plus ports between workstations with few problems.

When porting among different hardware platforms, the host system's method of storing binary information is always of key concern. Since TAE Plus's resource files are binary, we provide a utility to produce a straight ASCII equivalent of the file. This file can then be transferred to any platform that accepts the ASCII character set and then converted back into a binary file, which can be read by TAE Plus applications (including the WorkBench) operating on the target platform.

As mentioned earlier in this paper, the C language was selected for implementing TAE Classic and it has proven to be an efficient and standard language across different hardware platforms. The C++ code has proven to be less portable than anticipated. There are several differences, even syntactical, among the

various C++ compilers. Therefore it was decided to initially limit our support to just two compilers. They are the GNU C++ and the OASYS Designer C++ compiler. With the recent release of the AT&T V2.0 C++ compiler, C++ is becoming more standardized and the compiler issue is expected to dissipate as vendor's offer C++ as one of their standard language compilers.

The single most important factor contributing to the portability of TAE Plus is the X Window System. Generally, if a graphic workstation supports the Xlib and X Toolkit and operates either UNIX or VMS, TAE Plus can be ported to it with reasonable ease.

AVAILABILITY AND USER SUPPORT

After two years of prototyping and developing beta versions of the TAE Plus, an *industrial strength* version of TAE Plus (Version 4.1) was released in February 1990. It is available for public distribution, at a minimal license fee, from the Center of Software Management and Information Center (COSMIC), a NASA distribution center. While TAE Plus base development and testing is done on a Sun workstation under UNIX within the R&D laboratory at GSFC, TAE Plus is also ported and validated with formal acceptance testing on the following UNIX workstations: Apollo, Vaxstation II , Decstation 3100, HP9000, and Macintosh II (A/UX). TAE Plus is also available and validated on the Vaxstation II under VMS and DECWindows™. Other user sites have successfully installed TAE Plus onto the Masscomp, Silicon Graphics Iris and other Unix-based graphic workstations. In January 1991, a beta release of TAE Plus 5.0, which uses the latest version of OSF Motif™ (V1.1), will be available to licensed TAE users on the Sun workstations. In subsequent months, ports to other workstations will become available. There are plans to port TAE Plus to new architectures, including the new IBM 6000 workstation and the 386i class of workstations.

Since the first release of TAE Classic in 1981, we have provided user support through a fully staffed TAE Support Office (TSO). This service has been one of the primary reasons for the success of TAE. Through the TSO, users receive answers to technical questions, report problems, and make suggestions for improvements. In turn, the TSO keeps users up-to-date on new releases, publishes a newsletter, and sponsors user workshops and conferences. This exchange of information enables the Project Office to keep the TAE software and documentation up-to-date and, perhaps most importantly, take advantage of user feedback to help direct future development.

APPLICATIONS USING TAE PLUS

Since 1982 over 750 sites have installed TAE Classic and/or TAE Plus. The applications built or being built with TAE perform a variety of different functions. TAE Classic usage was primarily used for building and managing large scientific data analysis and data base systems (e.g., NASA's Land Analysis System (LAS), Atmospheric and Oceanographic Information Processing System (AOIPS), and JPL's Multimission Image Processing Laboratory (MIPL) system.) Within the NASA community, TAE Plus is also used for scientific analysis applications, but the heaviest concentration of user applications has shifted to support of realtime control and processing applications. This includes supporting satellite data capture and processing, monitor and control of spacecraft and science instruments, prototyping user interface of the Space Station Freedom crew workstations and supporting diagnostic display windows for realtime control systems in ground operations. For these types of applications, TAE Plus is principally used to design and manage the user interface, which is made up of a combination of user entry and data-driven interaction objects. TAE Plus becomes a part of the development life cycle as projects use TAE Plus to prototype the initial user interface design and have this designed user interface evolve into the operational UI.

Outside the NASA community, TAE Plus is being used by an assortment of other government agencies (22%), universities (15%), and private industries (35%). Within the government sector, users range from the National Center for Atmospheric Research, National Oceanographic and Atmospheric Adminstration, U.S. Geological and EROS Data Center, who are developing scientific analysis, image mapping and data

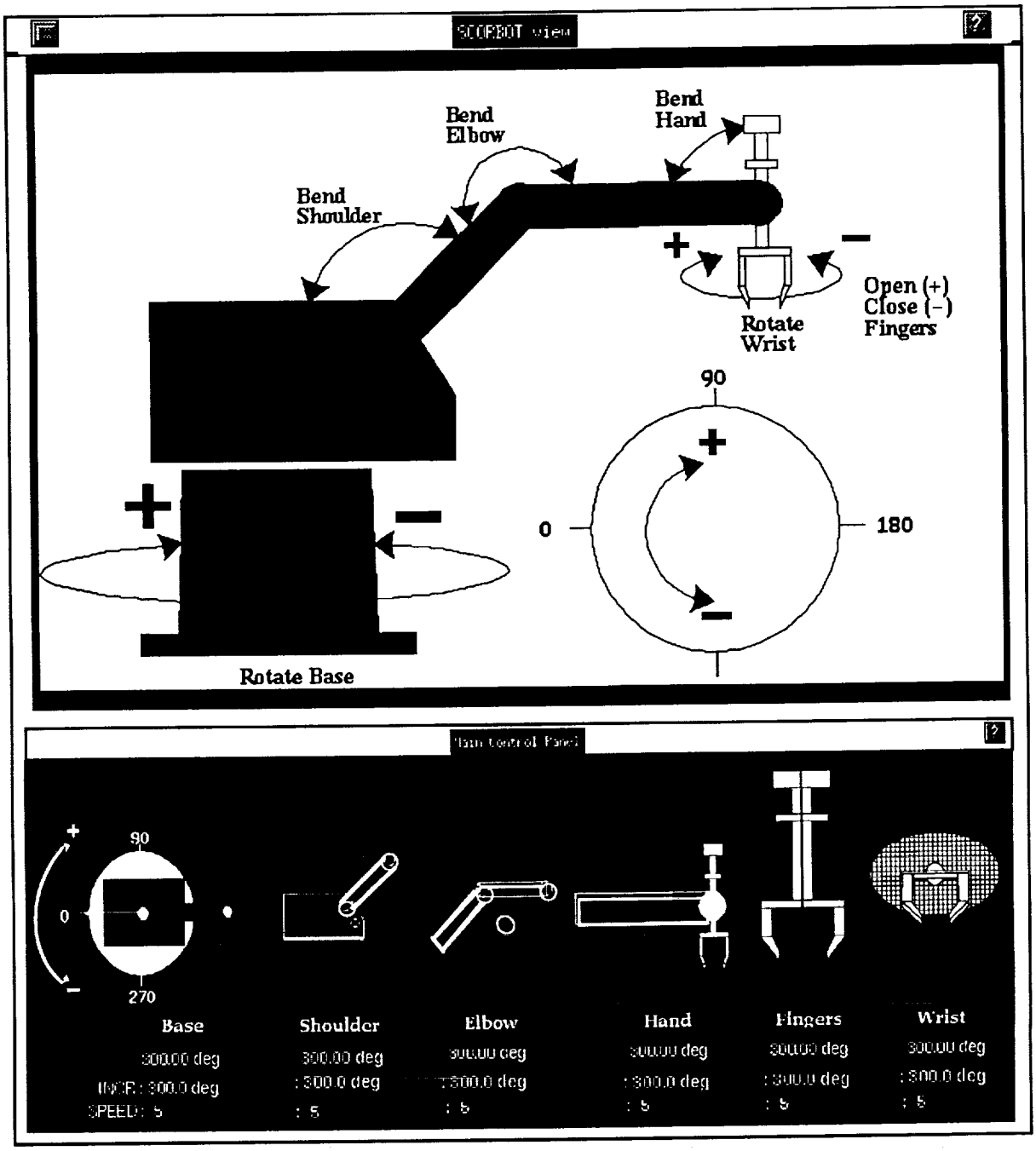

Figure 6.
Robot control panel developed at University of Colorado

distribution systems, to numerous Department of Defense laboratories, who are building command-and-control-related systems. Universities represented among the TAE community include CalTech, Cornell, Georgia Tech, MIT, Stanford, University of Maryland and University of Colorado. Applications being developed by University of Colorado include the Operations and Science Instrument Support System (OASIS), which monitors and controls spacecraft and science instruments and a robotics testbed for research intor the problems of construction and assembly in space. [Ref. 8] Figure 6 shows a view of the robot arm, built with TAE Plus. As the current location of the robot changes, the data-driven objects change respectively. Private industry has been a large consumer of the TAE technology and a sample of the companies that have received TAE Plus V4.1 include Apple Computer Inc., Ford Aerospace, Martin

Marietta, Computer Sciences Corp., TRW, Lockheed, IBM, Northern Telecom, Mitre Corp., General Dynamics and GTE Government Systems. These companies are using TAE Plus for an assortment of applications, ranging from a front-end for a corporate database to advanced network control center. Northern Telecom, Inc. used TAE Plus to develop a technical assistance service application which enables users to easily access a variety of applications residing on a network of heterogeneous host computers. [Ref. 9] Because of the high cost associated with programming and software-development, more and more software development groups are looking for easy-to-use productivity tools, and TAE Plus is becoming recognized as a viable tool for developing an application's user interface.

NEXT STEPS

The current TAE Plus provides a useful tool within the user interface development environment -- from the initial design phases of a highly interactive prototype to the fully operational application package. However, there are many enhancements and new capabilities that will be added to TAE Plus in future releases.

In the near term, the emphasis will be on enhancement features and upgrades, with the support for the Open Software Foundation's (OSF) Motif™ style and optimizing the TAE Plus software to improve real-time performance being of highest priority. All the requested enhancements are user-driven, based on actual experience using TAE Plus, or requirement-driven based on an application's design. For example, on the enhancements list are extensions to the interaction objects, (e.g., graph data-driven object, form fill-in), support for importing foreign graphics, and refinements in the code generation feature.

Future advancements include expanding the scope of the Transportable Applications Environment (TAE) to include new tools or technologies. For instance, the introduction of hypermedia technology and the integration of expert system technology to aid in making user interface design decisions are targeted for investigation and prototyping.

CONCLUSION

With the emergence of sophisticated graphic workstations and the subsequent demands for highly interactive systems, the user interface becomes more complex and includes multiple window displays, the use of color, graphical objects and icons, and various selection techniques. Prototyping of different user interface designs, thus, becomes an increasingly important method for stabilizing concepts and requirements for an application. At GSFC, the TAE Plus development team had the requirement to provide a tool for prototyping a visual representation of a user interface, as well as to establish an integrated development environment that allows prototyped user interfaces to evolve into operational applications. TAE Plus is fulfilling this role by providing a usable, generalized, portable and maintainable package of development tools.

TAE Plus is an evolving system, and its development will continue to be guided by user-defined requirements. To date, each phase of TAE Plus's evolution has taken into account advances in virtual operating systems, human factors research, command language design, standardization efforts and software portability. With TAE Plus's flexibility and functionality, it can contribute both more advances and more standardization in user interface development system technology.

ACKNOWLEDGEMENTS

TAE Plus is a NASA software product being developed by the NASA/Goddard Space Flight Center with contract support by Century Computing, Inc. The work is sponsored by the NASA Office of Space Operations.

TAE is a registered trademark of National Aeronautics and Space Administration (NASA). It is distributed through NASA's distribution center, COSMIC, (404) 542-3265. For further information, contact COSMIC and/or the TAE Support Office at GSFC, (301) 286-6034.

REFERENCES

1. Perkins, D.C., Howell, D.R., Szczur, M.R., "The Transportable Applications Executive -- an interactive design-to- production development system," *Digital Image Processing In Remote Sensing*, edited by J-P Muller, Taylor & Francis Publishers, London, 1988.

2. Scheifler, Robert W., Gettys, Jim., "The X Window System," MIT Laboratory for Computer Science, Cambridge, MA, October 1986.

3. Open Software Foundation, Inc., *OSF/Motif™ Programmer's Reference Manual*, Revision 1.1, 1990

4. Cox, Brad J., *Object Oriented Programming, An Evolutionary Approach*, Addison-Wesley Publishing Company, Reading, MA, 1986.

5. Stroustrup, Bjarne, *The C++ Programming Language*, Addison-Wesley Publishing Company, Reading, MA, 1987.

6. Szczur, Martha R., Miller, Philip, "Transportable Applications Environment (TAE) Plus: Experiences in 'Object'ively Modernizing a User Interface Environment," Proceedings of the OOPSLA Conference, September 1988.

7. Linton, Mark A., Vlissides, John M., Calder, Paul R., "Composing User Interfaces with Interviews," *IEEE Computer*, February, 1989.

8. Klemp, Marjorie, "TAE Plus in a Command and Control Environment", Proceedings of theTAE Eighth Users' Conference, June, 1990

9. Sharma, Alok, et al., "The TAS Workcenter: An Application Created with TAE", Proceedings of the TAE Eighth Users' Conference, June, 1990