TRANSPORTABLE APPLICATIONS ENVIRONMENT (TAE) PLUS

a NASA tool for Building and Managing Graphical User Interfaces

Martha R. Szczur
NASA/Goddard Space Flight Center
Greenbelt, MD 20771 USA
mszczur@postman.gsfc.nasa.gov
301 286-8609

ABSTRACT

The Transportable Applications Environment (TAE) Plus, developed at NASA's Goddard Space Flight Center, is an advanced portable user interface development environment which simplifies the process of creating and managing complex application graphical user interfaces (GUIs). TAE Plus supports the rapid prototyping of GUIs and allows applications to be ported easily between different platforms. This paper will discuss the capabilities of the TAE Plus tool, and how it makes the job of designing and developing GUIs easier for application developers. TAE Plus is being applied to many types of applications, and this paper discusses how it has been used both within and outside NASA.

BACKGROUND

With the emergence of low-cost graphic workstations and the subsequent demands for highly interactive systems, the design and develop of the user interface software has become more complex and difficult. With high resolution workstations, the user interface designer has to be cognizant of multiple window displays and asynchronous events from users and windowing systems, the use of color, graphical interaction objects and icons, and various user selection techniques (e.g., mouse, trackball, tablets).

To make user interfaces easier to create, many different types of tools have been developed. The X Window System™[2] has had a major impact on the user interface software in the UNIX and VMS-based workstation environments, and "X" has become the standard windowing system across these platforms. To make the task of programming the user interface using the X Window System easier, *toolkits* have evolve, which provide higher level services to the programmer along with a set of interaction objects (e.g., menus, buttons, scroll bars). Using the toolkit services, programmers can configure the objects to their specification. The most common toolkit in the UNIX and VMS workstation environment is the Open Software Foundation's Motif™ toolkit. Although X and Motif provide programmatic tools to aid the programmer, they are very complex to learn and do not offer any productivity advantages.

The tools that hold the most promise of dramatic productivity gains for user interface developers are What-You-See-Is-What-You-Get (WYSIWYG) user interface design and management tools. These tools allow you to directly layout your user interface, rehearse the graphical user interface (GUI), and even generate the source code, which will manage your application's user interface during operation.

During the evolution of the various GUI technologies, the Data Systems Technology Division at Goddard Space Flight Center built a user interface development environment, called the Transportable Applications Environment (TAE) Plus. This software tool has been built with the objective of providing a stable environment to support our on-going and diverse development efforts. Our two primary objectives were (1) to improve productivity of application user interface development and (2) to provide a buffer from technology changes.

To improve productivity we defined the following goals:

- support WYSIWYG design of the GUI objects
- support evolution from rapid prototype to baseline operational system
- provide reusable software components
- provide less complex set of application services
- support user interface experts, who may not be programmers

To protect our investments in the development of large-scale space applications that have a long "lifespan", we wanted to provide a mechanism that would allow GUI technology changes to be integrated into the systems with

minimal impact on the application-specific software. To provide this "change" buffer, we defined the following goals:

- separate the GUI definition from the application
- provide application programs with toolkit-independent runtime services
- support portability of applications across workstations (e.g., UNIX, VMS)

Elements of these goals 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 [1]. Over the years, TAE Classic matured into a powerful tool for quickly and easily building and managing consistent, portable user interfaces, but only for the standard alphanumeric terminal.

We took advantage of the lessons learned in the TAE Classic development when we decided to support the GUI environment. By utilizing some of the internal data structures and features of the original TAE software, we developed a set of tools which support the building and management of graphical user interfaces. This advanced version of TAE is called TAE Plus (i.e., TAE *plus* graphics support).

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 only needs 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 Open Software Foundation's Motif™ and the standard MIT X Window System™ to communicate with the graphic workstations.[2] As a further aid to the UI developer, the WorkBench provides an option to generate the source code (C, C++ or Ada) which will display and manage the designed user interface during runtime. This gives the programmer a working template into which application-specific code can be added.

INTERACTION OBJECTS AS BUILDING BLOCKS

The basic building blocks for developing an application's GUI 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: selection objects, text objects, and data-driven objects. Selection objects are mechanisms by which an application can acquire directives from the end user. They include menu bar with cascading menus, radio buttons, check boxes, scrolling selection list, icon button, option menu, scale (slider) pulldown menus and push buttons. Text objects are used by an application to request text information or to instruct or to notify the user. They include keyin, optimized dynamic text object that is updated dynamically by the application, label, multi-line edit and text displays (e.g., message, status, help). Data-driven objects are vector-drawn graphic 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, any of these objects can be grouped and arranged within panels (i.e., windows) in the WorkBench. There is also support for a X Workspace into which applications can write directly using X Window services. Refer to Figure 2 for a sample of the TAE Plus interaction objects.

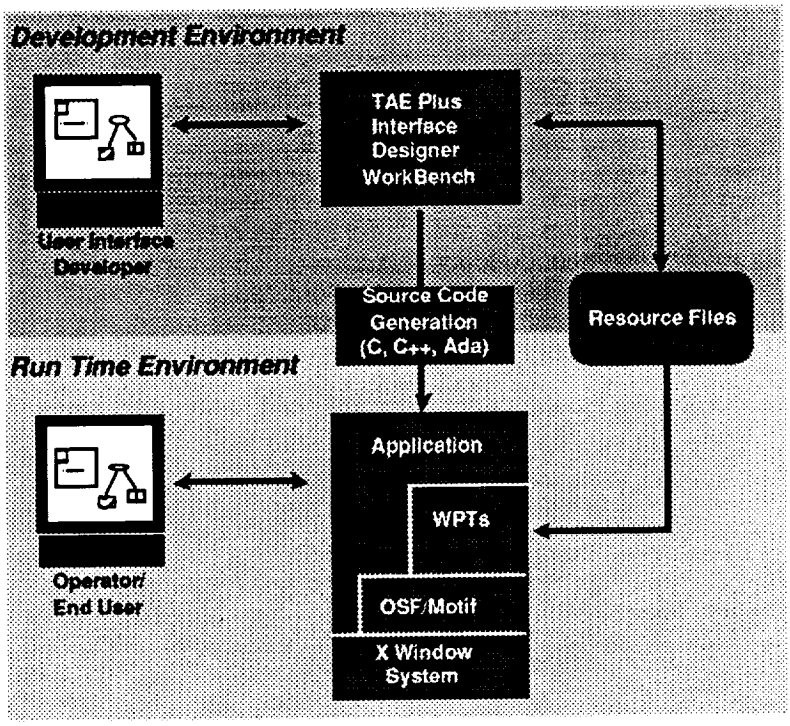


Figure 1: TAE Plus Structure

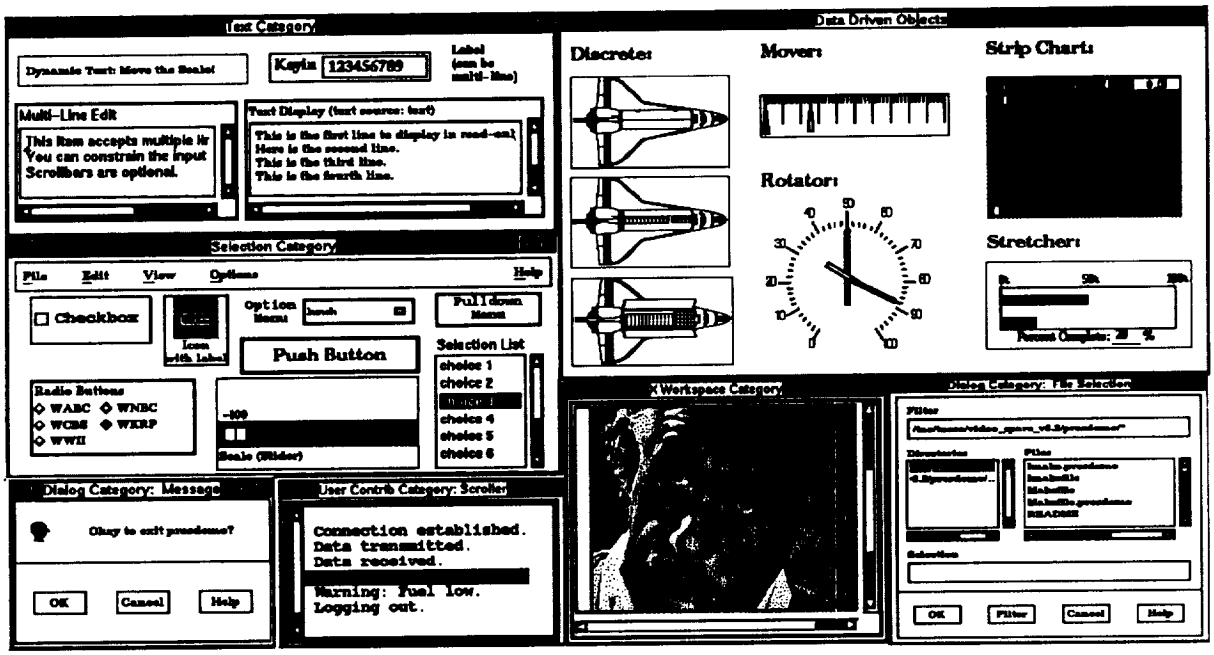
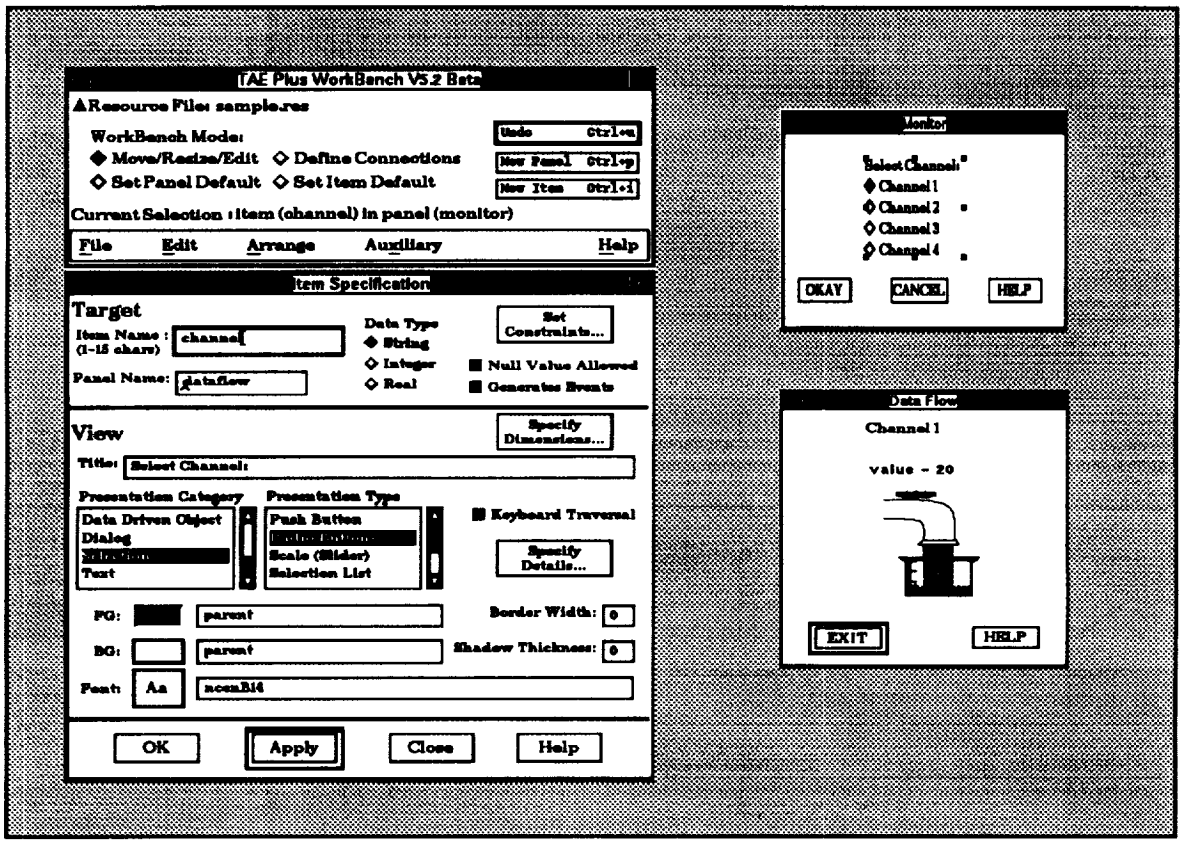


Figure 2: TAE Plus Interaction Objects

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 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 wants 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 foregrounds of a data-driven object can be drawn, edited, and saved.

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 action executes when an event is triggered. Events are triggered by user-controlled I/O peripherals (e.g., point and click devices or keyboard input).

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.

As a further aid to the application developer, 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, C++, Ada and the TAE Command Language (TCL) (an interpreted prototyping language) are supported. Providing this code template 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. The WPT package utilizes the the MIT X Window System, as its standard windowing system and the Motif toolkit and window manager.

The WPTs provide a buffer between the application program and the Motif toolkit. 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 Motif library 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 Motif and X. This can be a significant advantage, especially when considering the learning curve differential between 40 WPT routines versus over 400 X Toolkit intrinsics and over 200 Xlib services. Refer to Figure 4 for a sample list of the WPTs.

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 4: Sample List of Window Programming Tools (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 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 [4] for a discussion on the suitability of object-oriented languages for graphical user interfaces.) We selected C++ [5] as our implementation language for several reasons [6]. One of the reasons was the availability of existing, public domain C++ object class libraries. 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 [7]. The idraw utility is a drawing editor which we integrated into the WorkBench to support creating, editing and saving the graphical data-driven interaction objects. This reuse of existing software enabled the addition of a major new function without the significant cost and time of implementing a drawing editor from scratch.

TAE PLUS AS A PRODUCTIVITY TOOL

There are several ways that TAE Plus can contribute to improving software productivity. It provides a development tool that aids in prototyping; gets the best from people; makes steps more efficient; and supports the reuse of software components.

Prototyping

Most organizations now recognize the importance of prototyping and getting the end-user involved in the design process. However, prototyping is not usually thought of as a way to improve productivity. In fact, the prototyping step is frequently avoided or only carried out in a half-hearted manner because of the fear that the end-user will want numerous changes and thereby slow down the design process. This "ostrich head in the ground" syndrome frequently ends in an unpleasant confrontation when the application is delivered to the end-user and the UI fails to meet user expectations. The resultant retrocoding and correcting is often difficult and has to be absorbed as a maintenance cost. Creating a prototype, which allows easy changes and iterative rehearsing of the UI, improves the efficiency of the design and development phase and reduces the likelihood of serious UI changes in the delivered system.

Prototyping fosters a dialog between the developers and the user that can solidify the real system requirements and specifications. As a tool that enables rapid prototypes to be built quickly and easily, TAE Plus can be used to design more effective and user-accepted applications.

Getting the best from people

To get the maximum productivity from each member of a development team individuals should be utilized in the areas that they have an expertise. Too often the people designing application user interfaces are the programmers, who frequently do not have any training in human factors or graphic art techniques. This tends to be an ineffective use of the programmer's expertise, and often results in a less than optimum user interface. The WorkBench was designed to eliminate this problem by giving the user interface design experts a tool that is easy to use (i.e., does not require programming skills), while freeing up the programmer to concentrate on the application specific code.

Making steps more efficient

Another productivity option [8] is to automate a previous manual step, thus eliminating the step entirely. In several of the existing user interface development tools (e.g., Telesoft's TeleUSE™, Visual Edge's UIMX™) including TAE Plus, there is the capability to automatically generate the application code that manages the designed UI. This eliminates the process of the application programmer having to manually generate and key in this code, thus reducing the likelihood of keyboard errors or incorrect function calls. Particularly in cases where the application is heavily interactive, this automatic code generation can account for the majority of the application code and significantly improve productivity of the development process.

Reusing Components

Another way to reduce the amount of source code written for an application, thereby reducing the development cost, is to reuse existing software. In TAE Plus, the WPT runtime services offload all of the display and management of the UI from the application code. This approach enables the application programmer to

concentrate fully on the application-specific functions, and not be concerned with the UI code. Also, TAE Plus itself reuses existing windowing software (e.g., MIT's X Window System, OSF/Motif, Stanford's Interview object classes), thus improving the productivity of its own development.

TAE PLUS USERS' EXPERIENCES

One way to measure how effective TAE Plus is as a productivity tool is to develop the same application twice, one time using TAE Plus and another time not using TAE Plus. While most users feel certain that TAE Plus is saving them development time, they are on tight development schedules and do not have the interest in building parallel UIs. However, a few case studies in which the same user interface was developed with and without TAE Plus give evidence that the productivity gain can be impressive.

In Case 1, a programmer from General Electric developed a simple screen copy utility which gathers information through radio buttons, action icons, and text input. Then, it sends the information to an HP printer, as well as updating a text widget on the screen. When he did not use TAE Plus and wrote the UI code directly within the application code, it took him 80 hours to develop an operational application. When he used the TAE Plus WorkBench to develop the same operational application, it took him 4 hours. This productivity gain of 95% is illustrated in Figure 5. However, it should be noted that the gain does not take into account the unmeasured factor that "it is always easier the second time around."

Figure 6 illustrates Case 2. A programmer at NASA with no TAE Plus experience, but with X Window System experience, was tasked to write a simple application and account for the time spent on developing it with and without TAE Plus. The application has two panels, a few action icons, a radio button bank, and a dynamic mover object that moves along a static background when the associated data value changes. Including the time it took to learn how to use the WorkBench to the completion of the operational application, it took him 9 hours. (Note: an experienced TAE Plus user did the same application in 1.5 hours.) The application developed without TAE Plus (thus, making direct calls to the X Window System) took him 52 hours, and this implementation was still a "bit buggy." Even as a beginner TAE Plus user, it took him over four times longer to develop the application without TAE Plus. In the case of the experienced TAE Plus user, the productivity gain was even more dramatic, with a 96% increase in development of the application. A year later we had another programmer write this application with and without using TAE Plus. He was experienced with using the Motif toolkit and he developed the application in 17 hours making direct calls to the Motif toolkit. Using the WorkBench (which he had never used before) it took him 5 hours. Even with an experienced Motif programmer there was a 70% improvement in development time when using TAE Plus for the first time.

Although these case studies certainly do not provide enough statistical data to allow any grandiose conclusions to be made, they do demonstrate real cases in which using a GUI development tool, in this case TAE Plus, has significantly decreased the time it takes to develop the application. In general, TAE Plus reduces the time it takes a developer to create, test and deliver a software system.

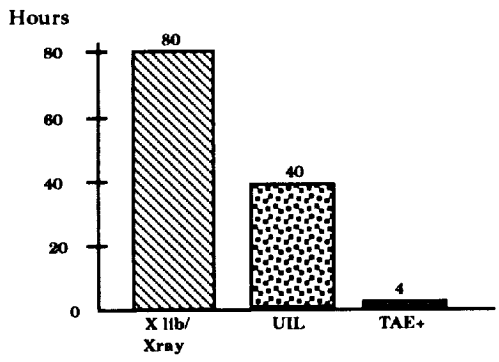


Figure 5: Case Study 1

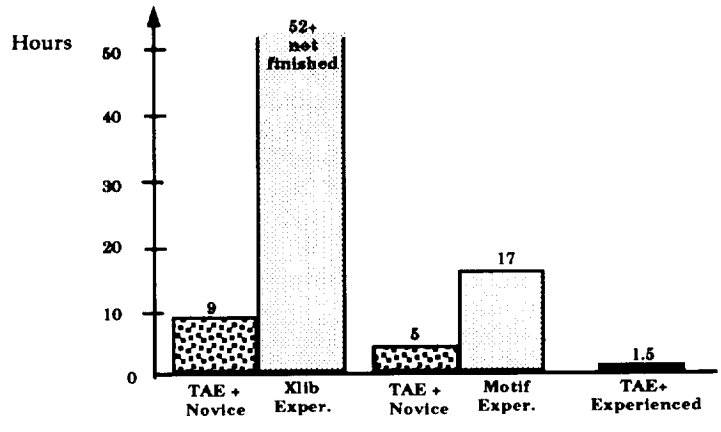


Figure 6: Case Study 2

AVAILABILITY AND MAINTENANCE

In December 1992 the latest version of TAE Plus (V5.2) became available from COSMIC, the NASA's software distribution center located at the University of Georgia. TAE Plus may be licensed by the public for a nominal fee and it is available on a variety of platforms: Sun workstations, Vaxstation II , Decstation 3100, HP9000, Masscomp, Silicon Graphics Iris and IBM RISC 6000. It is also available on the Vaxstation II under VMS and the NEC company has ported it onto their NEC EWS 4800/220 workstation for use by their customers.

Maintenance of a software system is a key factor in its success, and while every system is maintainable, how easy it is to maintain is the real issue. We knew when we began development that TAE Plus was targeted for wide application utilization and for different machines, so ease of maintenance has always been important. By providing the application-callable WPTs, applications are isolated from the windowing system. Thus, when the latest release or next generation windowing system shows up, only the WPTs will require updating or rewriting; the application code will not be affected.

User support is another facet of maintainability. Since the first release of TAE Classic in 1981, we have provided user support through a fully staffed Support Office. Users receive answers to technical questions, report problems, and make suggestions for improvements. In turn, the Support Office keeps users up-to-date on new releases, provides a newsletter, and sponsors user workshops and conferences. This exchange of information enables the Project Office to keep the TAE software and documentation "in working order" and, perhaps most importantly, take advantage of user feedback to help direct our future development.

APPLICATIONS USING TAE PLUS

Since 1982 over 900 installation sites have received TAE Classic and/or TAE Plus. Just over the past year, COSMIC has issued licenses to over 300 customer sites. 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 (13%), universities (15%), and private industries (40%). Within the government sector, users range from the National Center for Atmospheric Research, National Oceanographic and Atmospheric Administration, U.S. Geological and EROS Data Center, who are developing scientific analysis, image mapping and data distribution systems, to numerous Department of Defense laboratories, who are building command-and-control systems. Universities represented among the TAE community include 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 into the problems of construction and assembly in space. [9] Private industry has been a large consumer of the TAE technology and a sample of the companies that have received TAE Plus include Loral 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, 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.[10] General Software Corporation uses TAE in their commercial product, METPRO, a meteorological information processing system, which has been distributed in seven countries. Another company, Global Imaging, Inc. has embedded TAE into their commercial image processing system. 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 has become 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 enhancements and extensions to the WorkBench. 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 extensions to the dialog connections feature (e.g., graphic representation of the connection mapping, item-to-item connections).

Future advancements include expanding the scope of TAE Plus to include new tools and 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. Software tools, such as TAE Plus, are providing ways to make user interface developer's tasks easier and improve the overall productivity of the development process. This includes supporting prototyping of different user interface designs, as well as development and management of the operational application's user interface.

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 windowing systems, human factors research, standardization efforts and software portability. With TAE Plus's flexibility and functionality, it is providing a useful productivity tool for building and managing graphical user interfaces.

ACKNOWLEDGEMENTS

TAE Plus is a NASA software product being developed by the NASA/Goddard Space Flight Center with prime 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, (706) 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. Boehm, Barry, "Improving Software Productivity", IEEE Computer, September, 1987, pp. 43-57

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

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