N92-22717

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), supports prototyping, allows applications to be ported easily between different platforms and encourages appropriate levels of user interface consistency between applications. This paper will discuss the capabilities of the TAE Plus tool, and how it makes the job of designing and developing GUIs easier for the application developers. 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. Productivity tools to aid in the definition and management of user interfaces, thus, become an increasingly important element in the application's prototyping-to-operational development cycle.

Requirements for a prototyping-to-operational development environment
To support our development cycle 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,
- 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[1]. Over the years, TAE Classic has matured into a powerful tool for quickly and easily building and managing consistent, portable user interfaces, but only for the standard alphanumeric terminal. Not only did TAE Classic improve the productivity of a single application's development life cycle by providing the programmers with an easy and standard method for creating menus, prompting for parameters and building command procedures, but, because the tool was generic and reusable for multiple applications, the productivity gain for implementation increased exponentially. Other gains were realized in a significant reduction in application testing time (i.e., the user interface component, TAE, is reliable, debugged software) and maintenance overhead (i.e., application code uses TAE services, thus becoming hardware and operating system independent, which simplifies making application changes and enhancements.)

In the past six years, the emergence of the low-cost graphic workstation has enabled development of innovative graphical user interfaces (GUIs). Along with this new capability and flexibility comes a significant increase in the complexity of developing these graphical user interfaces. The array of new UI elements associated with GUIs (e.g., windows, panes, color, direct manipulation, programmable cursors) requires the developer/programmer to understand a complex new software environment (e.g., the X Window System™, "widget" architecture, OSF/Motif™ and AT&T's Open Look™ user interfaces.) This required expertise can translate into an increase in programmer training. Further, maintenance nightmares can occur as the low-level windowing systems are upgraded/changed. Frequently the cost of the GUI development can increase to the point where it exceeds the application-specific components. At GSFC we wanted to take advantage of the new GUI capabilities, but needed a way to improve the productivity of developing an application's graphical user interface component. We took advantage of the lessons learned in the TAE Classic development. 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 GUIs. 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 needs only to be concerned about what results are returned by the user interaction and not the look of the user interface.

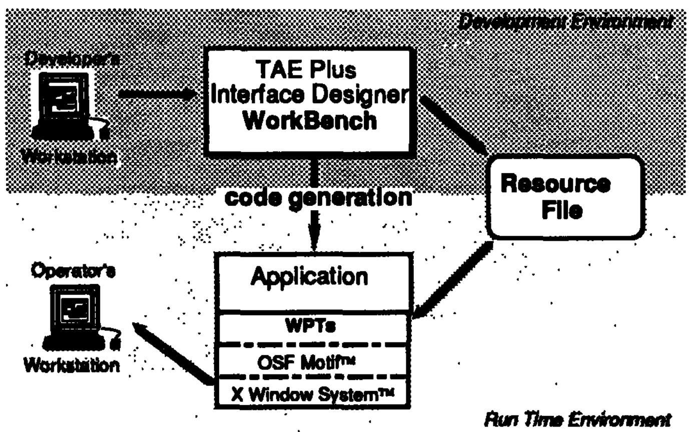
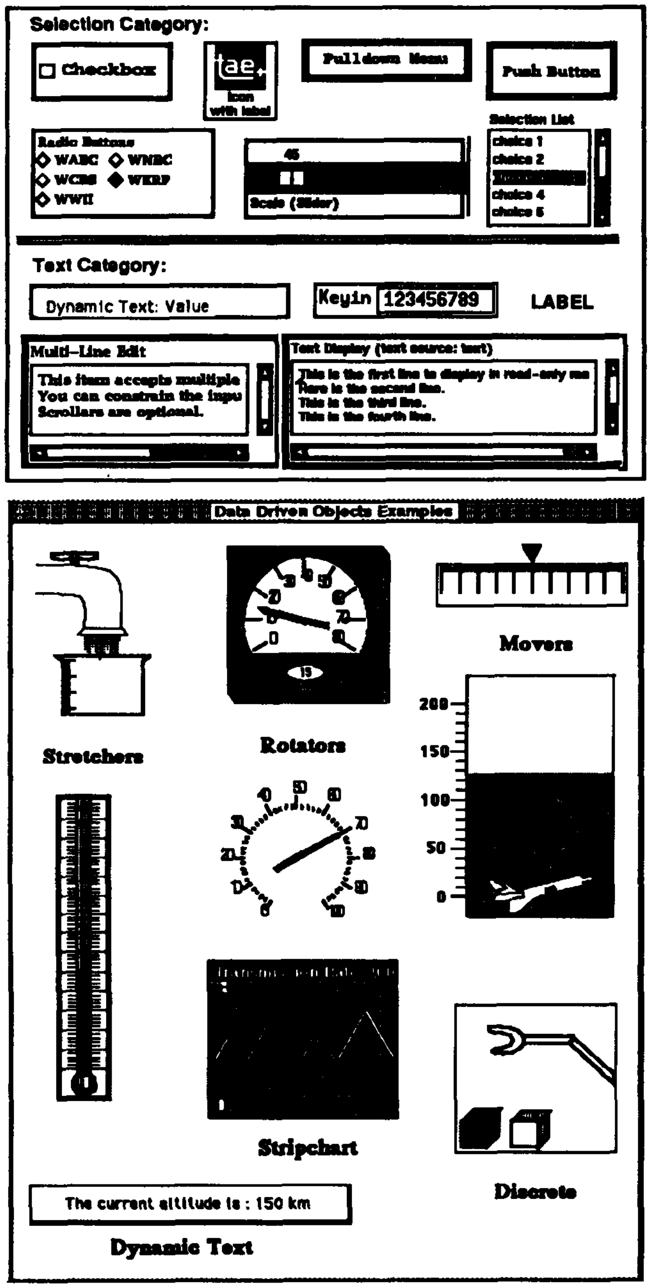


Figure 1. TAE Structure

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 which will display and manage the designed user interface. This gives the programmer a working template into which application-specific code can be added.



Dynamic Text: Value

This item accepts multiple
You can constrain the inpu
Scrollars are optional.

This is the first line to display in read-only mo
Here is the second line.
This is the third line.
This is the fourth line.

The current altitude is : 150 km

Figure 2. TAE Plus User Interface Interaction Objects

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: 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 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 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.

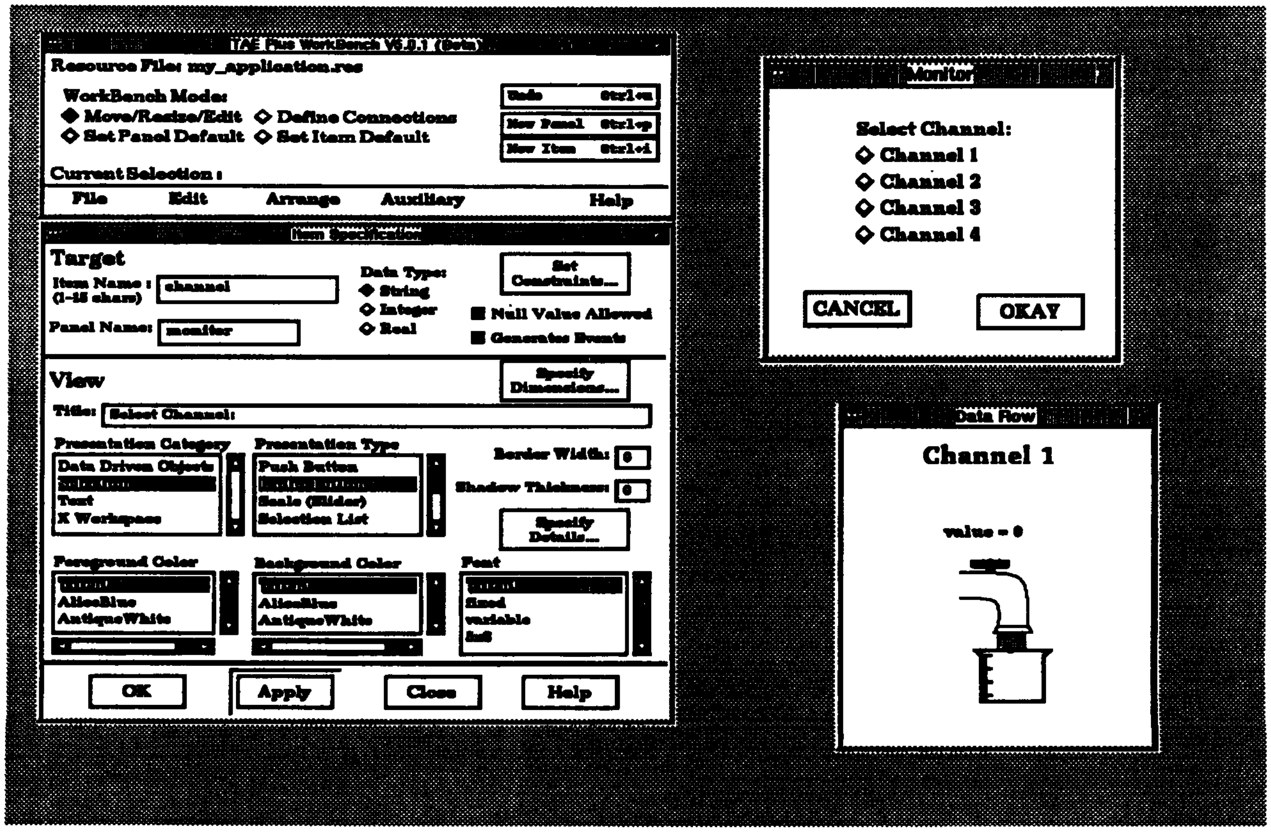
When creating a data-driven object, the designer goes through a similar process by setting the associated attributes (e.g., color thresholds, maximum, minimum, update delta) in the specification panels. To create the associated graphics drawing, the WorkBench provides a drawing tool within which the static background and dynamic 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 development 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, and the TAE Command


TAE Plus WorkBench V5.0.1 (Beta)

Resource File: my_application.res

WorkBench Mode:
◆ Move/Resize/Edit ◇ Define Connections
◇ Set Panel Default ◇ Set Item Default

Current Selection :

Item Specification

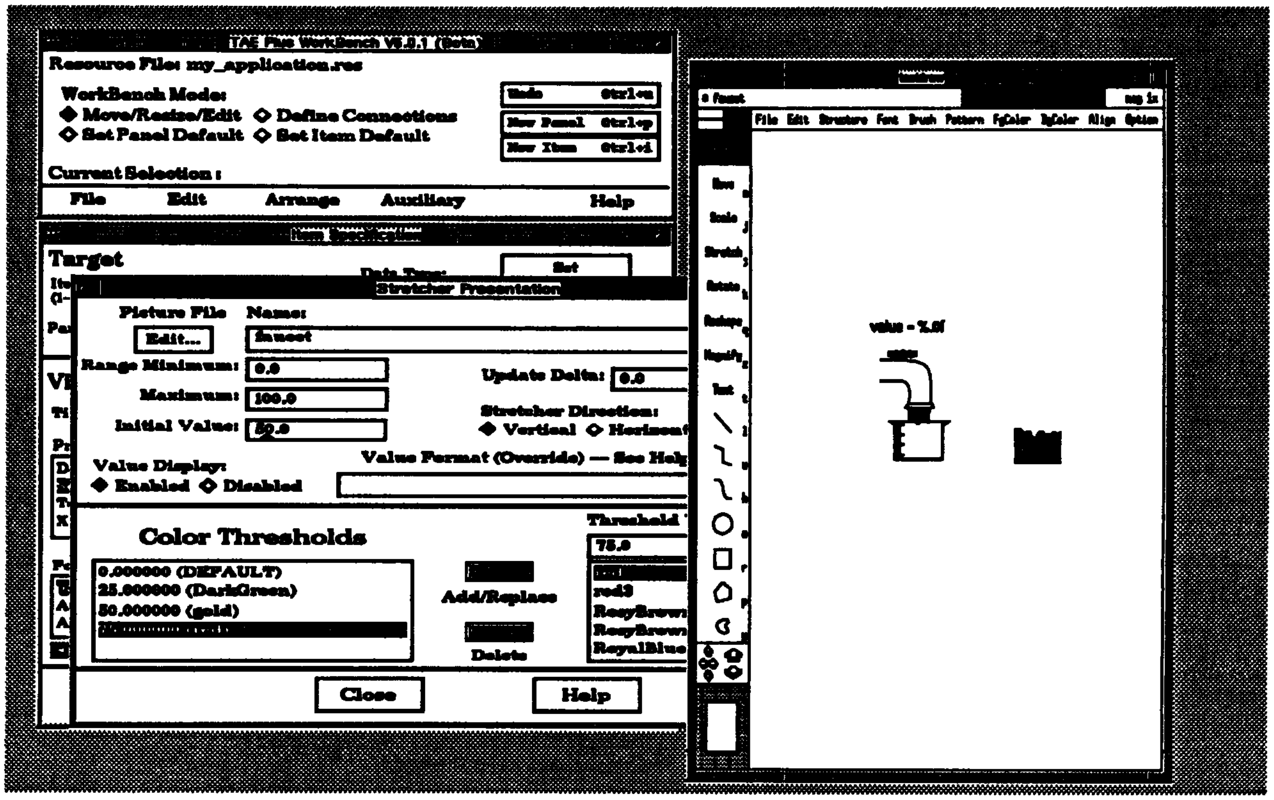
Target
Item Name : channel
(1-15 chars)
Panel Name: monitor
Data Type:
◆ String
◇ Integer
◇ Real
■ Null Value Allowed
■ Generates Events

View
Title: Select Channel:
Presentation Category: Data Driven Objects, Selection, Text, X Workspace
Presentation Type: Push Button, Radio Button, Scale (Slider), Selection List
Border Width: 0
Shadow Thickness: 0
Foreground Color: AliceBlue, AntiqueWhite
Background Color: AliceBlue, AntiqueWhite
Font: fixed, variable, 8x8

Monitor
Select Channel:
◇ Channel 1
◇ Channel 2
◇ Channel 3
◇ Channel 4

Data Flow
Channel 1
value = 0

Figure 3. Building a user interface with the WorkBench


TAE Plus WorkBench V5.0.1 (Beta)

Resource File: my_application.res

WorkBench Mode:
◆ Move/Resize/Edit ◇ Define Connections
◇ Set Panel Default ◇ Set Item Default

Current Selection :

Item Specification

Target

Stretcher Presentation

Picture File Name: faucet
Range Minimum: 0.0
Maximum: 100.0
Initial Value: 50.0
Update Delta: 0.0
Stretcher Direction:
◆ Vertical ◇ Horizontal
Value Display:
◆ Enabled ◇ Disabled
Value Format (Override) — See Help

Color Thresholds
0.000000 (DEFAULT)
25.000000 (DarkGreen)
50.000000 (gold)

Threshold
75.0
red3
RosyBrown
RosyBrown
RoyalBlue

value = %.0f

Figure 4. Creating a stretcher data-driven object

Language (TCL) (an interpreted prototyping language) 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.

The WPT package utilizes the the MIT X Window System, as its standard 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. Due to the growing acceptance of the OSF/ Motif user interface style as a defacto industry standard, the latest release of TAE Plus (V5.1) is based on the Motif software.

The WPTs also provide a buffer between the application program and the Motif toolkit and Xlib 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 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.

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]. 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[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.

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 the OSF/Motif X Toolkit and operates either UNIX or VMS, TAE Plus can be ported to it with reasonable ease. For instance, TAE Plus is operational on the following UNIX platforms: Sun workstations, Apollo, VAXstation II, DECstation 3100, HP9000, Masscomp, Silicon Graphics Iris, NEC EWS 4800/220 and Macintosh II (A/UX). TAE Plus is also available and validated on the VAXstation II and VAXstation 3100 under VMS.

TAE PLUS AS A PRODUCTIVITY TOOL

For years the software industry has been searching for ways to quantify the software development process allowing for accurate measurement of productivity. Due to the cerebral versus mechanical nature of software development this is a difficult task, which has lead to a large volume of published approaches on how to improve software productivity.[8] Barry Boehm identifies six primary options for improving software productivity[9] and TAE Plus addresses each one of these options at some level.

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 most often do not have any training in human factors or graphic art techniques. This tends to be an ineffective use of the programmers expertise, and frequently results in a less than optimum user interface. The WorkBench was designed to eliminate this problem 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.

Make steps more efficient
As stated by Boehm, "the primary leverage factor in making the existing software process steps more efficient is the use of software tools to automate the current repetitive and labor-intensive portions of each step." Prior to a tool like the WorkBench, the layout of the user interface involved either paper and pen mockups and layouts, or programmers creating the UI as they coded. In either case, the availability of an interactive user interface layout tool that allows the designer to define and build the UI in a WYSIWYG manner, makes the UI design process more efficient.

Eliminating Steps
The next productivity option is to automate a previous manual step, thus eliminating the step entirely. TAE Plus provides 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.

Eliminating Rework
Information hiding and prototyping are both ways that contribute to avoidance of reworking code. In TAE Plus, all the details of the user interface are hidden from the application. For instance, the application calls on a WPT routine (i.e., Wpt_NewPanel) to display a panel and its interaction objects. The application is not interested in whether the user is being presented with a radio button bank or a scrollable text list, but it is interested in which choice the user makes from this interaction object. The actual display and management of the UI is handled within the WPTs, thus isolating the UI code from the application. During an application's evolution, this approach of hiding details within the WPTs minimizes or eliminates the impact that changes to the UI hardware, the windowing system, the object class, etc., will have on the application.

Building simpler products
The number of software source instructions programmed during an application's development has the most significant influence on software costs. One approach to improving productivity is to reduce this number by building simpler products and eliminating "software gold plating: extra software that not only consumes extra effort but also reduces the conceptual integrity of the product."[9] Using rapid prototyping as a step in the specification process can frequently prevent the over specification of functions by users who are worried that if they don't specify everything

that can think of, then the system will not have some function they need. Although there is no guarantee that rapid prototyping will result in a simpler program, it 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 simpler applications.

Reusing Components
Another way to reduce the amount of source code written for an application is to reuse existing software. TAE Plus was designed with software reuse as a primary goal. 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 standard 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 CASE STUDIES

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.

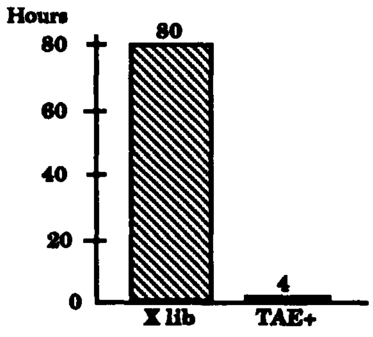


Figure 6. Case Study 1

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 6. 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 2 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. 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.

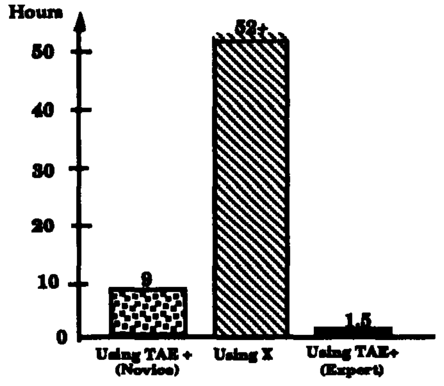


Figure 7. Case Study 2

AVAILABILITY AND MAINTENANCE

In April 1991 TAE Plus 5.1, which uses the latest version of OSF Motif™ (V1.1), became available from COSMIC, the NASA's software distribution center located at the University of Georgia. Versions for numerous UNIX work-stations (e.g., Suns, DECstation 3100, HP9000, Apollo) and for VMS/DECWindows™ may be licensed at a nominal fee.

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. This service has been one of the primary reasons for the success of TAE. Through the 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. 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 database 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 Adminstration, 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.[10] Private industry has been a large consumer of the TAE technology and a sample of the companies that have received TAE Plus include Apple Computer Inc., 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, 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.[11] 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, refinements in the code generation feature, extensions to the connections feature (e.g., graphic representation of the connection mapping, item-to-item connections), and multiple console support.

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, command language design, 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, (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. Evans, M.W., Piazza, P., Dolkas, J.B., *Principles of Productive Software Management,* John Wiley and Sons Publishers, 1983.

9. Boehm, Barry, "Improving Software Productivity", *IEEE Computer,* September, 1987, pp. 43-57

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

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