THE BENEFITS OF DESKTOP RAPID PROTOTYPING

Cynthia Hubbell
Tony Mancuso
Dale Wainwright
Hughes Aircraft Company
Ground System Group
Fullerton, California 92634

ABSTRACT

Navy combat system users have experienced an information explosion as the end product of recent technology advances. The by-product is an increase in the complexity of operation of these systems. Accordingly, a requirement has surfaced to provide an advance disclosure of the man-machine interface approach to illustrate a selected subset of the operator input and display capabilities. However, the use of custom design hardware and software can be prohibitively expensive in today's defense environment. Fortunately, with the improved processing capabilities of today's desktop computers, combined with the availability and affordability of commercial software demonstration packages, an engineer can now be provided the tools necessary to mock-up and demonstrate significant portions of a system's man-machine interface design in very short time, at significantly less expense.

This paper describes the advantages of using desktop computer systems to provide a low-cost rapid prototyping capability for the evaluation of the man-machine interface design of complex training systems. The example presented in this paper provided a cost savings of 75 percent over the cost to program the target hardware. Furthermore, the cost savings were achieved despite delivery of a significantly more complete disclosure of the man-machine interface.

INTRODUCTION

The Operational Environment.
Imagine a Naval training system with the capability to exercise in excess of 250 trainees, with more than 30 different operator modes. These modes are housed inside more than 25 separate mock-up areas of up to 7 mock-up types. The mock-up types can be any one of 18 Naval surface combatant ship class CIC's (Combat Information Center), or a submarine control center, or an ASW aircraft cockpit, to mention a few possibilities.

The operator modes range from:

- High-level Battle Group Commander and his designated warfare area commanders (riding any number of ship classes)
- Shipboard warfare area coordinators that are issuing weapon engagement orders inside each ship CIC mock-up, down to
- Sonar Watch Supervisor, who is reporting and monitoring individual sonar system detections.

To further compound the complexity, the operator modes to be simulated are not standardized in terms of the range of capabilities or man-machine interface (MMI) design throughout the Fleet. This lack of standardization is the result of: (1) variations in systems and their capabilities from ship to ship, and (2) the gradual modernization of these systems as the technology and the funds have become available.

The above is a description of the Device 20A66 ASW Tactical Team Training System, which illustrates the complexity of the system from both an operational and design standpoint. [1] The Desktop Rapid Prototyping System presented in this paper can be applied to many systems. The Device 20A66 Program used the Desktop Rapid Prototyping System (RPS) and therefore provides a good example from which to illustrate some key points and understand the central concepts.

MMI Specification. Defense systems, and Navy combat systems in particular, are becoming more and more complex with time. As the complexity of combat systems increases, so does the complexity of the training systems which must support them. Accordingly, customer evaluation of the proposed trainer student MMI is a difficult task. To fully comprehend the actions the many different trainee modes must perform, as well as to visualize the displayed information, is very difficult, if not impossible, without graphical documentation of the MMI design.

The DoD-STD-2167A Software Requirements Specification (SRS) may provide some level of detail for the MMI in terms of the display formats and the procedures. However, trying to understand the MMI by searching through thousands of pages of processing requirements falls short of the complete picture necessary to ensure that the contractor is providing an optimized design. A personal computer-based, graphically-oriented interactive representation of the MMI can provide a much greater level of MMI design disclosure with a minimum level of effort. This representation needs to be: (1) developed quickly, (2) with minimum cost, and (3) able to facilitate the iterative design process. In other words, a rapid prototyping capability is required.

DESCRIPTION OF THE SYSTEM ENGINEERING ENVIRONMENT

Top-Down Design Approach.
In general, large training systems are developed using the classic top-down or "waterfall" design approach as illustrated in Figure 1. [2] [3] As a result of a task analysis, system-level requirements are defined in the system specification. The system specification provides the basis for the Software Requirements Specification (SRS), which contains the processing, input/output, internal and external interface, and MMI requirements. This document is generally rather voluminous in terms of page count. For example, the Device 20A66 SRS

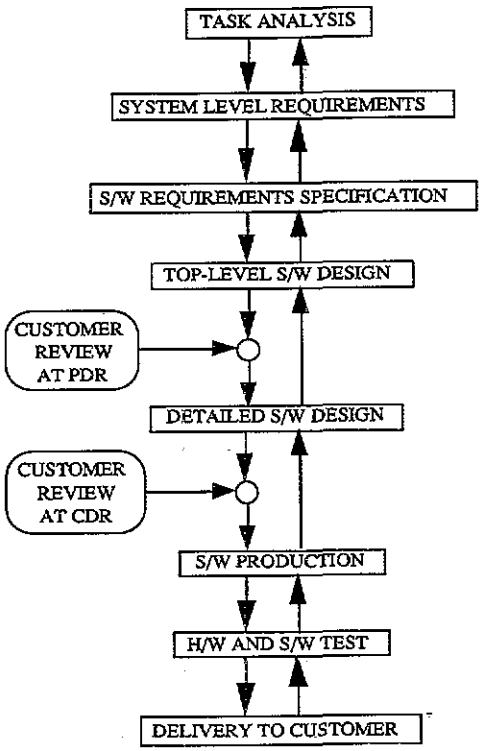

Figure 1. Simplified Top-Down Design Approach

exceeded 3000 pages. [4] For most systems, the man-machine interface details are not even discussed at this stage of the development cycle. Nevertheless, the evaluated design is then subjected to Top-Level Software Design (based upon the requirements listed in the SRS), which addresses the allocation of functional capabilities to software modules. The software is then defined in greater detail during the Detailed Software Design phase. Once the design is approved at the Critical Design Review (CDR), code development is begun, followed by various levels of software and hardware testing. After test and integration, the trainer is delivered to the customer.

The top-down design approach creates a major dilemma for the man-machine interface designer. The amount of attention given to the user-interface during the preliminary design phase does not provide proper evaluation during the early development phases. Once the detailed design and coding efforts have begun, it becomes costly to correct defects in the interface. In fact, it is generally acknowledged that the "Law of Ten" can be applied to the software development process: "If an error is detected during the detailed design phase, it is ten times more costly to correct than the same error detected during the top level design phase."

It should be further noted that software programs being developed today may have up to 70 percent of their code supporting the man-machine interface. Additionally, to the trainer system's end-users, the interface is the system. Therefore, every effort should be taken to refine the man-machine interface as early in the design phase as possible.

Star-Based Life Cycle Approach.

As a result of the difficulties cited above several alternative design approaches have appeared in the last five to ten years. The "star-based" life cycle approach depicted in Figure 2 has been found to be useful in the development of non-defense programs and certainly portions of this alternative design approach are applicable to the defense arena. [5]

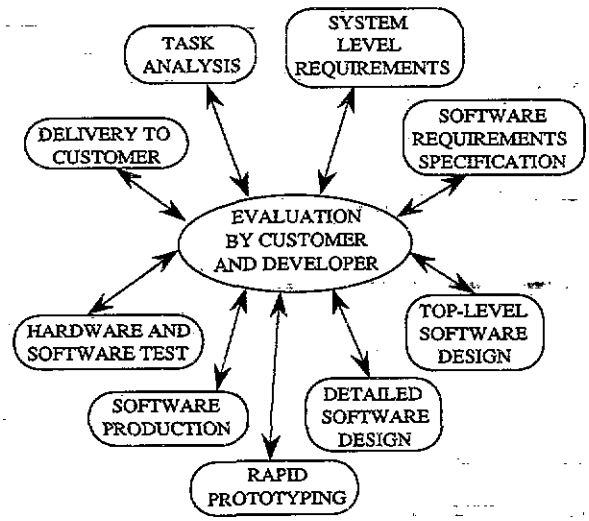

Figure 2. Simplified Star-Based Life Cycle Approach

In this approach the starting point for the design process is flexible. One may start with a short period of task analysis and evaluation and proceed quickly to the development of an initial prototype. Evaluation of the prototype can then drive the requirements generation process. Iterating the prototype throughout its life cycle produces a more mature set of system requirements. This star-based approach also supports concurrent activities in each of the identified areas where the final product is the result of an iterative process that relies heavily on evaluation of the design as depicted in the prototype.

RAPID PROTOTYPING DEFINED

General Definition and Associated Benefits.

Rapid prototyping in name is not a new concept to the design and development of military systems. It has become a "buzzword" in the systems development area and has been applied to various forms of system simulations and demonstrations. For the purposes of this paper the following general definition/objective of rapid prototyping will be employed: "The application of readily available computer hardware and software to improve the system engineering process."

The improvements obtained can be as diverse as:

- Early disclosure of system design to customer
- Resolution of target hardware issues
- Demonstration and refinement of software algorithms
- Greater communication between systems and software engineers
- Concurrent engineering for systems and software efforts and
- Documented MMI designs.

However, most often the main goal of the rapid prototype demonstration is to provide a design disclosure of a selected segment of the system MMI design for customer approval.

Prototype Environment Decision Factors.
When prototype efforts are initially undertaken, the program managers must decide:

- How much of the system will the prototype demonstrate (i.e., fully functional versus a facade simulating only the user-interface)?
- Will the demonstration be performed on the target hardware or on some other platform (i.e., pencil and paper, personal computer, workstation, etc.)?
- Will the prototype provide usable code for the final product or is it intended to be used for demonstration purposes only?

These questions significantly affect the level of effort in terms of required manpower and time to develop.

Level of Functionality. If the prototype is intended to demonstrate the complete functionality behind the capabilities selected (i.e., algorithmic results, database searches, dynamic data updates, etc.), then a significant amount of the design must be complete when prototyping efforts begin. As a result, the rapid prototype can require nearly the same level of effort required to build the actual system and therefore cannot be considered rapid or cost effective.

The other alternative is to provide a mock-up of a critical portion of the system, providing the visual appearance of the target system and addressing mainly the man-machine interface design issues. Such a prototype can be developed rapidly and early in the specification process, evolving as the design details mature.

Demonstration Hardware. If the prototype must be generated on the target hardware system, then there may be large costs associated with the acquisition of the hardware for demonstrations. If the hardware is not readily available, prototyping may be delayed. The selection of target hardware often dictates the use of difficult-to-use software tools or worse yet, the generation of a custom development environment. Prototype development is often slow, and financial as well as time constraints may dictate that only a small portion of the design be developed.

If personal computers provide sufficient speed and memory for the desired level of simulation, the use of readily accessible Commercial off-the-Shelf (COTS) hardware can be a source of great savings. Additionally, many COTS software tools exist for the generation of prototypes. Informative, interactive, and highly-capable demonstrations can be developed in a short period of time.

Code Evolution. If the software code written to build the rapid prototype demonstration is to be used on the final system, then the system design must be detailed and mature before the rapid prototype demonstration work even begins. The desire for evolvable code will also impact the software tool selection process.

Often programs that have committed to provide a rapid prototyping demonstration environment initially intend to build a prototype which contains more than the front-end MMI. They also want the generated code to evolve to the target language for delivery with the final system. However, to provide the customer with demonstrations early in the program (when they can be used to cost-effectively influence the detailed design in a pro-active versus a reactive manner), the above is usually unattainable.

As a result, demonstration prototypes are usually written in a language that is familiar to the software organization, easy to write, and forgiving of the many modifications and patches that occur during development process. The capabilities are scaled back significantly to demonstrate only critical, selected portions of the system, supplemented by a larger number of solvable and less risky functional capabilities.

It should be noted that the success of the contractor's prototype is largely dependent on the expectations of the customer based on the initial demonstration plan disclosure and the customer's previous experiences.

A DESKTOP RAPID PROTOTYPING SYSTEM

Prototype Hardware Requirements.
The Device 20A66 Program contract required MMI demonstrations at both the Functional Mock-up Review (FMR) and at the Preliminary Design Review (PDR) three months later. The FMR, which occurred 13 months after contract award, very specifically required the use of the target hardware, but left the software programming language to the discretion of the contractor. The contractor's initial goal was to provide deliverable code, but writing a demonstration in Ada six to twelve months after contract award and then delivering the same code with the System four years later proved unrealistic. This is a common dilemma experienced by program managers. If the demonstration is to occur early in the program when MMI changes can be entertained at less cost, then the code will more than likely be "throw-away" code. If the demonstration is scheduled later in the program when the design is mature, then it is often too late to entertain significant changes to the MMI at a manageable cost.

The PDR requirements did not specify the demonstration hardware, but probably assumed the use of the FMR hardware, since it was already available. In addition, time and money had already been spent to assemble and integrate the configuration for the FMR. However, to demonstrate the additional modes required at PDR (in less than three months after FMR), would require as much time and money as the FMR effort. A more efficient method had to be identified, which led to the decision to use the Desktop RPS.

The reasoning which lead to the selection of the RPS hardware first requires an understanding of the operational display systems. The bulk of the tactical displays simulated in the Device 20A66 Trainer System are from the UYQ-21 Navy Standard Display Console family as depicted in Figure 3. The figure indicates the locations of the Plan Position Indicator (PPI), Data Display Indicator (DDI), the Computer Controlled Action Entry Panel (CCAEP), the Category Select Panel (CSP) and the Variable Function Keys (VFKs). The PPI supports the display of track symbols, sensor video, complex graphic

images and VFK labels. The DDI supports tabular text displays to amplify the tactical symbology display. The CCAEP and the VFKs are the operator's primary method for entering commands to the combat system.

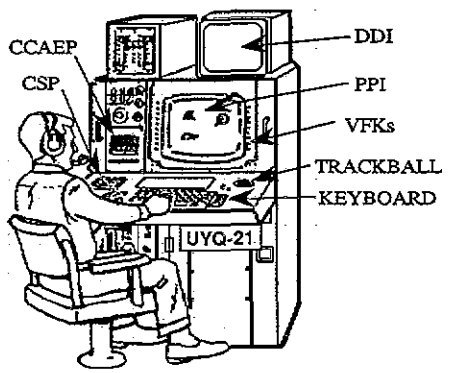

Figure 3. UYQ-21 Navy Standard Display Console

To properly demonstrate the operator MMI design, the RPS hardware must be capable of simulating the Device 20A66 display systems. The Device 20A66 display simulation hardware for student modes, referred to as the Vertical General Purpose Console (VGPC), is shown in Figure 4. The console configuration employs off-the-shelf hardware to simulate the militarized hardware referenced above. Although the console architecture differs from that of the military version, the two man-machine interfaces are similar, so as to facilitate the transition from the operational to the training environment by fleet personnel.

To satisfy the many display requirements described above on a desktop computer system, we chose to perform the rapid prototyping using the Macintosh* II personal computer based primarily on its graphics capability. This choice of COTS hardware facilitated demonstration capabilities at several of our plant facilities, as there were several of the PCs available for use. That would not have been the case if the target hardware had been selected for use.

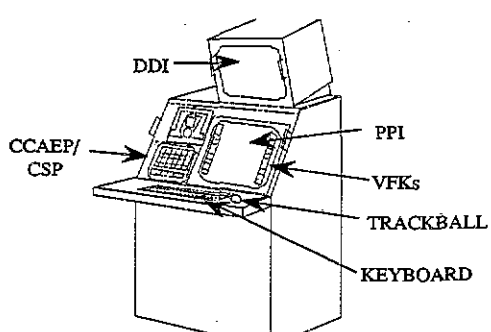

Figure 4. 20A66 Trainer Vertical General Purpose Console

The Desktop Rapid Prototyping System hardware configuration consisted of a Macintosh II personal computer outfitted with 8 MB of RAM and three monitors, as shown in Figure 5. The large 19" color monitor corresponded to the PPI on the VGPC. A sample set of RPS supported PPI symbology is shown in Figure 6.

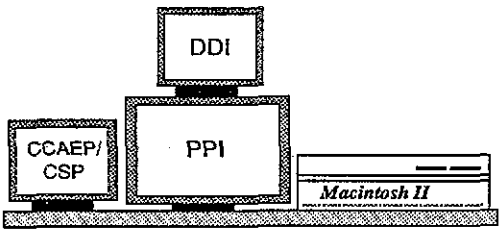

Figure 5. Desktop Rapid Prototyping System

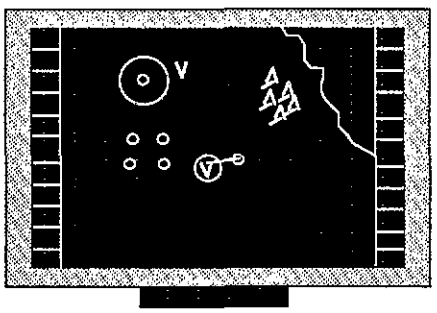

Figure 6. Sample Desktop RPS PPI Display

One of the RPS' two 13" monochrome monitors corresponded to the DDI on the VGPC. The other 13" monitor corresponded to the CCAEP on the VGPC, which is a touch-sensitive, button-oriented input device. Figures 7 and 8 provide sample depictions of DDI and CCAEP displays as supported by the RPS hardware. The mouse (or a trackball) on the Macintosh corresponded to the trackball on the VGPC, which is used for cursor (balltab) manipulation.

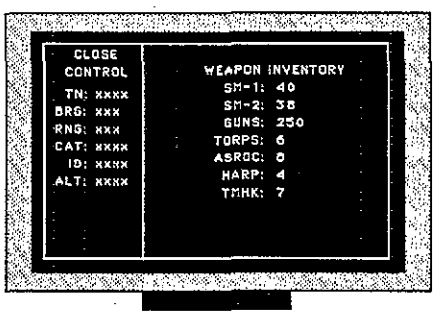

Figure 7. Sample Desktop RPS DDI Display

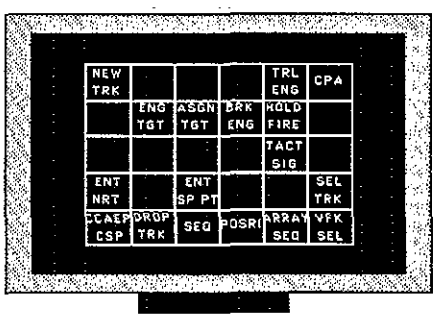

Figure 8. Sample RPS CCAEP Display

Prototype Software Requirements.

For prototyping an MMI intensive project, such as the Device 20A66 ASW Tactical Team Trainer System Program, it is critical to select a demonstration tool that possesses several key characteristics:

- The tool must be graphics-oriented. Graphic objects must be easily created, modified, and manipulated. These graphic objects should be either bit-mapped or object-oriented.
- The tool must also support multiple display surfaces, as the target system uses three display monitors.
- A color display capability is also necessary to accurately model the target hardware configuration.
- The windowing environment should include the ability to re-size, clip, and display concurrent windows in order to facilitate the prototype demonstration development efforts.
- The last and most important characteristic the tool must exhibit is the ability to store and retrieve information in a non-linear fashion. This is because the structure of information needed to represent all the elements in a machine interface is not linear.

Data Structure Models

The data structure model which most closely parallels this non-linear set of data is the "network data structure," which consists of a "tree," where each node of the tree can have more than one "parent." An example network data structure for two CCAEPs is depicted in Figure 9. Different forms of information are stored at each node of this tree, and "links" are created between related nodes. The links are formed according to how the user has the information organized in his head, not according to how a programmer thinks the information should be organized. The information contained at each node can be of virtually any form. Some nodes may be associated with textual information, while other nodes contain graphics, and still others may contain databases.

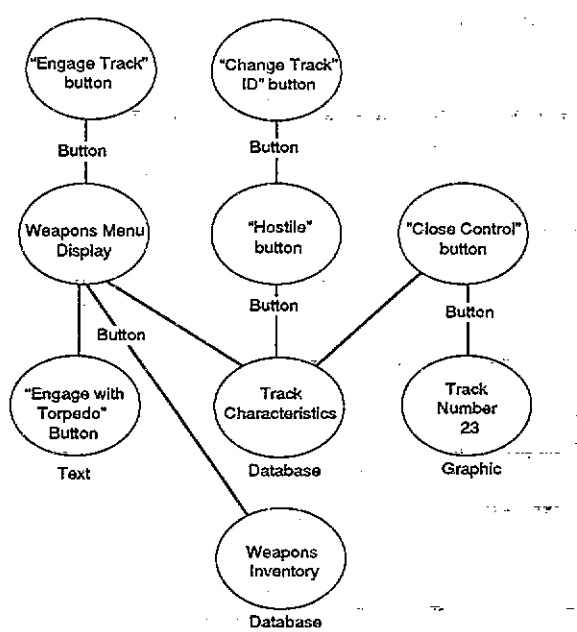

Figure 9. Sample Network Data Structure For "Engage Track" And "Change Track ID" CCAEPs

Just as important as the ability to store and retrieve information under this structure, is the ability to easily navigate through the structure. The nature of these data storage and retrieval requirements make HyperText** products a logical candidate for selection as the demonstration tool for the rapid prototyping of MMI. HyperText provides a way of accessing information through relations and links.

Features of the SuperCard† HyperText Application.

Several HyperText products were evaluated against the criteria mentioned above. SuperCard by Silicon Beach Software was selected for its high compliance rate with the above noted evaluation criteria (i.e., graphic-oriented, color capable, multi-windowing, and non-linear data structure). [6]

SuperCard employs the concept of HyperText, and performs many diverse and advanced capabilities. It is a database management system, a graphics program, and a program generation tool which provides the following functionality all at the same time:

- Since it uses the HyperText philosophy, it can store and retrieve information in non-linear structures. As a database, SuperCard can also store and retrieve information linearly and even join multiple databases like relational database management systems.
- Its graphics portion supports the precision of bit-mapped ("pict") imagery as well as the ease of use of object-oriented ("draw") graphics.
- It supports multiple monitors, color, and concurrent windows.
- Although the program generation aspect of SuperCard was not on the list of required features for the demonstration tool, this became a very useful feature. It allows for the conversion of a SuperCard "project" (a file that requires the SuperCard application to run) to a stand-alone application that requires no additional programs or resources to run.

SuperCard uses windows as the primary method for displaying information. Contained in each window is one or more "cards." Cards are the basic unit of information in SuperCard. Each card contains any number of user-created objects (i.e., "fields," "buttons," and "graphics"). Cards express information and allow the developer to control their project. A card can be linked with cards from the same window, with cards from another window, or even with cards from another project. It is these user-created card linkages that give SuperCard its HyperText capability, offering prototype developers enormous flexibility and power.

Desktop Rapid Prototype Capabilities.

The implementation of the MMI Desktop Rapid Prototyping System for the Device 20A66 program features a single SuperCard window for each of the three monitors. The CCAEP window contains one card for each CCAEP array of buttons applicable to the operator mode being prototyped (see Figure 8). Each CCAEP button is represented by a SuperCard button which is linked to the appropriate information according to the requirements specifications for the Device 20A66 trainer. For example, if the requirements state that depressing a certain button on the CCAEP shall result in the display of information in a certain format on the DDI, that's exactly how the RPS behaves.

Figure 10 provides a sample script that links a CCAEP button to the display of DDI and PPI information. When the associated CCAEP is selected via a mouse selection in the CCAEP array window, the mouse selection message is evaluated against the script details for the CCAEP array script. The portion of the script which defines the action of the selected CCAEP is then executed. All the tools for performing this script linkage are at the developer's fingertips. The developer can: (1) draw the CCAEP buttons using SuperCard "graphics" tools, (2) format the DDI data using SuperCard "field" tools, and (3) create the physical link using SuperCard "button" tools, all in a matter of minutes. With the Desktop RPS, if the information changes due to fluctuating requirements, such as changes to the DDI format contents, the developer can make this change in seconds or minutes, not hours or days as would be required in a conventional display oriented program.

Object Script for "AttackDisplay" VAB

```
on mouseUp
  open card "Attack DisplayDDI" in window "DDIs"
  open card "Initial PPI" in window "PPIs"
  set the visible of graphic HostileSub4421 to true
  set the visible of graphic HostileSub5543 to true
  set the visible of graphic HostileSub8042 to true
  set the visible of graphic HostileSub4407 to true
end mouseUp
```

Figure 10. Object Script For
The "Attack Display" CCAEP

COMPARISON OF TWO APPROACHES

FMR Demonstration.

For the FMR demonstrations a trainee console, instructor console, plot table and a working Local Area Network (LAN) formed the hardware suite with supporting software defining and demonstrating selected portions of five trainee modes, two instructor modes, and the plot table operations. To develop the demonstration three separate disciplines were required: systems engineering to identify and write the requirements, hardware engineering to assemble and test the components, and software engineering to write the code. (For purposes of this comparison, the hardware assembly, integration and test effort was not factored in because it represented a separate and necessary task independent of the requirement to demonstrate operator MMI.)

The process used to develop the demonstrations required systems engineering to identify the capabilities to be demonstrated and write scripts for those capabilities. The scripts provided detailed procedure sequences (beyond what was available in the SRS) for input to software engineering for programming. The systems engineering effort continued throughout the software development process as sustaining support for update and review. In fact, it required portions of seven systems engineers' time over a four-month period, representing 6.1 man-months. The total system engineering effort represented 24.7 percent of the labor cost for the demonstration. The software engineering effort required larger portions of seven programmers' time over the same four-month period, totalling 18.6 man-months.

Rapid Prototyping System Demonstration.

For the PDR scheduled three months after the FMR, the contract required expansion of the five trainee and two instructor modes to incorporate the recommended changes from FMR, plus the demonstration of four additional trainee modes. Realizing that an effort comparable to that expended for the FMR would be required if the target hardware were used, we decided, with customer approval, to use the Desktop Rapid Prototyping System described above. (For purposes of the MMI design disclosure, this would have been preferred for the FMR as well, but due to contractual hardware requirements we were not at liberty to use the Desktop Rapid Prototyping System at FMR.)

The RPS demonstration development process occurred within one engineering discipline, which in our case was systems engineering. In fact, using one engineer familiar with the software requirements and another to program the demonstration on SuperCard significantly reduced the effort previously required for software support. For example, to implement eight trainee modes and one instructor mode using the Desktop Rapid Prototyping System required only 5.6 man-months total effort. To do the entire demonstration on the RPS required 8.5 percent fewer man-hours than the systems engineering support portion of the FMR demonstration. In addition, it represents just 23 percent of the labor cost expended for the FMR demonstration, which provided fewer capabilities and in some cases a sequentially scripted scenario. If we were to further account for the fact that the desktop Rapid Prototyping System demonstration is more complete from an MMI design standpoint, then the numbers become even more favorable.

CONCLUSION

To put the above level of effort figures in terms of dollars, assume a level labor rate of $50.00/hour. The FMR effort cost is $191,600 (or 3832 man hours) while the desktop Rapid Prototyping System effort cost is $43,250 (or 865 man hours). By using the desktop RPS, a savings of $148,350 (or 2967 man hours) was realized.

In addition to the cost savings, the demonstration can be developed in the engineer's office and shown to the customer at some other more convenient location, if desired. Due to the rapid nature of this development process, comments can be received early in the design process when changes are less costly. The MMI design can be validated more completely by the customer's subject matter experts, and the customer is provided the opportunity to participate more fully in design process and thereby assumes ownership of the design. Unlike other methods, the MMI design is not created in a technical vacuum, but is jointly developed by the contractor and the customer, thereby facilitating the incorporation of a broader range of critical input.

REFERENCES

[1] Specification for the ASW Tactical Team Training, Device 20A66, Naval Training System Center, Orlando, Florida, 13 January 1989.

[2] W. W. Royce. Managing the Development of Large Software Systems: Concepts and Techniques. *Proc. WESCON*, August, 1970.

[3] B. W. Boehm, Software Engineering. *IEEE TRANS. Comput.*, Vol C-25, pp.1226-1241, December 1976.

[4] Software Requirements for the Real-Time CSCI of the Device 20A66 ASW Tactical Team Trainer, Prepared by Hughes Aircraft Company, Training and Support Systems Group, Long Beach, California for Naval Training Systems Center, Orlando, Florida, 11 May 1990.

[5] H. Rex Hartson and Deborah Hix. Human-Computer Interface Development: Concepts and Systems for Its Management. *ACM Computing Surveys, 21*, pp 5-92, 1 March 1989.

[6] Gookin, Dan *The Complete SuperCard Handbook* Compute! Books 1989.

ABOUT THE AUTHORS

Cynthia Hubbell is a Systems Engineer in the Combat Systems Department at Hughes Aircraft Company, Ground Systems Group. She is currently acting as lead engineer for the Desktop Rapid Prototyping System effort for the Device 20A66 ASW Tactical Training System. She holds a Bachelor of Science degree in Cybernetics from the University of California at Los Angeles. At present she is a Hughes Master's Fellow pursuing a Masters of Science in Electrical Engineering from the University of Southern California.

Tony Mancuso is a Systems Engineer in the Combat Direction Systems Development Department at Hughes Aircraft Company, Ground Systems Group. He holds a Bachelor of Arts degree in Mathematics and a Master of Science degree in Computer Science. Mr. Mancuso provided the initial RPS concept using the SuperCard application and was instrumental to the implementation of the demonstrations. He has been involved in the specification and development of the Man-Machine Interface design for various combat systems during the past four years including the Advanced Combat Direction System (ACDS) Block 1 and the Battle Force Inport Trainer (BFIT).

Dale Wainwright is a Systems Engineer in the Combat Systems Department at Hughes Aircraft Company, Ground Systems Group. He is currently the lead systems engineer for the Student Subsystem design of the Device 20A66 ASW Tactical Training System. He holds a Bachelor of Science degree in Engineering Physics from the U.S. Naval Academy. Mr. Wainwright's work experience includes thirteen years of combined active duty and reserve service with the U.S. Navy and six years at Hughes Aircraft Company.

Footnotes:
* Macintosh is a registered trademark of Apple Computer, Inc.
** HyperText is a registered trademark of Apple Computer, Inc.
† SuperCard is a trademark of Silicon Beach Software, Inc.