Intelligent User Interface Concept for Space Station

by:

Edward Comer, Cameron Donaldson, and Kathleen Gilroy
Software Productivity Solutions, Inc.

and

Elizabeth Bailey
Software Metrics, Inc.

Introduction

The space station computing system must interface with a wide variety of users, from highly skilled operations personnel to payload specialists from all over the world. The interface must accommodate a wide variety of operations from the space platform, ground control centers and from remote sites. As a result, there is a need for a robust, highly configurable and portable user interface that can accommodate the various space station missions.

This paper presents the concept of an intelligent user interface executive, written in Ada, that would support a number of advanced human interaction techniques, such as windowing, icons, color graphics, animation, and natural language processing. The user interface would provide intelligent interaction by understanding the various user roles, the operations and mission, the current state of the environment and the current working context of the users.

In addition, the intelligent user interface executive must be supported by a set of tools that would allow the executive to be easily configured and to allow rapid prototyping of proposed user dialogues. This capability would allow human engineering specialists acting in the role of dialogue authors to define and validate various user scenarios. The paper will discuss the set of tools required to support development of this intelligent human interface capability and will outline the prototyping and validation efforts required for development of the Space Station's user interface.

The Space Station User Interface Problem

The space station user interface represents one of the greatest challenges in human-machine interaction to date. The development, operation and use of the space station will involve thousands of people from all over the world. The space station user community will include private industry, universities, and other government agencies as well as the various NASA centers and their contractors.

The space station user interface must provide support for traditional ground-based, on-orbit and payload operations, each of which involves numerous operational roles. The test and integration function is representative of the diversity of these roles [DOR83]:

- o mission and operations planning
- o simulation and modeling
- o manufacturing development and test

- o pre- and post-launch integration and testing
- o on-orbit integration and testing
- o on-orbit maintenance and repair
- o payload integration and testing
- o user payload data processing
- o environment monitoring and control
- o real-time flight and operaitonal functions

While the need to support traditional operational roles like launch and flight control will continue with the space station, an increasing number of users will not have experience with NASA mission operations. As a multi-purpose facility, the space station will support users of its scientific laboratories and payloads, users running manufacturing and repair operations, and users providing transportation services.

Space station activities will be distributed over many sites (both government and commercial), including space platforms, maneuvering vehicles, ground-based command stations, and data collection centers. Many of the activities currently performed by ground-based personnel on specialized systems will have to be executed on the space platform using multi-purpose equipment.

Analysis of user interface technology currently in use on NASA projects demonstrates that it is clearly not adequate to meet the space station challenge. Some of the problems that must be addressed include:

- o Integration with other systems and off-the-shelf products (currently difficult or not possible)
- o Lack of support for advanced interaction techniques
- o Inadequate development tools
- o Lack of uniformity - Interfaces differ from system to system, payload to payload, and site to site
- o Difficult to use - require the assistance of specialists to accomplish mission (not applications-oriented); not tailorable to needs of individual users; poorly human-engineered;
- o Modifications often require reimplementation
- o Difficulty in performing validation in either off-line or real-time modes

Benefits to be derived from improving the current user interaction approach include:

- o Reduced life cycle cost by providing the necessary flexibility for users to accomplish new mission operations, and longer life of the operational software due to increased adaptability [BAS85]

- o Greater level of automated support, providing easier operations, use, modification, maintenance and validation [DOR85]

- o Increased operational confidence because personnel can perform activities themselves [DOR85]

Types of Interaction to be Supported

A variety of user-interaction styles have been made possible by advances in hardware technology. With an increase in the amount of the information that can be displayed and the operations available, the potential for effective and highly usable interfaces is greatly increased. Given the anticipated number of users and wide variety of user profiles for the Space Station computing system, it is essential that the user interface take advantage of proven sophisticated technologies such as advanced graphics, animation and natural language.

Graphics may be used in any of a number of ways to support the Space Station mission, including map generation, reading and analysis, decision support aids, teleconferencing, modeling and simulation, and the generation of forms, reports and presentations. This variety of applications places special requirements on the graphics functionality. Functional requirements can be separated into graphics output capabilities, graphics input capabilities, and the storage, retrieval and transfer of graphics information. An excellent detailed discussion of the classes of graphic interaction and techniques to support them is provided in [FOL84].

Output capabilities needed include support for display of chart, graph and other two-dimensional diagrams, display of image data, and support for high quality typography, a variety of colors in display output, and animation (discussed in later paragraphs). The use of color has evolved so rapidly that most systems do not effectively use it. Useful guidelines for the exploitation of color can be found in [MUR84]. It is expected that dynamic display capabilities, such as "zoom," "shrink," "pan," and "highlight" will be needed also.

Command and control applications typically require a significant amount of user interaction. Graphics capabilities must support interactive text entry, input of data in video, fax, or digital format, the development and management of menu-driven systems, the development and management of sophisticated multi-window applications, and screenpainting. The use of menus and windows is discussed in later paragraphs.

While pictures may be created, displayed and discarded "on-the-fly," it is often desirable to store pictures or portions of pictures for later use. Without a storage and retrieval capability, the reuse of a commonly needed picture (such as a map) would have to be accomplished by including the map-drawing program in every new application. A better method is for the application program to simply request that the map be retrieved from the common database and displayed. In addition to storage, the capability to transfer a picture from one system to another must be provided. Support must be provided for the generation and interpretation of pictures expressed according to a particular protocol.

A number of national and international graphics standards exist, the benefits of which are independence of application programs from device and vendor dependencies, thereby improving the portability of application programs and data. These standards represent a hierarchy of graphics software layers which can be compared to the Open Systems Interconnection (OSI) model for

communications software. [HIN84] Among the most promising standards are the Graphical Kernel System (GKS) and Programmer's Hierarchical Interactive Graphics Standard (PHIGS). GKS functionality ranges from simple passive output to complex interactive graphics, and developments are underway to support three-dimensional graphics (GKS 3-D). ANSI language bindings, including Ada, are part of the GKS standard, and Ada implementations of GKS exist. [LEO85] PHIGS was designed to be upward compatible from GKS and an Ada binding for PHIGS has been developed.

Personal workstations with high resolution bit-mapped displays continue to decrease in cost, making available and very attractive the creation of animated drawings. Animation is a wonderful technique for illustrating dynamic objects and their actions. A very interesting discussion of the use of animation within a programming environment is provided by London and Duisberg. [LD85] The authors describe the need for an animation toolkit - a set of easily learned, easily applied, portable animation routines to relieve the tedious programming associated with computer animation. Such a toolset would include a library of reusable and connectable animation routines for creating new views. An animation toolset should be provided in the Space Station Development Environment along with packages for developing menu- and window-based systems.

Menu-driven systems have become commonplace for command and control applications. For the number of options and the sizes of databases we anticipate for the Space Station computing system, typical tree-structured menu systems will not be sufficient. The problem with these menus, which typically have numerous entries, is that they consume precious screen space and force the user to spend valuable time searching for a particular entry. Popular mechanisms to solve this problem include partitioning of entries according to logical function, pop-up submenus for related but more specific entries, paged menus and scrolling menus.

In any menu system, accomodation must be made for both naive and expert users, which implies that there must be an alternate route for commanding or selecting entries. Experience has shown that expert users will memorize such alternate routes to avoid using menus whenever possible. These alternate routes should be made obvious in training and documentation and designed to be consistent across the user interface. With an intelligent user interface which understands the user's role and experience level, continual customization and optimization of menus could be made (tailoring to the user profile). For example, reorganization of menu entries in accordance with frequency or infrequency of use may be in order. Existing menu systems used by NASA, such as the Transportable Applications Executive (TAE) do not include these flexible and intelligent capabilitites. [TAE85]

Multiwindow communication is desirable in situations where the user is concurrently performing many tasks. An example would be monitoring and analysis in a mission control center where displays are updated simultaneously by one or more real-time processors. Windowing capabilities are provided by a window manager, which both presents information in windows and allows the user to manipulate windows. Most window management systems fall into one of two categories: "tiling" or "desktop." Tiling involves arranging windows so that no overlap of windows occurs on the display screen (tiling is used in the Xerox Cedar System and the Microsoft Windows system). A desktop window manager does exactly the opposite - rectangular windows overlap like pieces of

paper on a desk (the Smalltalk environment developed at Xerox PARC demonstrates a desktop window manager).

There are advantages and disadvantages to both types. Desktop window managers offer the user the most flexibility in arranging windows but at the same time require the user to perform an inordinate number of functions relating to the rearrangement of windows. The tiling model relieves the user of most of the window management functions but typically performs automatic resizing and rearranging which may not be suitable or desirable. Perhaps the best choice is a combination of the tiling and desktop schemes, where the desktop model is employed when the user is performing many different tasks at one time, and the tiling model is employed when the user is coordinating many views or actions to accomplish a single goal. It is essential also that the user be able to easily and quickly move, size, and cover windows, and be able to move information from one window to another (cut and paste operations). The proposed intelligent user interface could assist in the manipulation of windows by understanding the application domain and choosing sizes and arrangement as appropriate.

Some user interfaces employ the use of icons in conjunction with windows. Often, icons are used to symbolize available software utilities (such as mail) and document folders. Icons could be used to provide the user with valuable information regarding the context of his working environment. For example, for each window an icon could be provided which tracks the progress, associated files and problems with the window's associated task. Such information assists users who may otherwise lose track of what they are doing. This useful concept is illustrated in the PERQ Sapphire window manager. [SAP] Because of the space station computing system's projected international use, the use of icons may be helpful throughout the interface, although care must be taken not to use an icon which is culture-peculiar (e.g., a "mailbox" may not be very communicative outside of the U.S.).

Requirements for Intelligence in the User Interface

The diversity of users and missions for the space station presents a formidable challenge in the design of a generalized user interface executive. Current technologies in natural languages and expert systems point to numerous potential instances whereby the performance of the user interface could be significantly enhanced through the addition of intelligence.

One significant opportunity for improving user interfaces discussed previously (especially for naive users) is to incorporate natural language interfaces. Currently, we do a reasonably good job of literal interpretation of English sentences in static contexts and limited, well structured domains of application. [ITW83] Yet many of the natural language technologies can also be applied to intelligent command interpreters and query processors. As a result, significant benefits can be realized by both the inexperienced or casual users (e.g., payload telescience) and by highly skilled operations, test and integration personnel.

An intelligent user interface could translate loose or shortened queries or commands provided by the user into correct and fully qualified messages to the space station computing systems based on stored knowledge of:

- o Missions
- o Individual user roles within the missions
- o Operational environment configuration
- o Operational environment state
- o Individual user characteristics
- o User's current context

Stored knowledge of space station missions would define the underlying bases for communication by:

- o **Establishing the vocabulary**, including abbreviations, acronyms, synonyms, generalizations, set memberships, abstractions, type inhertances, etc. [BRA83]

- o **Defining acceptable actions**, that would include prepatory decisions, test actions, main goals, cautionary actions, concluding actions or enablement actions. [GAL84]

- o **Establishing thematic role frames**, that specify anticipated or allowable action themes involving the thematic object being queried or commanded, the agent for action, instruments involved in the action, along with action descriptors including source and destination, trajectory, location, time or duration. [WIN84]

This level of knowledge allows robust interpretation of queries or commands, whether provided by natural language input or by more structured language-based inputs. Additional capability can be added to the user interface if knowledge of individual user roles within the missions are also provided. This would allow the user interface to:

- o **Restrict user actions**, providing another level of security at the user interface.

- o **Forgive erroneous or flawed input**, now that the bounds of an individual's interaction is known.

- o **Provide more power** in the user interface by calling up scripts [SHA85] of frequent or allowable action sequences.

Knowledge of the environment configuration and the environment state would allow yet another level of user input checking. Because ground-based commanding of space station operations or payloads must bear the response delays of ground routing and satellite links, it is desirable to provide the maximum amount of user input checking at the point of input. Certainly, one would want to restrict any actions that are dangerous or detrimental to the platform, payloads or mission. While there would undoubtably be checks made at the point of commanding, an additional layer of user input fault tolerance is often necessary.

Knowledge and observation of user characteristics would allow static and dynamic tailoring of the user interface for individual users. The most obvious application would be in acknowledging or inferring the skill levels of users and modifying the user interface input and presentation modes accordingly. This would circumvent the frequent problems associated with powerful interfaces being hard to learn and "user-friendly" interfaces getting

in the way of experienced users. In addition, the user interface could employ selective dissemination of information techniques which can dynamically tailor the method presentation (e.g., text vs. graphics) to the user profile [ITW83] or selective omission of information techniques (e.g., abstraction, indexes, summarization) [WIN84].

The technologies exist today to provide the capabilities described above in a cost-effective, low risk and timely fashion for space station. Over time, the completeness and robustness of the knowledge base would improve, providing an increasingly powerful user interface.

In the long term, knowledge of the users context will provide the most significant improvement in the user interface. Here, the intelligent user interface would attempt to understand the users intent and recognize the plan being pursued. In this mode, the user interface would constantly be analyzing a user's interaction in light of his role and in light of the state of the environment to answer the following questions: [SHA85]

- o Why is this character doing what he is doing?
- o What are is motivations?
- o What are his plans?
- o What's his intention?

To accomplish the understanding of a user's context, additional research is needed in concept modeling and reasoning about goals and actions of rational agents (i.e., the user). [ITW83]. Once achieved, the user interface could become an active element of the user-computer dialog, instead of a passive one. This could be most important when responding to emergency or abnormal circumstances where the user must quickly take some form of alternate action. In such a situation, the user does not have the time (or often the presence of mind) to completely describe a new course of action. Instead, he would leverage the machines knowledge of context and succinctly execute a new set of actions, such as: [HAM84]

- o Use alternate agent to accomplish the goal
- o Use alternate plan
- o Execute script rapidly
- o Wait out current state
- o Jump into the middle of the script
- o Counterplan against a potential future state
- o Put up with it
- o Recover

In addition to augmenting the operational user interface with knowledge-based capabilities, there is also significant potential for assisting the dialog design, prototyping and management tasks with expert system capabilities. Mission and user role knowledge can assist in simulating user interfaces and rapid prototyping. An expert system could assist the dialog authors in selecting interaction approaches and configuring the user interface accordingly. Similarly, there is a potential for assisting in the interpretation of user interaction data and metrics and in suggesting improvements or explaining perceived behavior. [ITW83]

Although there has been only limited work in incorporating artificial intelligence technologies into intelligent user interfaces, we are convinced that there is significant potential, particularly for a program as complex as space station.

User Interaction Design and Validation

A real challenge lies in combining these user-interaction capabilities in a way that is consistent and conherent for the particular users, their tasks, and their environment. This is an especially difficult task because capabilities or features which may be desirable for one class of users, type of activity or operational environment may not be for another. For example, features which support ease of learning are needed for the inexperienced or infrequent user while features to enhance efficiency and power are likely to be far more important for experienced or every-day users. The characteristics of the operations or tasks that are carried out have implications for user-interface design as well. For tasks that are repetitive in nature (e.g., data entry and text editing) efficiency of physical actions (such as number of key strokes) is important. For tasks requiring a high level of mental effort, minimizing the user's mental load and reducing errors is more important.

In short, the features required to support effective user interaction can mean different things for different users and types of tasks. In addition, users themselves are not static entities. Ideally, one would like user interfaces which evolve as a given user gains experience and sophistication, both with the task and with the computer system.

The major components of effective user-interface design include:

- o the ability to evaluate key features of user interfaces, especially at an early point in the development
- o the availability of a toolset to support user-interface development
- o a system architecture which allows development of user interfaces to proceed independently and in parallel with development of the rest of the system.

Evaluating User Interfaces

The design of user interfaces should proceed in a much more iterative fashion than the design of other parts of the software. There are too many unknowns concerning which combination of user-interface capabilities will best suit the various types of users, their tasks and operational environments. At the same time, there are few design principles to which a developer can turn for concrete guidance. Even obviously important principles such as "consistency" can be difficult to apply in practice since the designer's concept of consistency may not fit the users. Design decisions which seem obvious to the developers can lead to confusion among users.

User behavior can be a valuable source of guidance in selecting user-interaction capabilities. By observing how users accomplish a given task, what errors they make, how much time they require, and so on, the designer has an objective and meaningful basis for choosing among alternatives and for confirming the usability of choices already made. The earlier one can begin

to gather this type of information the better, using prototypes and simulations to test out design alternatives.

These evaluations can range from informal observational studies to formal standardized experiments. If the evaluation is concerned solely with identifying the strengths and weaknesses of a single design, then an informal observational study is sufficient. If the purpose of the evaluation is to compare alternative designs, then one must turn to the methodology of controlled experimentation, using a standard set of procedures in order to produce as unbiased an evaluation as possible. In either case, in order for the evaluation to be valid, the users, their tasks, and surrounding conditions must be representative of those that will be supported by the operational system.

Simulations may be of special interest in the design of the space station because they can be used to evaluate user-interaction capabilities that do not yet exist, thus providing information about the likely benefits resulting from various technologies that may require substantial resources to implement. Gould, Conti, and Hovanyecz [1] carried out this type of study by simulating a "listening typewriter" that could take human speech as input and produce a printed version of that speech as output. A human typist hidden from view simulated the speech recognition capabilities required for the typewriter.

Tools for Developing User Interfaces

In light of the above discussion, facilities are needed for recording user interaction. The level of detail of the information captured can vary from an atomic level (e.g., every keystroke) to a much higher level such as the total time required to complete a given task or a summary of the different commands used. The level of detail will obviously depend on the question of interest. In evaluating a text editor, for example, one may wish to log a time-stamped record of all keystrokes. Tools are also needed for prototyping user interfaces not only in terms of static displays but in terms of the dynamic aspects of an interaction as well.

Tools and associated databases are needed to assist in defining user-input languages and in creating, editing, and storing displays and display definitions of all types including graphics, text, animation, menus, and forms. The Dialogue Management System (DMS), developed by Hartson and his colleagues [3], contains many of these capabilities.

Architecture for the Intelligent User Interface System

Given the iterative nature of user-interface design, one of the key properties desired of user interfaces is flexibility. The space station computing system must allow changes in user interfaces as a result of improvements suggested by user testing or the addition of new users, new operations, or new operational sites. These changes must be made easily, quickly, and without adversely impacting other parts of the software.

Software designers have traditionally isolated the software from the effects of hardware changes. In the same way, the computational or functional portion of the software should be isolated from changes in the portions controlling the user interface. Hartson and his colleagues [3] have written extensively about the architectural issues involved.

Communicating with the user, including all input checking, should be the responsibility of the user-interaction components while the correct and efficient functioning of the system functionality should be the responsibility of the computational components. Hartson has argued for a parallel separation in the skills required to design, implement, and test these two components with the user interface falling within the domain of the human-factors specialist and the computational portion belonging to the traditional software designer and programmer. Once the interface has been defined between these two components, the two types of specialists can work independently and in parallel without interference.

Because of the complexity of such a user interface and the complexity of the various missions or roles, it is necessary to develop a support system for the user interface. We propose a comprehensive user interface system consisting of the following:

- **User Interface Executive.** This Ada software package would provide the user interface utilities embedded within the operational space station computing systems and be configured for the specific machine and missions for each installation via a resident database and knowledge base.

- **User Interface Prototyping Subsystem.** This would bundle the user interface executive with generalized simulation capabilities and data monitoring and collection routines. This subsystem would provide a pnambic ("pay no attention to the man behind the curtains") laboratory for user interface experimentation.

- **User Interface Configurator.** This software would customize the user interface executives for installation and also the prototyping subsystem for experimentation. The customization provided by this tool would include input and presentation options and the higher level dialog customizations. The low level customizations would be accomplished through a combination of program directed software builds from a library of user interface primitives and parametric or language-driven initializations. Higher level customizations will be accomplished through a rule compiler that will configure the user interface executive with the required knowledge and inference algorithms.

- **Dialog Management Subsystem.** This subsystem would input knowledge regarding the mission, users, configuration, etc., and be used to compose and configure dialog sessions. Data received from the user interface prototyping subsystem would be analyzed to validate dialog session before deployment.

The user interface system would be designed with an open architecture to allow easy expansion as new user interface technologies become available. The approach discussed in this paper will naturally put greater requirements on the local processing capabilities of the user interface devices. Current declining cost trends in high resolution graphic workstations leads us to believe that the increased functionality received from an intelligent user interface will be a cost effective solution for the space station. In addition, the proposed open architecture will lend itself to additions of new technologies over the space station life cycle.

References

[BAS85] Basili, V. "A View of Language Issues", presented at Open Forum on Space Station Software Issues, Apr. 25, 1985.

[BOE85] Boehm, B. "A View of Software Development Environment (SDE) Issues", presented at Open Forum on Space Station Software Issues, Apr. 24, 1985.

[BRA83] Brachman, Ronald J. ,"What IS-A Is and Isn't: An Analysis of Taxonomic Links in a Semantic Network," Computer, IEEE Computer Society, Vol. 16 No. 10, October 1983.

[DOR83] Dorofee, A. and Dickison, L. "High Order Language: Second Level White Paper, Space Station Operations Working Group, KSC/DL-DED-22, Jul. 29, 1983.

[DOR85] Dorofee, A. and Dickison, L. "Space Station Operations Language System Requirements and Concept Definition (Preliminary)", KSC, Aug. 1, 1985.

[GAL84] Galambos, James and Black, John, Using Knowledge of Activities to Understand and Answer Questions, Yale University Cognitive Science Program, Cognitive Science Technical Report #29, August 1984.

[GCH82] Gould, J.D., Conti, J., and Hovanyecz, T. "Composing Letters with a Simulated Listening Typewriter". In Proceedings of the Human Factors in Computing Systems Conference, ACM, Washington, D.C., 1982.

[HAM84] Hammond, Kristian, Indexing and Causality: The Organization of Plans and Strategies in Memory, Yale University Department of Computer Science, YALEU/CSD/RR#351, Dec 1984.

[HJ85] Hartson, H.R. and Johnson, D.H. "Dialogue Management: New Concepts in Human-Computer Interface Development". ACM Computing Surveys, 1985.

[ITW83] Report of the Information Technology Workshop, National Science Foundation, AD-A144-212, 1 Oct 1983.

[SHA85] Shank, Roger C., Questions and Thought, Yale University Department of Computer Science, YALEU/CSD/RR#385, August 1985.

[SOW84] Sowa, John F., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley Publishing Company, 1984.

[WIN84] Winston, P. H., Artificial Intelligence, Addison-Wesley Publishing Company, 1984.

[YH86] Yunten, T. and Hartson, H.R. "A SUPERvisory Methodology Notation (SUPERMAN) for Human-Computer System Development". In H.R. Hartson (Ed.), Advances in Human-Computer Interaction, Ablex Publishing Corporation, 1986.

[HIN84] Hinden, H. "Graphics Standards Finally Start to Sort Themselves Out," Computer Design, May 1984, pp. 167-180.

[LEO85] Leonard, T. "Ada and the Graphical Kernel System," Ada in Use: Proceedings of the Ada International Conference, May 1985, pp. 136-150.

[LEO85] Leonard, T. "Ada and the Graphical Kernel System," Ada in Use: Proceedings of the Ada International Conference, May 1985, pp. 136-150.

[GAR84a] Garman, J. "Data Processing for Space Station", Oct. 18, 1984 (presentation)

[GAR84b] Garman, J. "Networking and Data Processing for the Johnson Space Center", Dec. 13, 1984. (presentation)

[HAL85] Hall, D. "Space Station and the Role of Software", presented at Open Forum on Space Staton Software Issues, Apr. 24, 1985.

[SAP] Myers, Brad A. "The User Interface for Sapphire," IEEE Computer Graphics and Applications, Volume 4, Number 12, December 1984.