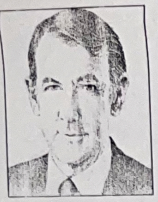
APPLIED INTELLIGENCE

SAA's Presentation, Dialog Interfaces Ease Programming



JAMES MARTIN

This is Part 6 of a series of articles on IBM's Systems Application Architecture (SAA). The introduction of integrated computing environments such as SAA will have a major impact on the software technologies of the '90s.

A chief benefit of integrated computing environments such as IBM's SAA is the provision of a standard user interface that makes all applications appear to behave in a consistent and familiar manner. The SAA component that governs the user interface is called Common User Access (CUA), which is based on the interface developed at Xerox Palo Alto Research Laboratories and popularized by the Apple Computer Inc. Macintosh.

CUA supports text-only functions appropriate for a non-programmable terminal and graphically oriented functions designed for a programmable workstation. With a workstation, the user has windows open for different applications or different parts of the same application. Menu bars with associated pull-down menus provide access to the various functions.

Users are in complete control of the interface. A mouse can be used to select a window or to move to another part of the screen. Users might open a new window or close an old one. Scroll bars might be used to access material that is too large to fit within the view displayed in the window.

This type of graphically oriented interface has proven comfortable for users and is easy to learn. It is based on the metaphor of the desktop: The user selects objects on the on-screen desktop and manipulates them in a consistent manner. Consistency means that scroll-bar operation and window manipulation be the same for all applications. The menus and multiple windows vastly reduce—and can even eliminate—the need for the user to learn any command syntax or to study complex documentation. There are no differing "modes" of operation with confusing behaviors. The user can move anywhere in the interface at any time.

Although this type of interface is desirable, programmers need to master not only a new set of tools but also a completely different manner of designing and building applications. The event-driven interfaces required to respond to user interaction are not only unfamiliar to most programmers, they are also hard to write.

IBM is attacking the problem with two programming facilities for building user interfaces.

The Presentation interface is the programmer's tool for building a full event-driven interface. It is a very powerful facility and is difficult to use.

The Dialog interface is a high-level tool that provides an easy way to implement event-driven interfaces for certain classes of applications. For example, the Dialog interface makes it easy to develop transaction-processing applications that give the user filled-in forms for gathering data to maintain a database. It could not be used for applications that require more intensive user interaction, such as text processing, or for applications that require high-speed graphic interaction.

The main strength of Presentation Manager's interface is the freedom it gives the user to control the interface to applications. The interaction might be application-specific, such as entering requested information, or it might occur at system level, such as when moving windows or initiating another application.

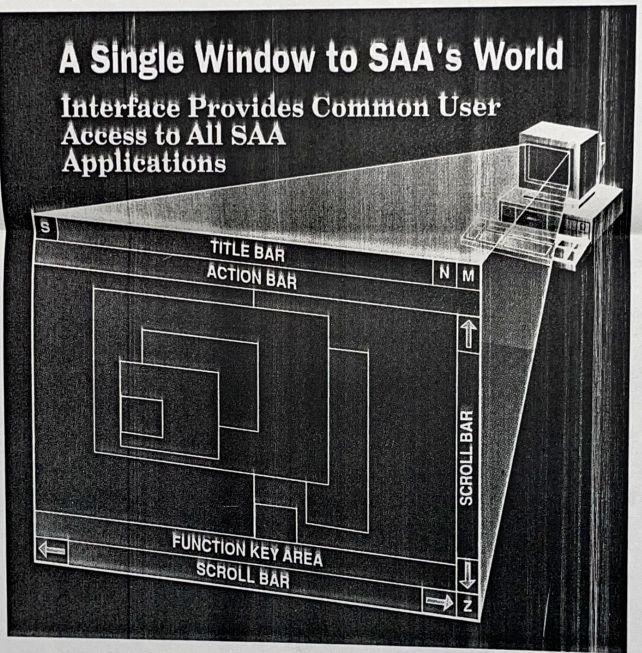
The main difficulty in developing an application controlled by Presentation Manager is providing the intelligence to deal with all possible user actions.

Using the Presentation interface, the highest level of organization in the user-interface program is the dispatch loop that recognizes certain user actions and then calls the appropriate modules for processing.

This is in sharp contrast to conventional applications that are organized by application function. When a function has to communicate with the user, it initiates a dialogue with the user. The dialogues are scattered throughout the application.

Presentation Manager requires that user I/O be the highest organizing principle, with subservient application functions. In conventional applications, the function is considered to be the highest organizing principle.

When function comes first, the user is at the mercy of the application and is forced to respond with a specialized command syntax that is unique to the program. The user can do only what is allowed by the application at that point. In general, the user often develops a helpless, anti-computer, frustrated feeling.



John Avakian

The main strength of PM's interface is the freedom it gives the user to control the interface to applications. The interaction might be application-specific or at a system level.

Programmers are very happy with the concept of function first, because the logic of the program is cleanly laid out in the application. While the user might be unhappy, the programmer is satisfied with the logical structure.

When user interaction comes first, the user's attitudes are completely different. All the functions of the machine are always available, and the user decides what to do and when. The psychology behind the interface leads to happier and more productive users.

No matter how it is organized, communication between user and application is a headache—either for the user or for the programmer. It is often difficult to decide whether to make an investment in training to use Presentation Manager. Although the benefits to users are demonstrable, it is not clear if these benefits are worth the cost in training and development.

Not only is it difficult to build applications with Presentation Manager, but due to the fundamental reorganization of application logic, converting existing applications to a Presentation interface may also be difficult. The real power of the Presentation interface is available only in C, so programmers must learn that language as well.

The difficulties inherent in converting to a Presentation interface are not appealing to large, conventional COBOL programming organizations; however, for those organizations willing to make the commitment, there are tremendous advantages.

The easiest way to bring this new technology into conventional organizations is through the use of higher-level tools. A layer of development software must be built that lets programmers concentrate on the application logic, not the user interface. The tools should automatically build the interface for the application.

Dialog Interface: A First Step

The Dialog interface, built atop the Presentation interface, is a first step in that direction. As with all high-level programming tools, it automates the development process for a restricted class of applications.

The programmer specifies, in a non-procedural manner, the attributes of a dialogue, such as menu entries and fields in a form. The dialogues are compiled separately. The programmer simply calls them when needed. The Dialog Manager run-time system handles all of the top-level control necessary for delivering an event-driven user interface.

The programmer uses a conventional development style that maintains program function as the highest organizing principle. When communication with the user is required, the Dialog Manager is called with the name of the dialogue.

Use of the Dialog Manager still requires some redesign of an application. A conventional full-screen application requires the user to flip from panel to panel. In a windowing environment, this is no longer necessary. An application that was originally composed of separate functions can be better designed as multiple applications, each in its own window, operating within a workstation environment.

Next week, I'll describe the application-generation environment provided by SAA. ■

To learn more about the subject of these articles, please call The James Martin Report, an information service updated quarterly, at (800) 242-1240. For information on seminars, please contact (in the United States and Canada) Technology Transfer Institute, 741 10th St., Santa Monica, Calif. 90402 (213) 394-8305. In Europe, contact Savant, 2 New St., Carnforth, Lancs., LA5 9BX United Kingdom (0524) 734 505.