Behavior Research Methods & Instrumentation
1980, Vol. 12 (2), 96-102

SESSION III
PRESIDENTIAL ADDRESS

N. JOHN CASTELLAN, JR., *Indiana University, Presider*

"Basic black" renascent: A new wardrobe

DANIEL E. BAILEY
University of Arizona Computer Center, Tucson, Arizona 85721

In the dozen years since publication of Uttal's fundamental paper, "'Basic Black' in Computer Interfaces for Psychological Research," many changes have taken place in on-line real-time computing. A look at the basic wardrobe today reveals some striking changes and certain fundamental constants. Most of the interface elements in the wardrobe specified by Uttal are still commonplace in the fashion catalogs of real-time computing. However, they are offered now as fully integrated circuits, chips, or subsystems that are added by inserting a chip into a socket or a printed circuit board into a chassis. Missing in Uttal's specification are the essential software tools for using the hardware of that wardrobe. The current real-time computing laboratory psychologist will find that the most significant elements in a really adaptable wardrobe are software tools.

One of the earliest uses of computers by psychologists was to control psychological experiments and collect research data on-line in real-time. In the time since this enterprise began, a decade and a half or more, a great transition has taken place in the fundamental wardrobe of on-line computing in the psychology laboratory. In order to set the tone of my retrospective look at on-line real-time computing, I have selected a paper that appeared in the first volume of the journal that prints the proceedings of the National Conference for the Use of On-Line Computers in Psychology. The paper is by William R. Uttal, who is perhaps the dean of on-line and real-time computing in psychology; he is certainly my nominee for that imaginary position. Uttal's (1968b) book *Real-Time Computers: Technique and Applications in the Psychological Sciences* was the first systematic description of scientific applications of computers for instrument control and data acquisition in psychology. In the same year, he published a paper from which the title of this paper derives: "'Basic Black' in Computer Interfaces for Psychological Research" (Uttal, 1968a). Although that paper deals with interfaces only, it marks the place, the technology, and the concerns of a psychologist/computer scientist of the late 1960s.

Before going on, let me introduce younger readers to a great fashion idea that was prevalent at the time Uttal (1968a) wrote his paper: Any woman with a dress of simple design, black, generally free of adornment and decoration, along with a collection of various accessories, such as belts, scarves, jewelry, shoes, bags, and so on, was able to dress with style for virtually any occasion, formal or informal. The dress and accessories made up the "basic black" outfit.

To take a simple computer and to make it do all the things that a psychologist in a laboratory wishes to do, without resorting to resources that psychologists often do not have, required a "basic black" approach to the computer and the interfaces to the psychological laboratory. In this paper I examine the status of the wardrobe today in the historical perspective of "basic black" of the middle and late 1960s. We are entering the 1980s, and a look back before an examination of the present seems to me appropriate.

"UTTAL BASIC BLACK"

Uttal's (1968a) paper begins with a definition of a set of logical symbols taken from electrical engineering: flip flops, inverters, "or" gates, and "and" gates. The logical character of interfaces and devices is the topic of this paper. However, Uttal warns of "other complications," such as voltage levels, rise time, polarity, and so on, lying in wait of the unsuspecting psychologist. Figure 1 is taken from Uttal's (1968a) paper: basic symbols used in describing basic interfaces.

The view of computing interfaces at the level of inverters, flip flops, "or" gates, and "and" gates is an abstraction. The electrical and electronic specifics of implementing the basic logical design is not presented, although physical meaning of certain interfaces could be developed in terms of the symbols involved. Certain concepts sneak into Uttal's (1968a) paper to reveal the

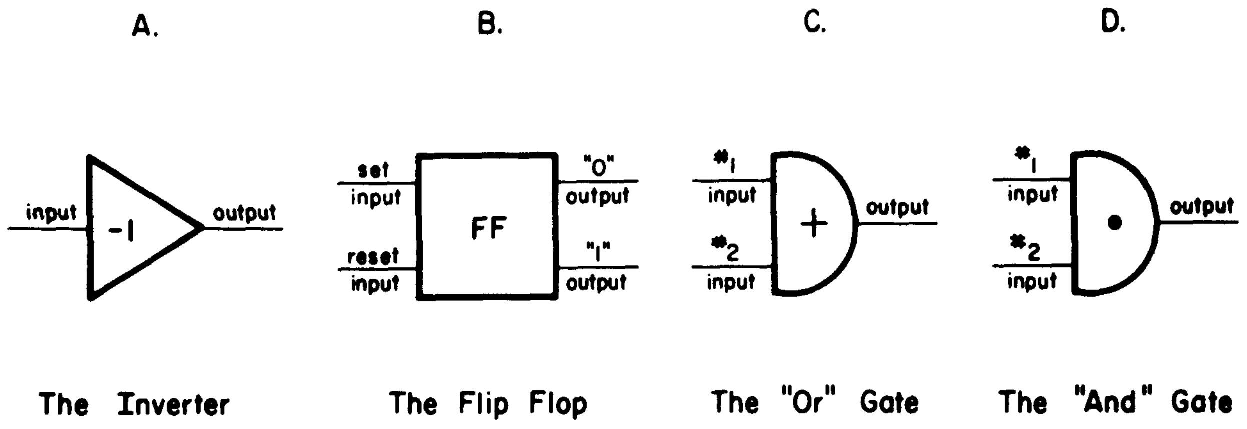

Figure 1. Symbols for the four basic components used in computer logical design. (Reprinted, with permission of the Psychonomic Society, from Uttal, 1968a.)

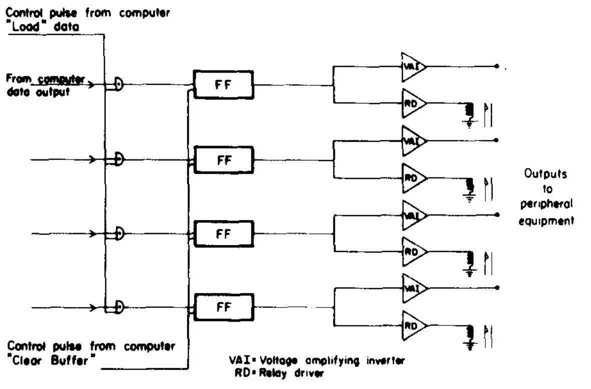
Control pulse from computer "Load" data
From computer data output
FF
Outputs to peripheral equipment
Control pulse from computer "Clear Buffer"
VAI = Voltage amplifying inverter
RD = Relay driver

Figure 2. Multiple power-level output interface device. (Reprinted, with permission of the Psychonomic Society, from Uttal, 1968a.)

detailed level of electronic implementation just below the surface of Uttal's presentation of logic. For example, he describes a couple of devices that are not presented simply at the logical level, such as the multielement power output interface (relay drive). Figure 2 is Uttal's representation of the driver, introducing several new symbols and a differentiation in inverters not needed elsewhere in the paper.

The elements listed in the wardrobe specified by Uttal (1968a), past the component level and past the logical level of flip flops and gates, are still in the latest wardrobe. The list shown in Table 1 is as modern as 1980, and it is safe to say that it will probably not grow old because the functions are so elemental and basic.

What has changed is the manner in which the present-day psychologist setting up a real-time computer laboratory will implement these elements in his computer system. Now, instead of building the devices, the laboratory computer system builder will insert a single integrated-circuit (IC) chip into a socket provided by the computer manufacturer, or at most, insert a single printed circuit board into the chassis of the computer system. Attaching the external cable is perhaps the most difficult part of putting together a basic black element—that is, providing the connector on the computer chassis or on the cable to the outside environment.

Indeed, the designer may have many options to choose as standard modules (chips or boards to plug into the system): voltage level outputs to interface with a variety of noncomputer equipment, high-voltage solid state relays, many types of analog-to-digital and digital-to-analog converters, digital inputs and outputs, relays, and so on, in single-line and many-line combinations.

New kinds of elements might be added to the basic black in computer interfaces, but the instrument control and data acquisition functions specified by Uttal (1968a) are still fundamental and basic. In addition to the basic

Table 1
"Basic Black" Computer Interfaces

STIMULATOR TRIGGERS TO PROVIDE CONTROL OF EXTERNAL STIMULUS EQUIPMENT.

RESPONSE SWITCHES TO DETECT SWITCH CLOSURES, KEYPRESSES, ETC. IN THE EXTERNAL RESPONSE EQUIPMENT.

MILLISECOND REAL-TIME CLOCK TO MEASURE LATENCIES BETWEEN EVENTS, AND TO CONTROL DURATION OF EVENTS.

DIGITAL TO ANALOG CONVERTERS TO PROVIDE FOR TRANSLATION INTO ANALOG FORM DIGITAL DATA FOR USE AS STIMULUS CONTROL AND INSTRUMENT CONTROL.

ANALOG TO DIGITAL CONVERTER TO CONVERT CONTINUOUS VARIABLE SIGNALS TO DIGITAL FORM FOR DATA ACQUISITION PURPOSES.

MULTIPLE POWER LEVEL BIT CODED DISPLAY CONTROLLERS TO PROVIDE MULTIPLE DISCRETE PATTERNED OUTPUTS FOR STIMULAE OR CONTROL OF INSTRUMENTS.

functions listed by Uttal, we might incorporate many higher order functions that are achievable by "plugging in" devices that, from the designer's point of view, are simple, elementary units. For example, we have single-chip or board-level devices for voice synthesis, music generation, graphics control of videoscreens, character generation for CRT terminals, telecommunications, and so on.

It is just this proliferation and simplification of device controller, special device, and special function implementation, using the new large-scale integration technology, that prompts me to argue that the basic black has changed. No longer are the elements in the wardrobe hardware elements. The basic fabric is now changed.

"BAILEY BASIC BLACK"

In my renascent basic black, the interface hardware is presumed. The hardware interface to clocks, triggers, analog sources, discrete output lines, and so on, are "off-the-shelf" devices that plug into a chassis slot or an IC socket.

In my renascent basic black, we have an entirely new wardrobe to deal with: the user interface. While Uttal (1968a) was correct in focusing on the hardware interface to laboratory equipment, the 1980 focus should be on another variety of interface: the interface between the user and the computer system. The key element is a carefully selected set of software tools.

A number of years ago in the early days of the microelectronic revolution, there was a lot being said in the engineering literature about having to learn new technology, new methods. Many people were said to be in need of dramatic, extensive retraining to bring them into the modern era. At first I was puzzled by this clamor over the need to retrain technical people. The technology appeared to be straightforward computer technology based on a dramatic simplification of hardware. The physics and electronic implementation were different, but vastly simpler. Why then was it common to hear calls for retraining?

The microcomputer was being advanced as a replacement for hard-wired electronics with discrete components. The engineer was being told that he had to stop dealing mainly with electrical circuits and to begin dealing with logical circuits and programming. And the contents of the early papers and books on applying microprocessors were devoted mostly to programming. The engineer had to be trained in the old technology of programming. The puzzlement I felt when I could see nothing new was based on the fact that I began in 1955 where the engineer was supposed to be going in 1975–machine language programming.

The realization that microprocessors were forcing reinstitution of programming practices of "ancient" times was startling and dismaying. Few who went through the early days of machine language programming as a forced way of life would choose to repeat it. However, the point is clear–The major advancement has been in hardware technology, and little advancement has been made in programming technology. The low cost of microprocessors and microcomputers has lured countless individuals into the thicket that caught up some of us 25 years ago. It is painful to see valued friends and colleagues suffering the same rites of passage that should have been endured only by the early pioneers.

The computing profession has not made the fundamental advances that will be required before the Bailey Basic Black can be fully discerned in shape and texture. We have not learned how to program computers very well. And, when we learn to program these devices, we will find that the components in the basic black formula will have changed, in all probability.

An Overview

Figure 3 presents a simplified block diagram of the elements in the Bailey Basic Black. Some of the blocks are shaped like "cans" or "pots" to imply that the way these elements are made available is of significant importance. These structures are elements in the wardrobe, but they will not appear in each and every costume. Rather, they are stored and available for use by the real-time programmer and computer user. The image implies some rotating mechanism like a disk, but that is only an epiphenomenon of current technology. The important aspect is that the wardrobe have these large storage devices to contain, in a dynamically retrievable manner, all sorts of things to use in piecing together the desired effect.

The rectangles generally represent processes instead of objects, although the results of running these processes may be objects. One of the rectangles has special significance: the rectangle with the radiating arrows, labeled "control command structure." This box represents the high-level capability of the user of a computer system to govern the behavior of the entire system. It is the primary user interface, within which lie the major determinants of success, failure, and quality of work. The quality of the elements in this primary user interface determines, to a major extent, the quality of satisfaction of the user's needs.

The control command structure provides for access to all of the major utilities in the operating system, the libraries, and certain portions of the operating system. It is with this command structure that the user orders up services and activities and makes use of the basic structure of the computer system. The specific manner in which this is used depends on the choices made by the system designers. However, we do not depart from generality by assuming that the facilities within the control command structure are part of the "operating system," that part that is used more or less directly

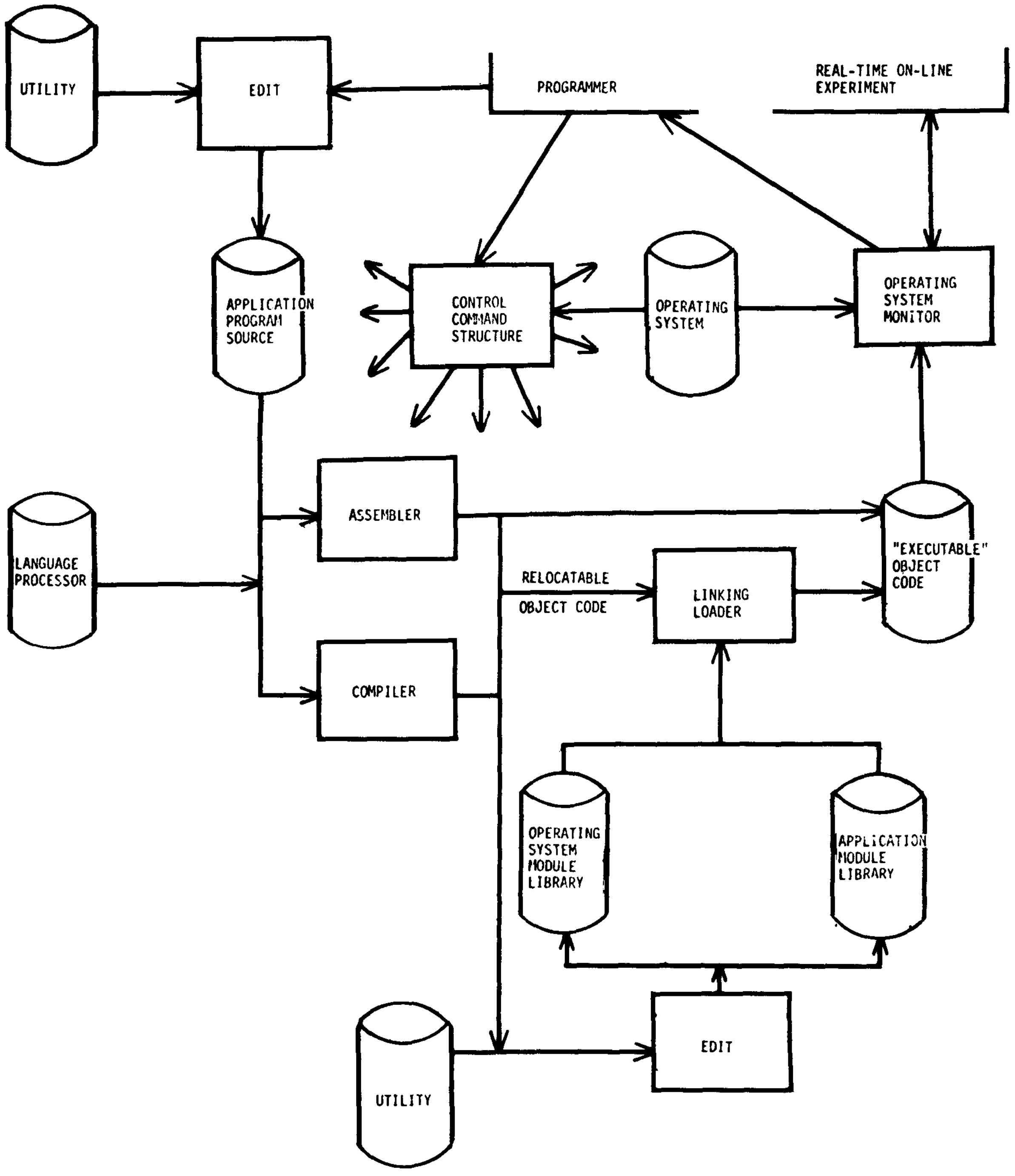

Figure 3. "Bailey Basic Black" in ensemble.

by the programmer/user of the system. As such, it is not listed separately in the elements of the Bailey Basic Black.

The Elements of Bailey Basic Black

The several distinct elements in the Bailey Basic Black, listed in Table 2, are to a large extent familiar to anyone who has had contact with modern large-scale computers or well developed minicomputer systems. The point of my presenting these concepts here is not so much to instruct computer users in what they already know. However, it is to point out and to be explicit and clear about how much our wardrobe has changed in the short span of a dozen years.

On the other hand, we are entering an age of déjà vu, where the old hands in the computing game are seeing again our colleagues struggle with computer systems without all of the necessary elements. Now, however,

Table 2
Elements in the "Bailey Basic Black"

ELEMENTS IN THE BAILEY BASIC BLACK
TEXT EDITOR
FILE EDITOR
ASSEMBLER
COMPILER
LINKING LOADER
OPERATING SYSTEM, MONITOR
YE OLDE TOOL SHOPPE
YE OLDE TOOL SHOPPE
OPERATING SYSTEM MODULE LIBRARY
APPLICATION MODULE LIBRARY

it is not the hardware that is largely lacking—that one can buy. But software is lacking.

I must qualify my comments and warn that there is a good chance that the hardware available may not perform in a way such that software can be written to make the hardware usable for a given real-time on-line application. For example, there are floppy disk drives available on the market that can be attached to a number of microprocessors. However, on close observation, it appears that some of these drives do not have basic interrupt capabilities, so that a microprocessor must depend on continuous sampling to know when the disk is ready to deliver or receive data. This is a limitation that could be sorely felt in a number of real-time applications. While the software of the microprocessor is trying to assess disk readiness, events in the outside world may be missed, or when events in the outside world are being observed, the disk processes can be missed.

Newcomers to the world of on-line real-time computing are starting with primitive software and operating systems. Microprocessor software systems are at the stage of development that minicomputers went through a decade ago or more. One may be able to purchase a microprocessor system for a small number of today's dollars, but the real cost is in terms of yesterday's pains and agonies when one attempts to make use of the computer and device interfaces through the software that is available. Largely, the hardware threads are all there, or they can be with only a small amount of perceptive caution. It is the software that needs careful evaluation.

Table 2 contains a list of all of the basic elements in computing software. However, the integration of the elements and the flow of control from element to element is not evident from the list. Figure 3 is a graphical attempt to illustrate the flow of control and information in the Bailey Basic Black, and hence its integration.

Text editor. This component, generally supplied as a utility in the general operating system package provided with a computer system, permits the user to compose program code, messages, and other text-like objects and to manipulate these objects. The general intent is to work on a limited range of material, but with powerful operations available to manipulate that material. There possibly are as many text editors as there are computer programmers with more than 1 full year of experience, reflecting the inadequacy of all of the editors that have been invented. Minimal sets of operations in text editing are simple to specify and are small in size. They include operations such as insert, delete, locate, reset line pointer, replace, input, output, and so on. The way that the operations are referenced and parameters of the operations are specified are perhaps as important as the operations provided.

File editor. The object of a text editor is to work on a single entity, which eventually becomes a file, regardless of the computer medium that is used to store the file. A file editor is a utility that operates on collections of files, perhaps modifying individual files in the process. The functions incorporated vary over a wide range. The minimum set of file editor functions needed for such operations includes defining new files, deleting files, concatenating files, inserting one file into another, copying contents of a file into another file, and so on. Again, the user interface, how the function is referenced and how parameters are specified, is as important as the functions provided.

Assembler. In the modern world, one might wish for an assembler-free computer system, but such a wish is without practical justification, even in 1980—and will probably still be so even in the fabled 1984. However, in some of the most modern microprocessor systems, the most advanced and adequate software available is an unsophisticated assembler. In some systems, the only software provided is an interpreter of one sort or another, with little or no way to provide elements in a user-developed tool shop (see below). Consequently, the user with such a system is limited to situations covered by the wisdom and forethought of the system designer and builder. As soon as an application falls outside of those bounds, the user needs to have tools to deal with circumstances and functions not anticipated by the original system designer. Perhaps an assembler is not the most desirable systems programming tool, but it may be the only tool that is appropriate to the task of dealing with the unanticipated demand. In non-research application areas, such demands are not likely to be encountered. However, it is exactly in the domain of research where one does not have the luxury of doing everything the way that someone else imagined it was going to be done. And, so, the user of even the most modern computer technology must anticipate the need on occasion to resort to the arcane black arts, approaching the machine language of the computer, perhaps as far down as the assembly language level.

Perhaps when the appropriate high-level tools for systems programming are available in adequate quality, problem solving with assembly language will be unnecessary. The tools will probably come in the form of a compiler.

Compiler. Perhaps the most important single element in the Bailey Basic Black is a high-level compiler. The language is important. Of course, fads in computer science come and go, as in any other field. However, currently the PASCAL appears to be the emotional favorite for a higher level language. In the real-time on-line area, there is no language that supports the basic elements of the Uttal Basic Black hardware for real-time functions. However, the ease with which the language can be "bent" to the desires of the user and the applications can be used as a criterion of acceptability and quality of the language in the context of the application. The ability to define functions and procedures for hierarchical, structured development of a set of software tools is very much an asset, if not a necessary property of the language. The ability to support concurrent processes with the high-level language is also a very valuable asset, and a necessity if multiuser or multiprocess applications are to be implemented.

A critical feature that is easily overlooked is the ability to develop systems-level software in the compiler language. If the supplier of the system has not completely anticipated the needs of the research laboratory computer system, then the user is forced to supplement and augment the tools that are provided. In the previous paragraph, the availability of an assembler, and its use, were justified on the grounds that such systems-level work was likely to be necessary in research environments. It is much better if the compiler can support such work instead of requiring the use of an assembler. The same reasoning applies even more to user-developed software tools that play such an important role in productive and efficient use of computer systems in the laboratory.

Linking loader. A somewhat inappropriate term is used in the heading of this paragraph to refer to a number of software functions that prepare a program for execution in the context of specific features of the computer at the time of execution. Whether or not this function takes the shape of a linking loader, an operating system function that manages virtual memory, or an interpretative system is not really material. What is important is that the specific applications program written by the scientist in the laboratory be reasonably free from constraints of absolute computer structures, such as specific memory addresses and memory limitations. Also, an efficient system for referencing and using tools from the systems and applications libraries is vital for the completion of the programming efforts of the user.

Operating system. The operating system of a useful computer system will contain a large number of individual utilities and provide a large number of services to the user. Among these services, the first encountered is of course the control command structures that permit the user to access other utilities, start and stop processes, and so on. Other useful functions that might be important enough to serve as guides in the evaluation of the adequacy of this element are functions such as error logging, memory management, general resource management, and so on. Of course, as in all of the other elements, the user interface to the resources of the operating system ranks among the most important aspects of the operating system.

Ye olde tool shoppe. This is the crux of software issues in the user environment. A set of tools to do things with the hardware is the ultimate resource required. To a certain extent, all of the elements in the Bailey Basic Black are elements in the tool shop. However, more specifically, the tools that are of critical importance are the functions, subroutines, and procedures that act as elements in the programs produced by the user. The specific tools that are important in a given context will depend on the characteristics of that context. The tool shop for a real-time on-line research laboratory will certainly need tools to drive devices and hardware elements in a correct and efficient manner. Elements in the tool shop in this environment must be able to read real-time clocks, set time-out alarms, deal with interrupts signalling external events, collect data from hardware devices, and so on. In other words, the tool shop must be able to deal with hardware elements in the Uttal Basic Black and more.

In addition, the tool shop should provide for the development of new tools. The development of new software, even at the systems level, should proceed on a structured and hierarchical level, with each element simple, well-defined, and functionally clean.

The tool shop might reasonably be broken down into two components, one supplied by the manufacturer or vendor of the system, and the other developed within the specific applications context. In Table 2 these are referred to as libraries.

Operating system module library. A rich library of tools to make the computer perform useful things in general, regardless of the application context, is an important element in the basic black wardrobe. Such elements provide for utilization of the fundamental properties of the computer hardware, input/output devices, specialized facilities such as clocks, interrupts, and so on. In order for the elements to be more useful, their access, definition, and parameter structures should be as general as possible at the lowest level, perhaps with a hierarchical set of software interfaces to them at higher levels with more specific references.

Applications module library. The user of a system, or a collection of users, should be able to develop a rich library of tools out of the basic set of tools provided by the manufacturer or vendor of a computer system and reference those tools in a manner that permits them to be integral elements in the system. Generally, the user will develop tools that are oriented to application, rather than to the general operating system. However, the distinction is one of convenience, and separation of the tool shop into operating system library and applications library is to a large extent arbitrary.

The Ensemble

Putting all of the Bailey Basic Black together graphically yields Figure 3. The programmer or user orchestrates the assembly of elements from the collections of utilities, the operating system, the language processors, the editors, linking loaders, resource management components of the operating system, the tool shop, and so on. The consequence is a flowing of information from one point in the system to another point, and the processing of that information at selected points in the system.

Perhaps the most important aspect of the entire ensemble is represented by an inconspicuous arrow in Figure 3. The arrow leading downward from the language processors, the assembler and compiler, to the utility/editor line leading to the operating system module library and the application module library, signals the ability of a system to grow and to modify itself, to become ever more useful and flexible. This is an important differentiation between the Uttal Basic Black and the Bailey Basic Black. The Uttal (1968a) version presented a static, hard-wired world for real-time computing. The Bailey version presents a dynamic, self-modifying, growing, logic-oriented world for real-time computing.

The present state of the art provides a challenge: Make good use of the Uttal Basic Black. The challenge will be met with software, by the Bailey Basic Black, or some collection of elements similar to those described in this paper.

REFERENCES

Uttal, W. R. "Basic Black" in computer interfaces for psychological research. *Behavior Research Methods & Instrumentation*, 1968, 1, 35-40. (a)

Uttal, W. R. *Real-time computers: Technique and applications in the psychological sciences.* New York: Harper & Row, 1968. (b)