EXTENDING THE USER ACTION NOTATION FOR RESEARCH IN INDIVIDUAL DIFFERENCES

Derek Brock
Deborah Hix†
Lynn Dievendorf, Jr.
J. Gregory Trafton
Naval Research Laboratory
Washington, DC
†Virginia Tech
Blacksburg, VA

Software user interfaces that provide users with more than one device, such as a mouse and keyboard, for interactively performing tasks, are now commonplace. Concerns about how to represent individual differences in patterns of use and acquisition of skill in such interfaces led the authors to develop modifications to the standard format of the User Action Notation (UAN) that substantially augment the notation's expressive power. These extensions allow the reader of an interface specification to make meaningful comparisons between functionally equivalent interaction techniques and task performance strategies in interfaces supporting multiple input devices. Furthermore, they offer researchers a new methodology for analyzing the behavioral aspects of user interfaces. These modifications are documented and their benefits discussed.

Introduction

In the course of reviewing results from a colleague' s individual differences research, we became interested in the possibility of using a formal interaction representation technique to analyze and evaluate the software user interface being studied. Of particular interest was the representation problem itself due to the nature of the interface and the behavioral issues in question. Our colleague's research was aimed at investigating how individual user differences in cognitive skills and abilities relate to performance and learning strategies when using an interface that utilizes more than one input device, in this case, a keyboard and a mouse. Such interfaces generally provide a way, or ways, to use each form of input to accomplish many, if not all, of the utilitarian and high level tasks in a given application's domain, such as opening and saving files, selecting and manipulating data, and so on. These interfaces have the advantage of encouraging individual styles of use, but they also inherently engender complexity in the interface that can be difficult, at best, to represent in a form that is both immediate and useful.

After surveying a number of existing user interface representation techniques, we judged the *User Action Notation (UAN)*, developed at Virginia Tech , to be the most practical and malleable choice for specifying the sort of interface just described. In this paper, we describe modifications to the UAN that substantially increase its expressive power. The modifications allow tasks that can be performed in a number of different ways using different input devices to be represented in an efficient and straightforward manner. We also provide examples of how these extensions can be used to facilitate various aspects of individual differences research, allowing empirical data to be explored and described in useful new ways.

The User Action Notation

A distinction between behavioral and constructional domains (Hix and Hartson, 1993) has emerged in the study of user interface development. The *behavioral domain* emphasizes the user and the *constructional domain* emphasizes the implementation. Accordingly, *interface design representation techniques* in the behavioral domain specify the user's role in the interactive process, i.e., what a user can or must do to perform a task.

The User Action Notation, as its name implies, adopts the behavioral perspective and describes actions made by the user and the interface as they interactively perform a task together (Hartson, Siochi, and Hix, 1990). This *task* is the notation's defining abstraction. The sequences of user inputs and interface responses of which a task is composed are shown with other information in a task description table that employs a compact, mnemonically expressive specification language. Design elements better suited to expression at higher levels of abstraction (e.g., task hierarchies and context) are supported with additional elements in the notation.

TASK: save open existing file
USER ACTIONS
use mouse M: (2 clicks
| click-and-drag)
| use keyboard K: (AltFS
| CtrlS)

Figure 1(a)

TASK: save open existing file - M(2 clicks)	
USER ACTIONS	INTERFACE FEEDBACK
(~["File" in menu-bar]	
M(L v)	cursor ! "File" in menu-bar ! ∀ items in menu-bar ≠ "File": item -! display (File menu) ∀ items in File Menu: item -! display (File menu msg in msg line)
M(L ^))	cursor -! "New" in File menu ! display(New file msg in msg line)
(~["Save" item in File menu]	
M(L v)	cursor ! "Save" in File menu ! ∀ items in File menu ≠ "Save": item -! display (Save file msg in msg line)
M(L ^))	open file already exists: {erase (File menu) erase (msg line) display (request for information dialogue)}

Figure 1(b)

TASK: save open existing file - M(click-and-drag)	
USER ACTIONS	INTERFACE FEEDBACK
(~["File" in menu-bar]	
M(L v)	cursor ! "File" in menu-bar ! ∀ items in menu-bar ≠ "File": item -! display (File menu) ∀ items in File Menu: item -! display (File menu msg in msg line)
~["Save" item in File menu]	while cursor-@"Save" item in file menu: {item@cursor in File menu ! ∀ items in File menu-@cursor: item -! display (item@cursor in File menu msg in msg line)} "Save" in File menu -! ∀ items in File menu ≠ "Save": item -! display (Save file msg in msg line)
M(L ^))	open file already exists: {erase (File menu) erase (msg line) display (request for information dialogue)}

Figure 1(c)

TASK: save open existing file - K(AltFS)	
USER ACTIONS	INTERFACE FEEDBACK
(K(Alt v)	
K(F v)	"File" in menu-bar ! ∀ items in menu-bar ≠ "File": item -! display (File menu) "New" in File menu ! ∀ lines in File menu ≠ "New": line -! display (New file msg in msg line)
K(Alt ^ & F ^))	
(K(s v)	open file already exists: {erase (File menu) erase (msg line) display (request for information dialogue)}
KJ(S^))	

Figure 1(d)

TASK: save open existing file - K(CtrlS)	
USER ACTIONS	INTERFACE FEEDBACK
(K(Ctrl v)	
K(S v)	"File" in menu-bar ! ∀ items in menu-bar ≠ "File": item -! display (File menu msg in msg line) open file already exists: {erase (msg line) display (request for information dialogue)}
K(Ctrl ^ & S ^))	

Figure 1(e)

A UAN Example

Figure 1 gives a UAN specification for the user task of saving an open, previously existing file in a DOS-based graphing application named SigmaPlot™ whose interface employs many standard elements of a graphical user interface. As noted earlier, the UAN abstract unit specification is that of a task. The task shown here can be performed in several different ways. Each of the ways the task can be performed is composed of a sequence of user actions and interface responses and these are shown in individual UAN task description tables.

As can be seen, even a relatively simple task turns out to be surprisingly complicated to specify when two input devices are utilized. The description in Figure 1(a) says the user can use the mouse or the keyboard (use mouse M: {...} ... | use keyboardK: {...}) to perform the task and that each device provides two different ways to do this.

(Note: In their complete form, UAN task description tables are composed of four columns: "User Actions," "Interface Feedback," "Interface State," and "Connection to Computation" (Hix and Harson, 1993). Changes to the UAN in this paper are made only to the User Actions column. For this reason and our specific focus on the user 's role in the interaction equation, the figures used herein only show and use the first two columns.)

Figure 1(b) expands on the "2 clicks" strategy using the mouse. This specification is read from left to right, starting at the top, across the User Actions and Interface Feedback columns. The first line indicates that the user uses the mouse to move the cursor to the word "File" in the menu-bar at the top of the screen (~["File in menu-bar]). Nothing is specified in the Interface Feedback column since the act of moving the cursor with the mouse is an integral feature of the device. On the next line, the notation M(L v) indicates that the user then presses the left mouse button down. The interface responds in several ways: the cursor highlights by changing shape, the word "File" is highlighted alone in the menu-bar, the File menu is displayed (with none of its item lines highlighted) and a File menu message is displayed in the message line (at the bottom of the screen). The entire response is essentially instantaneous. On the next line, the user releases the left mouse button, M(L ^), which completes the first "click." The interface responds by unhighlighting the cursor (restoring it to its original shape), highlighting the "New" item in the File menu, and displaying the New file message in the message line.

Next, the user moves the cursor to the "Save" item in the File menu and begins the second click with another press of the left mouse button. The interface responds by highlighting the cursor, highlighting the "Save" item

in the File menu, and displaying the Save file message in the message line. Finally, the user completes the second click by releasing the mouse button again. In the Interface Feedback column, the interface response is predicated on the currently open file's having been previously saved. When this is the case, the File menu is removed, the message line is cleared, and a request for information dialogue appears in the middle of the screen. This dialogue prompts the user to verify the file's replacement or cancel the save and is not included in the specification because it is essentially a separate task.

Figures 1(c), (d), and (e) are read similarly. In the click-and-drag alternative, rather than immediately releasing the mouse button after clicking it to open the File menu, the user continues to hold the button down and moves the cursor to the Save item in the File menu, releasing the button there. In both of the keyboard alternatives, the user must use key combinations. The letter keys used are mnemonically suggestive of their function but must be pressed after the associated Alt or Ctrl key has been pressed. The notation then indicates the keys may be released in any order. A short table of UAN symbols and their meanings, as used herein, is provided at the end of the paper. A full treatment of the UAN is provided in Hix and Hartson (1993).

Discussion

Interfaces that utilize more than one input device are now commonplace and, as mentioned earlier, have the advantage of encouraging individual styles of use. In Figure 1, it should be noted that each of the alternative ways the task can be performed is functionally equivalent. This is inherently interesting to individual differences researchers because such functional equivalencies typically differ in terms of their comparative efficiencies and demands on users. What factors encourage some individuals to learn and make use of one strategy and others to choose a different one? Can predictions be made about learning, performance, and skill acquisition based on characteristics of interface design complexity and measures of cognitive skills and abilities?

To answer questions of this sort, it would greatly help to be able to visualize and therefore more easily compare advantages and disadvantages of one strategy versus another. The UAN specification in Figure 1 provides a comprehensive and articulate, if lengthy, description of all of the ways the task can be accomplished and how the interface will respond. But it is difficult to tell how the alternatives compare and interact by examining the figure. The information is available, but not obvious. A close examination of the Interface Feedback columns across the task description tables also reveals a great deal of duplication. These and other considerations led us to rethink the standard layout of the UAN task description table, and, in particular, the User Actions column.

Extensions to the UAN

Figure 2 introduces an alternative to the specification given in Figure 1. This new specification, motivated by the kinds of analytical concerns just raised, employs a small set of extensions to the UAN that substantially facilitate the analysis, specification, and comprehension of user interfaces with multiple input devices.

The first and perhaps most notable change made to the specification is to simply divide the User Actions column into individual, device-specific subcolumns. For comparative purposes, this makes the specification clearer by allowing each of the ways the task can be performed to be described concurrently. It also solves the problem of duplicating Interface Feedback column specifications for each alternative. However, using concurrent device columns raises the technical problem of indicating where functional equivalence occurs among sequences of user actions. What is needed is an appropriate, non-arbitrary level of granularity. This turns out to be the level of an interaction technique.

Interaction techniques are defined as how input devices are used to enter task-specific information into computers. They are identified as sequences of individual physical user actions (Foley, van Dam, Feiner, & Hughes, 1990). Buttons and keys, for instance, can be pressed, held down, and released; pointers moved in various ways; and so on. These are basic actions that form the fundamental capabilities of input devices. In software, these basic actions are interpreted as tokens of input (Jacob, 1986). Meaningful units of information are entered into a computer by sequences of user actions that combine to form specifiable interaction techniques. An interaction technique, such as a mouse click or the use of a key combination, is seen by the interface software as a series or combination of input tokens generated by the user that corresponds syntactically to some pre-defined sequence.

In the UAN, user actions are usually specified, not in terms of interaction techniques, but, rather, in terms of individual actions that are demarcated by horizontal lines, as in Figure 1. This is a strength that allows corresponding interface responses to be accurately specified. However, were it to suit some analytical or communicative purpose, the first two lines in the User Actions column in each of the task descriptions in Figure 1 (Figures (b) through (e)) could just as well be combined into a single line as no feedback is specified for the user action on each first line. In order to show single actions *and* individual interaction tech-niques, a UAN specification must resort to the already overloaded and confusing use of parentheses across lines in the User Actions column as a grouping

TASK: save file				
USER ACTIONS				INTERFACE FEEDBACK
mouse		keyboard		
~["File" in menu-bar] M(L v) M(L ^)	~["File" in menu-bar] M(L v)	K(Alt v) K(F v) K(Alt ^ & F ^)	K(Ctrl v) K(S v)	M: cursor ! "File" in menu-bar ! ∀ items in menu-bar ≠ "File": item -! -K(S v): display (File menu) -K(F v): display (File menu msg in msg line) M: ∀ items in File Menu: item -!) K(F v): ("New" in File menu ! ∀ items in File Menu ≠ "New": item -! display(New file msg in msg line)) M: (cursor -! "New" in File menu ! ∀ items in File Menu ≠ "New": item -! display(New file msg in msg line))
~["Save" in File menu] M(L v) M(L ^)	~["Save" in File menu] M(L ^)	 K(S v) K(S ^)	 K(Ctrl ^& S ^)	M(L v): while cursor -@"Save" in file menu: (item@cursor in File menu ! ∀ items in File menu-@cursor: item -! display (item@cursor in File menu msg in msg line)) M: (cursor ! "Save" in File menu ! ∀ items in File menu ≠ "Save": item -! display (Save file msg in msg line)) open file already exists: (erase (File menu) erase (msg line) display (information request dialogue))

Figure 2

mechanism as has been done in Figure 1.

The specification in Figure 2 takes all of this into account and neatly solves the problem of showing functional equivalence. Using horizontal lines in the User Actions columns of Figure 1 to demarcate individual user actions is discarded in favor of a new definition. Horizontal lines in the User Actions column now define cells which bound individual interaction techniques. At this level of granularity, functional equivalence among sequences of user actions is easily expressed both within and across devices. Within each cell, individual actions still appear sequentially on separate (though not drawn) lines and are placed so that they continue to correspond laterally to the appropriate interface responses shown in the other UAN columns. Functional equivalence is shown by adjusting a cell's vertical dimension relative to concurrent interaction techniques, interface response elements, and the overarching task.

Finally in Figure 2, the mechanism of individual device columns within the User Actions column has been taken a step further. Interfaces frequently offer more than one way to complete a task using the same device, as is the case here in our example task. In Figure 2, the individual device columns have been further subdivided as necessary in order to specify this.

An interesting and useful property of the arrangement of this new layout is that it provides a way to see at a glance when and if alternative ways of performing a task interact, that is, where a user may shift from one device (or interaction technique) to another while continuing with the task. This property is illustrated schematically in Figure 3: The user may shift from one device to another wherever the end of one cell and the beginning of another are coincident horizontally. The User Actions column is thus transformed into a quasi-matrix that provides a clear and concise way of representing all possible combinations of interaction techniques that may be employed to complete a specified task. Another useful property of this new arrangement is its ability to qualitatively illustrate the comparative efficiencies of functionally equivalent task performance strategies. Shown side-by-side in Figure 2, the merit of using the Ctrl-S key combination, relative to each of the other strategies, is immediately apparent.

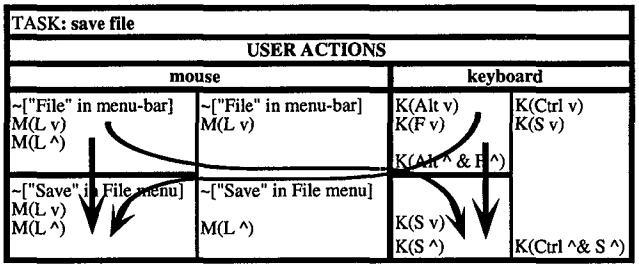

Figure 3

Application to Individual Differences Research

Our extensions to the UAN provide the individual differences researcher interested in user strategies with a new technique for analyzing the behavioral aspects of user interfaces. The modifications enable a UAN specification to show interaction strategies in a manner that is easy grasp because strategies can be seen as simple choices among interaction techniques. The specification effectively becomes a road map for the interface be-

ing studied that can be used by the researcher to make qualitative comparisons among strategies and techniques and support predictions about patterns of use.

For instance, Schmidt-Nielsen and Ackerman (1993) reported on individual differences in keyboard and mouse strategies when using the SigmaPlot™ graphing application that the task in Figures 1 and 2 is taken from. An implicit assumption in their work was that the keyboard based interaction techniques in the application were generally more efficient than the equivalent mouse actions, and therefore preferable. A complete UAN specification using our extensions would have given them a technique for comparing alternative strategies more effectively and would have enabled them to concentrate on those keyboard interactions that were truly more efficient, rather than combining all keyboard inputs together.

In addition, Schmidt-Nielsen and Ackerman could have used these extensions to explore their data, much like a box-plot is used to explore more quantitative data. Indeed, one of the first things scientists should do before undertaking an inferential statistical analysis is to examine data visually and descriptively (Tukey, 1977). The block-like representation of device-specific interaction techniques in the User Actions column is easily reduced to a schematic, as in the abstract example given in Figure 4, thus graphically laying out the essential features of the interface. This would have been a propitious vehicle for recording and analyzing individual protocols, making it possible to immediately identify distinct classes of users and unusual cases.

<table>
<tr><td colspan="4">TASK:</td></tr>
<tr><td colspan="2">mouse</td><td colspan="2">keyboard</td></tr>
<tr><td colspan="2">mouse1</td><td colspan="2">key1a | key1-2</td></tr>
<tr><td colspan="2">mouse2</td><td colspan="2">key2a | key2b</td></tr>
<tr><td>mouse3a</td><td rowspan="2">mouse3-4</td><td>key3a</td><td>key3b</td></tr>
<tr><td>mouse4a</td><td colspan="2">key4</td></tr>
<tr><td>mouse5</td><td colspan="2">key5a | key5b</td></tr>
</table>

Figure 4

Finally, these techniques could have been used in Schmidt-Nielsen and Ackerman's reporting to articulate important conceptual points. For example, since Figure 2 makes it explicitly clear that many fewer user actions are required to save a file with keystrokes than with a mouse, this figure could have been used to concisely express a critical factor in their theoretical claims.

Conclusion

In this paper, we have presented modifications to the User Action Notation that substantially enhance the notation's ability to specify new and existing user interfaces that utilize more than one input device. Specifically, we have introduced the notions of dividing the User Actions column into device-specific subcolumns and changing the unit of specificity for user actions to that of interaction techniques. Device-specific subcolumns are further subdivied as necessary to accommodate within-device task performance alternatives. The result is a more compact and powerful UAN specification that nevertheless maintains the UAN idiom of representing associated behaviors by lateral alignment. We are not aware of any other behavioral representation technique that allows the specification of concurrent input strategies to be shown as visually and clearly as the method introduced here. These modifications allow the reader of a UAN specification to make meaningful comparisons among alternative methods for completing a specified task and provide researchers with a facilitated means for empirically capturing, analyzing, and reporting individual user strategies.

Acknowledgments

We thank Astrid Schmidt-Nielsen for providing us with access to her work and her many helpful comments and advice; and Manuel Peréz and Rob Jacob for their valuable input. The authors' e-mail addresses are: brock, dievendo, and trafton@itd.nrl.navy.mil and hix@vt.edu.

UAN SYMBOLS USED IN THIS PAPER

What is Represented	UAN Symbols	Meaning
Cursor movement	~	move the cursor
Object context	[X]	the context of object X
Cursor movement	~[X]	move the cursor into the context of X
Switch operation	v	depress
Switch operation	^	release
Negation	-	negates the operation it preceedes
Location	@	specifies location
Not at	-@	not at specified location
Grouping mechanism	()	grouping, parameter, and scope
Highlight	!	highlight object
Unhighlight	-!	unhighlight object (see negation)
Display	display(X)	display object X
Erase	erase(X)	erase object X
For all	∀	for all objects...
Inequality	≠	value on left is not equal to value on right
Predicate	:	condition of viability (if left: then right)

References

Foley, J. D., van Dam, A., Feiner, S. K., & Hughes, J. F. (1990). Computer graphics: Principles and practice. Reading, MA: Addison-Wesley.

Hartson, H. R., Siochi, A. C., & Hix, D. (1990). The UAN: a user-oriented representation for direct manipulation user interfaces. ACM Transactions on Information Systems, 8(3), 181-203.

Hix, D., & Hartson H. R. (1993). Developing user interfaces: ensuring usability through product & process. New York: John Wiley & Sons.

Jacob, R. J. K. (1986). A Specification Language for Direct Manipulation User Interfaces. ACM Transactions on Graphics, 5(4), 283-317.

Schmidt-Nielsen, A., & Ackerman, P. L., (1993). Acquiring computer skills: Individual differences in style and ability. Poster presented at the annual meeting of the Human Factors and Ergonomics Society, Seattle, WA.

Tukey, J. W. (1977). Exploratory data analysis. Reading, MA: Addison-Wesley.