Human Aspects of Computing

Henry P. Ledgard Editor

The Case Against User Interface Consistency

Designers striving for user interface consistency can resemble Supreme Court justices trying to define pornography: each of us feels we know it when we see it, but people often disagree and a precise definition remains elusive. A close examination suggests that consistency is an unreliable guide and that designers would often do better to focus on users' work environments.

Jonathan Grudin

Many writers have presented the case *for* user interface consistency. Ben Shneiderman's [48] first "Golden Rule of Dialogue Design" reads: "Strive for consistency. This principle is the most frequently violated one, and yet the easiest one to repair and avoid (violating)." Rubinstein and Hersh [46] conclude their book, *The Human Factor: Designing Computer Systems for People*, with the directive, "Build consistent human interfaces." Such methodological encouragement has been buttressed with empirical work presented in support of user interface consistency. Polson [43] summarizes several experiments by stating, "Experimental results . . . show that consistency (leads) to large positive transfer effects, that is, reductions in training time ranging from 100% to 300%." Smith and Mosier [49] conclude the introduction to their compendium, *Guidelines for Designing User Interface Software*, in which over 60 guidelines contain "consistent" in the title, by stating, "the common application of design rules by all designers working on a system should result in a more consistent user interface design. And the single objective on which experts agree is design consistency."

Several steps must be taken before such advice is useful. First, consistency must be defined. Second, one must be able to identify good consistency, since foolish or undesirable consistency is possible. More subtly, one needs a procedure for discriminating among conflicting approaches to achieving consistency. Finally, a method must be found to determine when other design considerations overshadow consistency in importance.

There has been little progress in these matters. In 1981, Reisner [45] wrote, "What is (not) clear, however, is precisely what we mean by consistency and, more importantly, how to identify its absence." In 1988, a two-day workshop of 15 experts was unable to produce a definition of consistency [35]. Acknowledging the difficulty of defining the term, one prominent interface designer said, "I know consistency when I see it." One researcher has suggested that *perceived* consistency might be a better goal [26].

This article argues for a shift in perspective, suggesting that when user interface consistency becomes our primary concern, our attention is directed away from its proper focus: users and their work. Focusing on consistency may encourage the false hope that good design can be found in properties of the interface. There is an easily overlooked conflict between the search for formal properties of interfaces and the call for "user-centered" design and task analysis. The thesis that interface consistency is an important goal supports the search for interface properties that can be measured and manipulated in isolation and suggests that identifying these properties may lead to successful design.

Such a position holds out the attractive possibility of automating the design and evaluation of some aspects of interfaces [e.g., 10, 28, 41, 42, 53]. This article presents the antithetical view: interface consistency is a largely unworkable concept; the more closely one looks, the less substance one finds. In the examples that follow, the dimensions on which one might argue for consistency shift or disappear when examined closely.

Rejecting consistency as a primary user interface goal does not argue for randomness in user interface design. The studies that appear to support particular applications of consistency [e.g., 5, 16, 43] are not wrong, but their proper interpretation may be more contextually bound, less general than is explicitly recognized. Shifting the focus from general interface properties to the users' tasks and work context, to physical constraints, and to psychology does not mean defocusing. Attacking the maxim "strive for consistency" may seem a step backward, but it may be a necessary first step in finding a better path forward. By placing the case for interface consistency in its proper context—alongside the case against interface consistency—we can develop a new synthesis.

An Illustrative Example from Everyday Life

Consider the household interface design problem of deciding where to shelve implements, specifically knives: butter knives, table knives, steak knives. All may be kept in the same drawer. Consistent, easy to learn, easy

to remember: there is one place to go for knives. This arrangement makes it easy for guests to be more help than hindrance when they assist in cleaning up.

But it isn't quite so simple. To begin with, there are the carving knives, too large for the drawer, stuck in a wood block. At least it is close by, in plain view; a good strategy for handling exceptions, perhaps. But then, there is the *real* silverware packed away in a drawer in a cabinet holding crystal, china, and other finery for special occasions. The silver knives are not in the kitchen, are not visible, but are not too far distant. But what about the putty knife, stowed in a workbench drawer in the garage, or the Swiss army knife, packed away with the camping gear in the basement?

By distributing the knives, we have introduced inconsistency and increased the time needed to learn where to find them. Why is it the best solution? We have made the knives easier to use by placing them according to how they are used, according to the tasks in which they are involved. User tasks or activity patterns totally dominate such design decisions. A formal examination of the properties of household objects would reveal that the knives all have very much in common, yet we distribute them in such a way that they are grouped more closely with objects such as forks, tent pegs, and a jar of putty than with other knives. The organizing element is the context of use. A shift in that context, in work organization, may lead to a design change. I recently lost my corkscrew, leaving me to rely on the corkscrew in the Swiss army knife, which I moved to the kitchen (though to a drawer of assorted implements, not the silverware drawer).

Interface objects, too, must be designed and placed in accordance with users' tasks. Once this is fully appreciated, the concept of "interface consistency" becomes surprisingly insignificant in many design situations. But designers who have only a sketchy or partial understanding of users' tasks will find it difficult to appreciate the dominant role tasks should play in interface design. In the absence of task analysis, the designer has little to go on and it becomes convenient to focus on properties of the interface—which in the absence of something better is likely to help more than hurt. But it is a mistake to believe that this is the *appropriate* focus.

Several interface features are presented in the following section that improve upon alternative "consistent" designs. Most of the design choices cannot easily be described in terms of consistency but can be understood through careful analysis of the users' work.

Three Types of "Consistency"

User interface consistency is used in three interrelated senses: the internal consistency of a design with itself; the external consistency of a design with other interface designs familiar to a user; and an external analogic or metaphoric correspondence of a design to features in the world beyond the computer domain. Although none of these is a reliable guide to user interface design, each has somewhat different properties. Each design example covered in subsequent sections will apply more to one or another use of "consistency."

Internal consistency of an interface design. The designer or design team may not know the context in which people will use their product, but they usually control a range of features affecting the dialogue of users with their particular application or system. Thus, internal design consistency receives the most conscious attention of designers. Consistency might be sought in physical and graphic layout, command naming and use, selection techniques, dialogue forms, etc. For each domain, different dimensions of consistency are possible. The graphic properties of a set of interface objects may be consistent in color, size, shape, or less easily defined style. Carroll [5] notes more complex dimensions in the design of command namesets: "Move backward" would be consistent with "Move forward," but "Go backward," "Backward," or "Reverse" would be inconsistent (in Carroll's terms "incongruent," "hierarchically inconsistent," and both, respectively). Green and Payne [16] regularized EMACS commands along a single set of organizing principles and produced dramatically better performance in a learning task. Other internally consistent designs would result from using one abbreviation algorithm, one approach to setting menu defaults, one functional mapping of mouse button assignments, and so forth. Initial learning in particular, but also ease of use and perceived quality are reported to benefit from internal consistency [5].

External consistency of interface features with features of other interfaces familiar to the users. A design can be internally consistent, yet conflict with other interfaces. This can happen for any of several reasons: the other interface may itself be internally inconsistent, the other interface may be internally consistent but incorporate different choices (e.g., different names for objects), or the other interface may only cover some of the same functionality and thus differs in the distinctions that it needs to make. External consistency may be achieved at the expense of internal consistency, as when a designer borrows a highly salient feature from a familiar system for use in an otherwise different interface. Alternatively, internal consistency may be achieved at the expense of external consistency; such "local optimization" of the interface often occurs when an application designer has limited knowledge of what the end-user is familiar with and what application set the product being designed will appear with. When an application will appear without modification in different environments, full external consistency may not generally be possible. "Transfer of training" is then a key concern.

Correspondence of interface features to familiar features of the world beyond computing. A user interface design may use metaphor or analogy to objects, attributes, or relations in the world outside of computer interfaces. With analogic consistency, there is some common structure, but the elements may have quite different

meanings; for example, the arrangement of arrows on cursor control keys may match the arrangement of compass point arrows on a map. This is quite different from the more precise external consistency of the cursor key arrangement on one keyboard with the cursor key arrangement on a different keyboard, where the elements have the same meaning.[1] Analogies are weaker correspondences, but because the real-world domains are part of everyone's experience, they may be significant aids to initial learning and recall—as well as possible pitfalls for the designer. Carroll [6] points out that if technology is to provide an advantage, the correspondence to the real world *must* break down at some point.

These distinctions among types of consistency may be less salient to users than to the designer. At a given moment, a user may be dealing with various applications and systems (as well as the world). Where the designers see two applications with internally consistent but externally inconsistent interfaces, a user may see one internally inconsistent system. However, the distinction may be important to the interface designer, who may be able to influence the internal consistency of the application being worked on while having no knowledge of or control over the applications it will be used with. And there can be performance consequences—the cost of one-time retraining on a new interface can be high, but if each interface is internally consistent the disturbance may be transitory, whereas users may never adjust to an internally inconsistent interface (or when two incompatible interfaces are used intermittently).

Ease of Learning can Conflict with Ease of Use

Much of the empirical work on consistency has focused on learning, whether initial learning or transfer of training to a new situation. Learning is prerequisite to usage, of course, and in the case of infrequent or casual use, a user may always be in a state best described as learning. It seems reasonable that a consistent interface will be easy to learn. An internally consistent design can often be characterized with a small number of names and relations. An externally consistent interface may allow the use of previously learned associations or be invoked by reference to a guiding analogy that will also aid in subsequent recall.

However, ease of learning can conflict with subsequent ease of use. When this happens, priorities must be established carefully. If learning isn't possible, use will not happen. However, people buy systems and applications not to learn them, but to use them. We generally build user interfaces, not learner interfaces. If a consistent interface supports learning but impedes skilled performance, and if its major use is by skilled users, then consistency is working against good design.

[1] Gentner [12] uses the terms "analogy" and "literal similarity" to describe this distinction.

The keyboard examples that follow are of particular interest because they have been extensively studied and because their designs have evolved over time. Subsequent examples will demonstrate that similar design considerations apply to software interfaces.

Design exercise #1: Function key layout

Figure 1a presents a design problem confronted by the International Organization for Standardization as this is written: Given the six possible positions illustrated, where should the four arithmetic operations be placed on a numeric keypad that is to support data entry and calculator functions?

It was proposed that the assignment be across the top row, in the sequence +, −, ×, ÷ [24]. The rationale was that this is *consistent* with the order in which we learn and remember these operations. When I gave 24 interface professionals this task, a plurality (8) produced this solution.

The proposed design is easy to learn and recall, but it is difficult to use. For data entry and calculator use, the keys struck most frequently are Enter, +, and 0. Thus, placing the + key in the upper left corner maximizes hand movement. Placing it directly above the Enter key minimizes finger movement and error. This is in fact a common arrangement on commercial keyboards.

Thus, the best design violates consistency—it is not consistent with the order in which we learn and spontaneously recall the operations, and it seems aesthetically inferior to spread the operations across multiple rows and columns as well. Developing the appropriate design required analysis of the users' task and an understanding of human motor control.

Figure 1b shows three cursor key configurations used to move the user's screen position left, right, up, and down. Setting aside factors introduced by the size and shape of the keyboard and the locations of other keys, which is the best design?

The set of 24 interface experts unanimously chose the star (*ii*) as the design that they thought novices would rate most highly and about half of them also chose the star as the best design. The star has an obvious aesthetic appeal and it is consistent by analogy with directional indicators that appear on compasses, maps, and so forth. But studies have shown that the inverted T (*iii*) is the most usable configuration [29] and its use is spreading through the industry. Users require access to opposed pairs of directional keys (e.g., to compensate for overshoot), and with (*ii*), with the index finger on the cursor left key and the ring finger on cursor right, the middle finger can cover both the cursor up and cursor down keys most efficiently. This performance advantage may have first been discovered by the designers of fast-action computer games, who often map the directional movements to the i, j, k, and l keys, which form the inverted T pattern.

The appropriate design is based on an understanding of the cursor manipulation task and of the user—in this case, the physical characteristics of the hand. In both function key examples, designs consistent with exter-

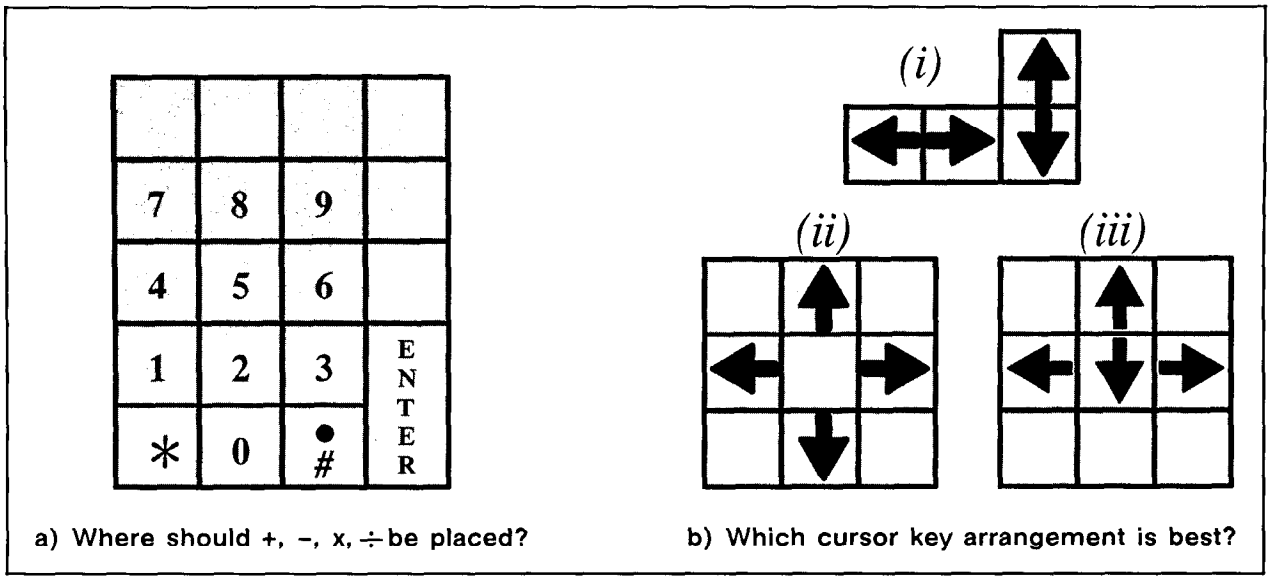

FIGURE 1. Function Key Design Decisions

nal experience were tried, but ultimately were found to work against performance efficiency.[2]

Design exercise #2: The typewriter keyboard

Consider the typewriter keyboard layout as a pure design problem, setting aside the retraining problems that a new design would entail. The earliest keyboards were alphabetic [3] and with the advent of calculators and computers alphabetic keyboards have again been proposed (see [36]). The rationale for alphabetic keyboards is a consistency argument—we are familiar with the alphabet, so key assignments consistent with alphabetic letter order are easier to learn and perhaps recall (but see [38]). Nicolson and Gardner [34] show that in some situations complete novices do better with an alphabetic keyboard.

It is not necessary to review the history of keyboard optimization efforts to discern in Figure 2 the key point: Apart from alphabetic keyboards, consistency has played no role. Typing performance is governed by several factors, including the balance of workload across hands and fingers, the average length of finger trajectories, and whether letter pairs frequently typed in succession are typed by the same finger, by two fingers of the same hand, or by fingers on different hands (the latter is fastest because we can carry out such motor actions in parallel). These factors are functions of the frequency of individual letters in the language being typed and the distance between keys often typed in succession by the same finger or hand, which in turn is a function of the two- and three-letter combinations (digraphs and trigraphs) in the language. Designers have given slightly more work to the stronger right hand and to the stronger forefingers and middle fingers, and concentrated typing activity on the middle row of the keyboard to minimize vertical finger and hand movements. Keyboard designers have also proposed "split" keyboards better adapted to our hands, arms, and shoulders [32, 39].

Thus, keyboard design requires knowledge of human physiology and motor control, as well as properties of the language to be typed. The evolution of designs has reflected the changing view of which factors are germane and how they should be balanced. It is a complex constraint satisfaction process—a slow, empirical process that may never be demonstrably complete. The resulting layouts are difficult to memorize; they are not consistent with experience and letters are not grouped according to simple rules. Whether an alphabetic arrangement *is* optimal for children learning to use a microcomputer [34] or leads to problems [38], even raising the issue emphasizes that the design process must center on a detailed understanding of the users' experience.[3]

Summary: Consistency that supports ease of learning can conflict with ease of use

In the three keyboard examples, designs consistent with user experience in other domains were marketed and provided easy learning and recall. Performance considerations then led to user-centered, task-centered approaches and new designs, with no role for the concept of consistency. Thus, the empirical studies of the benefits of consistency, which have largely measured ease of learning, recall, and transfer to new situations, must be treated cautiously. Learning and recall can be

[2] Of course, a narrow measure of performance efficiency for heavy users is just one design consideration. Aesthetic considerations may have a marketing role and may lead people to feel better about their work, perhaps boosting productivity. Function key placement may also be influenced by keyboard space constraints. To know how heavily to weight performance efficiency, one needs to know how much time users will spend with a feature. In short, only a higher-level analysis can determine the cost of overriding a design decision that was based on a task analysis focused strictly on performance efficiency.

[3] For full accounts of physical and linguistic influences on typing performance, see [13, 14]; for a detailed and entertaining review of the history of keyboard optimization efforts, see [39].

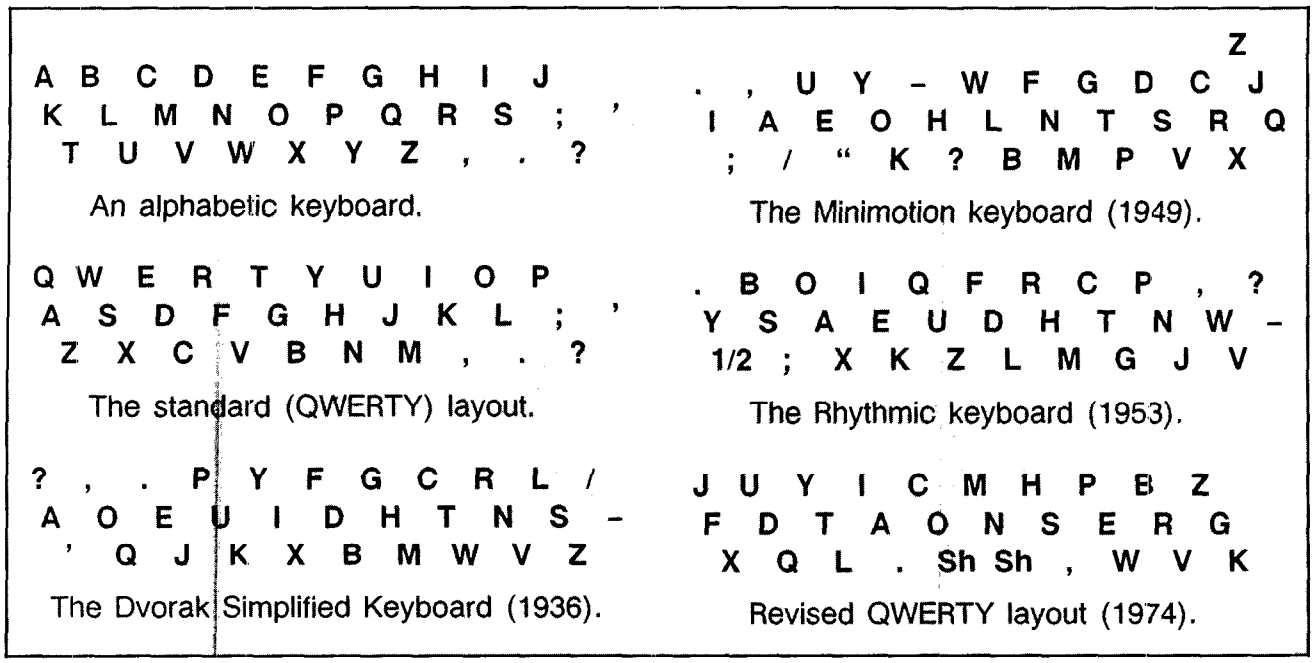

FIGURE 2. Attempts to Optimize the Typewriter Keyboard (after Noyes [39])

important, but using an interface to carry out a task can be very different from remembering what the interface is. Furthermore, as the next section illustrates, interface consistency sometimes works against both novice and skilled users.

Consistency can Work Against Both Learning and Use

Design exercise #3: Default menu selections
With many designs, when a menu appears, one item is highlighted, the default selection. Which item should it be? Commercially available systems have experimented with a number of approaches: the first item on the menu, the item that users most frequently access, or the item that this particular user most recently accessed.

The examples shown in Figure 3 and described below demonstrate that any such internally consistent design is sub-optimal. Menus are employed for a variety of tasks and optimal menu defaults vary with those tasks.

Consider the following scenario: A user wishes to italicize certain words in a document. After the first such word is found and selected, the edit menu is accessed—in the system illustrated, by using a mouse button to pop up the menu. It appears with the default selection of "Props" (for properties). The user overrides the default to select Fonts, then Italics from the second-level menu, and On from the third (as shown in Figure 3, 1a). When the user then finds the next word to be italicized and brings up the menu, it automatically comes up as shown in 1b)—with Fonts, Italics, and On already defaulted. This is the "last item accessed" rule.

Now the user wants to copy a paragraph using the Copy and Paste operations. Copy will place a copy of the text in the clipboard and Paste will then move it from the clipboard to the cursor location. The user brings up the menu of commands and overrides the default to select and execute Copy (Figure 3, 2a). Then the user moves the cursor to the destination and brings up the menu again. This time it automatically defaults to Paste! (Figure 3, 2b.) This violates the "last item accessed" rule but is what the user virtually always wants. After pasting, the user again brings up the menu, which now defaults to Copy (Figure 3, 2c). This toggling allows a user to make multiple copies rapidly—clicking the button six times produces three copies, without requiring close attention.

The rule "consistently default to the item the user is most likely to select *next*" would cover the cases examined so far. However, the next example illustrates that even this is not optimal. Now the user wishes to right-justify the text. The user accesses the Properties menu, selects Flush right, and brings up the menu to apply the change. It comes up defaulted to Apply, which would apply the change only to the current paragraph. The user wishes to right-justify the whole document, so overrides the default and selects Global apply. This can be an irreversible operation, so a confirmation step is required: a second-level menu with Confirm and Cancel choices appears, defaulted to Cancel. This forces the user to take the step of confirming that this action is desired (Figure 3, 3a). If the user then wishes to make another global change, the entire sequence must be stepped through again (Figure 3, 3b). Apply and Cancel will still be defaulted, even though Global apply and Confirm were chosen the previous time and even though Confirm is what the user will select the overwhelming majority of times this menu is accessed. The

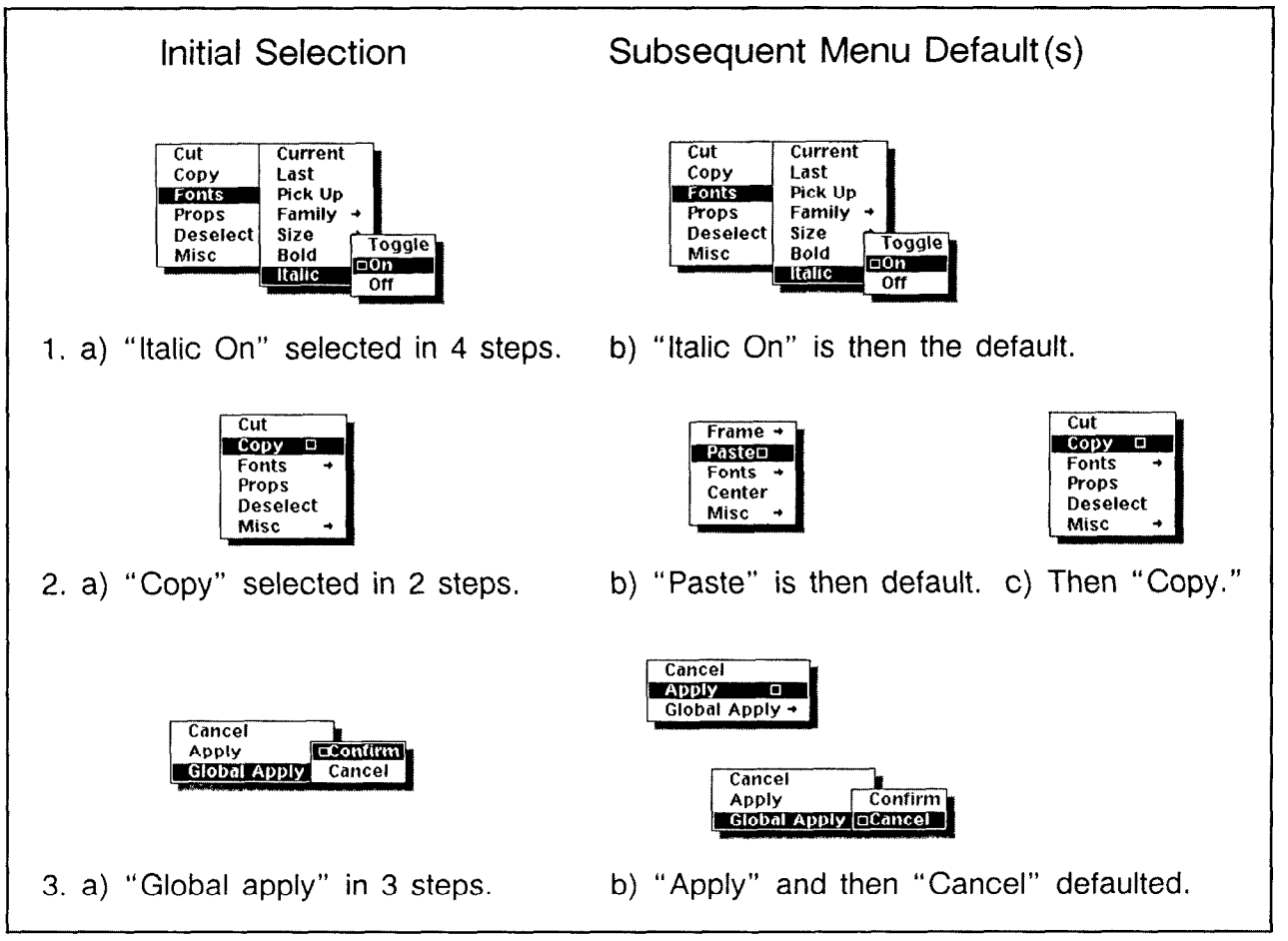

FIGURE 3. Menu Default Patterns Vary with Aspects of the Task

point of a confirmation step is to force the user to do it every time. Thus, the rule of "default to the item the user is most likely to select *next*" is violated when the most likely choice is a potentially irreversible operation.[4]

No rule, consistently applied, produces good menu defaults. Enforcing a blanket consistency will damage the interface. In fact, subtle shifts in the situation can change the optimal interface behavior. If the application were changed by introducing an Undo operation, so Global apply was reversible, there would be no need to default repeatedly to the less dangerous Apply operation. Then, after one Global apply was executed, the menu default could be Global apply, the most likely next choice.

Menu defaulting is typical of where one might hope to enforce interface consistency through examination of formal properties of the interface. But, as with knife placement, the "inconsistent" interface choices described earlier are so naturally molded to the users' tasks that users may not even notice the inconsistency.[5] No tool to produce or check for interface consistency could handle this interface design appropriately unless it contained a *detailed* knowledge of the users and their tasks, as well as high-level knowledge of the correspondence between those tasks and the functions applicable to specific interface objects (e.g., the difficulty of undoing particular operations).

Establishing Appropriate Dimensions for Consistency

Design exercise #4: Abbreviating command or operation names

Assume we wish to construct abbreviations for 20 command names. What abbreviation strategy would be best? Truncation? Vowel deletion? Single-letter abbreviations?

You may have anticipated the answer: It depends on how the abbreviations will be used. Truncation is the best abbreviation strategy if users will be typing familiar commands in a typical command-driven interface [50]. In this situation, the users know the command they want to enter and must reproduce the abbrevia-

[4] One of many scenarios in which "Global Apply" is irreversible: Some text is in bold, some is in italics, and one globally changes it all to be standard font. The system now retains no memory of what was previously bold, what was italicized, so the only way to recover the initial state is to try to recall it word by word. The system has no general "undo" capability.

[5] Conversations with users suggest that the inconsistencies may be noticed initially, then quickly forgotten.

tion. Truncation is a simple rule to recall and apply, generally leading to short abbreviations. However, if the abbreviations are being created to put on key caps, for example, then vowel deletion may be more appropriate.[6] In this case, the users will see the abbreviation and use it to reconstruct the full name of the operation, and longer abbreviations may be OK. Finally, if the command is one that users will type thousands of times or if a menu will be visible to remind the users of the options, as in many electronic mail programs, then a single-letter abbreviation strategy may be best. When commands are overlearned or otherwise cued, minimizing the number of keystrokes may be the highest priority. (The role of task and experience in abbreviation are further discussed in [1, 19].)

This picture is more complicated when an abbreviation serves multiple purposes. For example, a user might try to recall an abbreviated file name in order to type it or see the same abbreviated name in an index and try to recall what it stands for. A command might be overlearned for some users but not for other users. Allowing users to customize the interface may be possible, though it can be very risky [20]. The availability of customization does not eliminate the need for good design defaults.

A Snare: A Form of Consistency that is Always Available but Often Inappropriate

Design exercise #5: Printing a folder or directory
Consider the situation depicted in Figure 4. A folder containing several documents has been selected. A menu of operations has been accessed and the Print operation selected. The design question is: What should be printed when the operation is executed?

In informal studies, most people said that they would expect all of the documents in the folder to be printed. However, system architects have argued that issuing the Print command in this situation should cause a list of the documents within the folder to be printed. In one project, the implementers argued successfully for this design *on the basis of consistency with the internal system architecture.* A folder in this system was a list of pointers to documents. Printing a document produced a hardcopy of the contents of the document, and it was argued that printing a folder should similarly produce a hardcopy of its contents, namely a list of the documents contained in the folder. The developers argued that consistency was of paramount importance, in this case consistency based on the underlying software architecture.[7]

Clearly, the wrong dimension for consistency was chosen. But it is not clear how to describe the optimal design as being "consistent"—printing documents,

[6] "Vowel deletion" has various definitions, and rarely is as simple as literally deleting every vowel. Initial vowels are almost always left in place. Sometimes, final vowels are also left intact—only internal vowels are deleted. Streeter et al. [50] deleted all internal vowels following the first syllable.

[7] In the case described, a human factors engineer proposed that an index appear, allowing a user to select among possible options. This design was rejected in favor of consistency with the system architecture. (The system illustrated in Figure 4 actually prints all documents in the folder.)

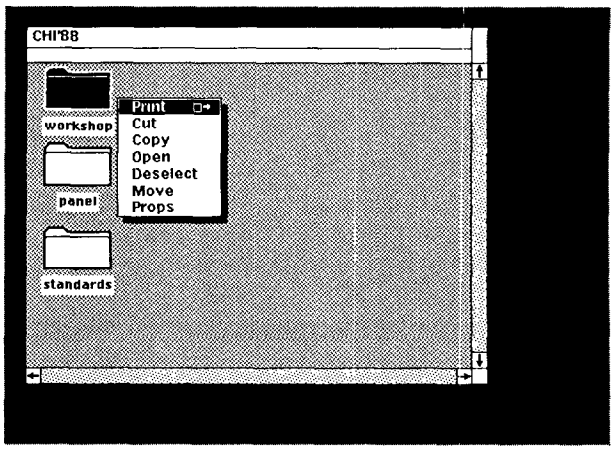

FIGURE 4. What Should Happen When 'Print' Folder (directory) is Executed?

whether document or folder is selected. Of course, it is "consistent with what users want," since most users wish to print documents more often than lists of documents. Once again we are focused on understanding the users' tasks.

Summary: The most available consistency is often harmful
This example is particularly important because the mapping of the system architecture onto the user interface is a common source of interface design flaw (see e.g., [2, 17]). Other examples of mapping the system design onto the user interface design include grouping menu items based on implementation considerations, assigning functions to function keys on similar grounds, and even consistently displaying information in screen regions based on data type. Recommending consistency as a design guide leaves designers free to look for consistency where they will, and they are often much more familiar with the system architecture side of the interface than they are with the "human architecture"—that is, users' tasks and psychology.

The advocate for the user must realize that the job often requires arguing *for inconsistency*—inconsistency of the user interface with the underlying software architecture. If on other occasions the advocate endorses consistency as a general rule, the engineer will understandably be confused or frustrated. A quest for a formally expressible consistency within the interface may almost inevitably take one in the wrong direction—toward the underlying system architecture, which we know how to express formally, and away from the human psychological and task architecture, which are not understood nearly as well.

DISCUSSION

The abbreviation and menu defaulting examples demonstrate that internal design consistency is an unreliable measure of good interface design. Choosing appropriate abbreviations requires knowledge of their ultimate use, and may require a careful tradeoff analysis. Choosing the appropriate menu default

requires detailed knowledge of the higher-level tasks represented and also depends on system characteristics such as the availability of an "undo" operation. Small changes in the environment may challenge an existing, consistently-applied principle. For example, the Macintosh conventions of pull-down menus at the top of the display and a strict correlation of mouse pointer movement with mouse movement are particularly suited to single applications and the original small display screen. With larger screens and multi-tasking environments, these conventions may force users to make frequent, large movements—Hypercard tear-off menus are a sign that skilled user performance issues will break down this consistent design feature.

The keyboard examples demonstrate the unreliability of designing by analogy to real-world structures. In each case designers first adopted consistency with a natural external domain but abandoned these designs when the users' tasks were better understood. Analogy to a familiar domain facilitated immediate learning but had negative consequences for subsequent performance. Initial learning, the focus of most laboratory research and usability testing, is an important and in some cases the primary issue. But skilled performance may be of greater concern; also, as we move to more interactive systems, certain aspects of learning become less relevant; in the case of menu defaults, novices as well as experts may operate smoothly and efficiently without consciously learning the design criteria.

A special case of designing by analogy is that of designing the user interface to correspond to the underlying system architecture, as in the print folder example. The system architecture is external to the user interface and will not be familiar to many users, but is typically very familiar to the designers. Although it may work against the user, mapping the system architecture onto the user interface is very seductive, appealing to the designer's sense of consistency and simplicity. Some have even argued that one should, in fact, teach the user veridical models or how it works [26, 27, 44]. But as our designs adapt to the ways that computer communities actually use systems, it is more likely that the user interface and system design will diverge, *reducing* the appropriateness of burdening the user with both descriptions.

The external consistency of a given interface with those of other computer applications and systems is an increasingly important issue as computer use spreads through the population and into different application areas. Interoperability and backwards-compatibility requirements reflect aspects of the users' experience and environment that should be reflected in an interface design, but designers who lack the necessary information about the installed customer base or the marketing strategy will optimize interface designs locally for their own product at the expense of a more global efficiency [18, 21]. These complexities and the resulting tradeoffs present further difficulties for formal approaches to generating or monitoring for consistency.

TOWARD A SYNTHESIS

The first step toward integrating the case against user interface consistency with the intuitive feeling and experimental evidence that some kinds of consistency are beneficial and that inconsistency can be harmful is to recognize that a "fully consistent system" [25] is not achievable. Interface design is an engineering problem that forces tradeoffs among many factors, often including many possible forms and dimensions of consistency. Next, we need to recognize the primacy of the users' work contexts in dealing with these tradeoffs, cognizant that in some circumstances, the highest priority may not be characterized as consistency at all. The importance of centering the interface design on users and their tasks is acknowledged [e.g., 15, 37], but often the focus is on individual users carrying out simple, cognitive tasks. Hutchins, Hollan and Norman [23] go quite far in shifting the focus by stressing the importance of the "task domain" and of making the "mechanisms of the system match the thoughts and goals of the user"; others have gone further into the work context to obtain an understanding of the physical, social, and even historical structures in which users' tasks are carried out [e.g., 4, 9, 51, 52]. This article describes a pattern in the history of interface design that this focus explains—the evolution away from consistent designs in areas where user tasks are understood.

In a concise review of the work on consistency, Kellogg [25] developed Moran's [30] taxonomy of user interface levels into a framework for consistency. Of six interface levels on which consistency might be sought—device, spatial layout, lexical, syntactic, semantic, and task—the work reviewed dealt with the middle four. Payne and Green [40] used a similar taxonomy in outlining a formal model to represent "task languages." Both Kellogg's "task level" and Payne and Green's "task language" are defined in terms of simple operations or system functions, such as entering a number. "Task" is thus very narrowly defined; Green and Payne call them "simple-tasks," and Kellogg describes her focus as "the structure of tasks defined by the system (e.g., whether similar tasks are decomposed into analogous subtasks)" and "the mapping of the user's understanding of the task domain to system procedures (e.g., whether tasks with analogous semantic procedures on the system are in fact perceived as similar tasks by the users)."

In this article, the term "tasks" encompasses users' actual work in a much broader sense: the different tasks of data entry and numerical calculation may employ the same function of entering a number; further, data entry done frequently is a different task than data entry carried out infrequently. Since computer applications and interfaces rarely represent the physical, temporal, or social contexts of users' work, Kellogg's framework did not have to address these factors. But these contexts do, or should, implicitly influence our design choices at lower levels, whether or not they are *explicitly* recognized in the design.

Given this hierarchy of user interface elements and the recognition that highest level (task) considerations can warrant overriding consistency at lower levels, one might propose that concerns at any one level override those of lower levels. For example, we saw that semantic aspects of users' tasks could justify overriding a syntactic approach to menu defaulting. Along these lines, Kellogg [25] suggested that a possible strategy is "to optimize low-level interactions for each environment, but to hold the underlying conceptual model of the system constant."

However, this conflicts with intuitive and empirical evidence that inconsistency in low-level interactions can be very frustrating—keyboards that differ in the position of a few keys (e.g., escape and control keys), programs that differ in the names of important commands (e.g., "quit" and "exit"), and so forth. Performance decrements in keypad use have been measured when the conceptual model was held constant and the low-level interactions were allowed to vary [8]. More generally, transfer of training experiments show that the learning or unlearning of low-level productions can be central [43]. In an experiment that varied both physical surface similarity and the absence or presence of an underlying conceptual model of an interface, Schumacher and Gentner [47] found that both surface similarity and the conceptual model had significant effects on transfer of training.

Thus, there may be no simple approach to determining the relative significance of consistency along various dimensions and levels. We know that detailed knowledge of the users' task context can identify or eliminate some dimensions on which a design should be consistent. Perhaps the significance of consistency on a dimension or level is related to the frequency with which users operate on the level or vary their activity along the dimension. In addition, we are helped by thinking of consistency as one goal, often in conflict with other goals that are at times more important. Where task knowledge is unavailable or does not constrain the design, some form of consistency may be the best resort, suggesting a role for formal approaches to generating or evaluating consistency [e.g., 10, 28, 41, 42, 53], but formal systems must provide enough flexibility to permit the dimensions of consistency to be changed or overridden as task knowledge increases, and the temptation to map the system design onto the user interface design must be resisted. The studies that identify dimensions on which unmotivated inconsistency can be costly lend support to design tools that bring such inconsistency to the attention of the designer [e.g., 22, 31, 33] and to design tools that provide for the representation of arguments for particular design decisions, including conflicting arguments [e.g., 7, 11].

When a user interface is designed for a system built for a wide range of applications and users, such as the Macintosh interface, it is more difficult to get a meaningful understanding of the users' tasks. In this situation, it may make sense to adhere consistently to some interface choices to provide users with the benefits of easier learning and transfer. But this analysis suggests that as these systems are pushed into specific application areas with dedicated users, maintaining that consistency will lead to increasing performance costs. Only careful attention to the users and their tasks will indicate where those costs outweigh the benefits.

The examples in this article were drawn from a limited number of domains—keyboards, menus, command names, file structures—but this may primarily reflect our greater familiarity with these domains. In these areas, designs often went through a sub-optimal consistent phase before our understanding of the relevant users' tasks grew and the consistent design was abandoned in favor of a better one. Revisiting the example from the introduction, if a Martian were arranging our household implements, with no idea of the tasks for which each would be used, placing all of the knives together would be a good idea. Once the Martian came to understand how we used different knives, it could come up with a better arrangement. Similarly, as we come to understand users' work environments better, we should find designs evolving away from consistent interfaces in many areas of interface design. The way to stay at the forefront of this evolution is to know your users and their tasks a little better than anyone else.

Acknowledgments: Many colleagues at the MCC Human Interface Laboratory, and earlier at Wang Laboratories and the MRC Applied Psychology Unit, have puzzled over these issues with me. In particular, I am indebted to Phil Barnard, Susan F. Ehrlich, and Steven Poltrock, as well as to discussions with Dedre Gentner, Don Gentner, Catherine Marshall, and Don Norman. Kenichi Akagi provided information on aspects of keyboard layout, and with Will Hill, Jim Hollan, Jim Miller, Tom Erickson, Larry Parsons, and Jack Carroll, commented usefully on an earlier draft. Editorial guidance by Henry Ledgard and two anonymous reviewers led to marked improvements. This paper would not have been written without the impetus provided by Jon Meads, Jakob Nielsen, and the participants in the Workshop on Coordinating User Interfaces for Consistency at CHI'88; discussions with Wendy Kellogg were particularly helpful.

REFERENCES

1. Barnard, P. and Grudin, J. Command names. In *Handbook of Human–Computer Interaction*, M. Helander, Ed. Elsevier Science Publishers, Amsterdam, 1988.
2. Barnard, P.J., Hammond, N.V., Morton, J., Long, J.B., and Clark, I.A. Consistency and compatibility in human–computer dialogue. *Int. J. Man–Machine Studies 15*, (1981), 87–134.
3. Beeching, W.A. *Century of the typewriter*. St. Martin's Press, New York, 1974.
4. Bodker, S., Ehn, P., Knudsen, J., Kyng, M., Madsen, K. Computer support for cooperative design. In *Proceedings of the CSCW'88 Conference on Computer-Supported Cooperative Work* (Portland, September 26–28, 1988).
5. Carroll, J.M. Learning, using, and designing filenames and command paradigms. *Behav. Info. Tech. 1*, (1982), 327–346.
6. Carroll, J.M., Mack, R.L. and Kellogg, W.A. Interface metaphors and user interface design. In *Handbook of human–computer interaction*. M. Helander, Ed. Elsevier Science Publishers, Amsterdam, 1988.

7. Conklin, J. Design rationale and maintainability. In *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences* (January, 1989).
8. Conrad, R., and Hull, A. The preferred layout for numeral data-entry keysets. *Ergonomics 11*, (1968), 165–174.
9. Ehn, P., and Kyng, M. The collective resource approach to systems design. In *Computers and democracy*—A Scandinavian challenge. G. Bjerknes, et al., Eds. Aldershot, UK.
10. Feiner, S., 1988. An architecture for knowledge-based graphical interfaces. In *Proceedings of the AAAI/Lockheed Workshop on Intelligent Interfaces* (March, 1988).
11. Fischer, G. and Morch, A. CRACK: A critiquing approach to cooperative kitchen design. In *Proceedings of the International Conference on Intelligent Tutoring Systems* (Montreal, June, 1988).
12. Gentner, D. Structure-mapping: A theoretical framework for analogy. *Cognit. Sci. 7*, (1983), 155–170.
13. Gentner, D.R. Keystroke timing in transcription typing. In *Cognitive aspects of skilled typewriting*. Springer-Verlag, New York, 1983.
14. Gentner, D.R., Larochelle, S., and Grudin, J. Lexical, sublexical and peripheral effects in skilled typewriting. *Cognit. Psy. 20*, (1988), 524–548.
15. Gould, J., and Lewis, C. Designing for usability—Key principles and what designers think. *Commun. ACM 28*, (1985), 300–311.
16. Green, T.R.G., and Payne, S.J. Organization and learnability in computer languages. *Int. J. Man-Machine Studies 21*, (1984), 7–18.
17. Grudin, J. Designing in the dark: Logics that compete with the user. In *Proceedings of the CHI'86 Human Factors in Computing Systems* (Boston, April 13–17, 1986).
18. Grudin, J. Organizational influences on interface design. In *Mental models and user-centered design*. A.A. Turner, Ed. Workshop Report. TR 88-9. University of Colorado, Institute of Cognitive Science.
19. Grudin, J., and Barnard, P. The role of prior task experience in command name abbreviation. In *Proceedings of the INTERACT'84 Conference on Human-Computer Interaction* (London, September 4–7, 1984).
20. Grudin, J., and Barnard, P. When does an abbreviation become a word? and related questions. In *Proceedings of the CHI'85 Human Factors in Computing Systems* (San Francisco, April 14–18, 1985).
21. Grudin, J., and Poltrock, S. User interface design in large corporations: Coordination and communication across disciplines. In *Proceedings of the CHI'89 Human Factors in Computing Systems* (Austin, April 30–May 4, 1989).
22. Hollan, J.D., Hutchins, E.L., McCandless, T.P., et al. Graphic interfaces for simulation. In *Advances in man-machine systems research, 3*, W.B. Rouse, Ed. JAI Press, Greenwich, Conn., 129–163.
23. Hutchins, E.L., Hollan, J.D., and Norman, D.A. (1985), Direct manipulation interfaces. *Human-Computer Interaction 1*, (1985), 311–338.
24. International Organization for Standardization. Information processing—keyboard layouts for text and office systems—part 41: Function zones of the numeric section. Draft Proposal ISO/DP 9995-41, 1988.
25. Kellogg, W.A. Conceptual consistency in the user interface: Effects on user performance. In *Proceedings of INTERACT'87 Conference on Human-Computer Interaction*, (Stuttgart, September 1–4, 1987).
26. Kellogg, W.A. Coordinating user interfaces for consistency (unpublished position paper for CHI'88 workshop of same name).
27. Kieras, D. Mental models for engineered systems and computers. In *Mental models and user-centered design*. A.A. Turner, Ed. Workshop Report. TR 88-9. University of Colorado, Institute of Cognitive Science.
28. Mackinlay, J. Automating the design of graphical presentations of relational information. *ACM Trans. Graph. 5*, (1986), 110–141.
29. Manuel, T. Molding computer terminals to human needs. *Electronics* (June 30, 1982), 97–109.
30. Moran, T.P. The command language grammar: A representation of the user interface of interactive computer systems. *Int. J. Man-Machine Studies 15*, 3–50.
31. Myers, B. Creating dynamic interaction techniques by demonstration. In *Proceedings of the CHI+GI'87 Human Factors in Computing Systems* (Toronto, April 5–9, 1987).
32. Nakaseko, M., Grandjean, E., Hunting, W., and Gierer, R. Studies on ergonomically designed alphanumeric keyboards. *Human Factors 27*, (1985), 175–187.
33. Neches, R. Knowledge-based tools to promote shared goals and terminology between interface designers. *ACM Trans. Off. Info. Syst. 6*, 215–231.
34. Nicolson, R.I., and Gardner, P.H. The QWERTY keyboard hampers schoolchildren. *British J. Psy. 76*, (1985), 525–531.
35. Nielsen, J. Coordinating user interfaces for consistency. *SIGCHI Bulletin 20*, (1989), 63–65.
36. Norman, D.A. *The psychology of everyday things*. Basic Books, New York, 1988.
37. Norman, D.A., and Draper, S.W., Eds. *User centered system design*. Hillsdale, Lawrence Erlbaum Associates, Hillsdale, N.J., 1986.
38. Norman, D.A., and Fisher, D. Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors 24*, (1982), 509–515.
39. Noyes, J. The QWERTY keyboard: A review. *Int. J. Man-Machine Studies 18*, 265–281.
40. Payne, S.J., and Green, T.R.G. Task-action grammars: A model of the mental representation of task languages. *Human-Computer Interaction 2* 93–133.
41. Perlman, G. Multilingual programming: Coordinating programs, user interfaces, on-line help, and documentation. *ACM SIGDOC Asterisk*, (1986), 123–129.
42. Perlman, G. An axiomatic model of information presentation. In *Proceedings of the 1987 Human Factors Society Meeting*. Human Factors Society, New York, pp. 1229–1233.
43. Polson, P. The consequences of consistent and inconsistent user interfaces. In *Cognitive science and its applications for human-computer interaction*. Lawrence Erlbaum. Hillsdale, N.J., (1988).
44. Polson, P. The role of how-it-works knowledge in the acquisition of how-to-do-it knowledge. In *Mental models and user-centered design*. A.A. Turner, Ed. Workshop Report. TR 88-9. University of Colorado, Institute of Cognitive Science.
45. Reisner, P. Formal grammar and human factors design of an interactive graphics system. *IEEE Trans. Softw. Eng. 7*, 229–240.
46. Rubinstein, R., and Hersh, H. *The human factor*. Digital Press, Bedford, Mass., 1984.
47. Schumacher, R.M., and Gentner, D. Remembering causal systems: Effects of systematicity and surface similarity in delayed transfer. In *Proceedings of the 32nd Annual Meeting of the Human Factors Society* (Anaheim, Calif.), pp. 1271–1275.
48. Shneiderman, B. *Designing the user interface*. Addison-Wesley, Reading, Mass., 1987.
49. Smith, S.L., and Mosier, J.N. Guidelines for designing user interface software. *Report 7 MTR-10090, Esd-Tr-86-278*. MITRE Corporation, Bedford, Mass., 1986.
50. Streeter, L.A., Ackroff, J.M., and Taylor, G.A. On abbreviating command names. *Bell System Tech. J. 62*, (1983), 1807–1826.
51. Suchman, L. *Plans and situated actions: The problem of human-machine communication*. Cambridge University Press, Cambridge; Mass., 1987.
52. Whiteside, J., Bennett, J., and Holtzblatt, K. Usability engineering: Our experience and evolution. In *Handbook of human-computer interaction*. M. Helander, Ed. Elsevier Science Publishers, Amsterdam, 1988.
53. Wiecha, C., Bennett, W., Boies, S., and Gould, J. Tools for generating consistent user interfaces. Manuscript draft.

CR Categories and Subject Descriptors: D.2.2 [**Software Engineering**]: Tools and Techniques; H.1.2 [**Information Systems**]: User Machine Systems

General Terms: Human Factors

Additional Key Words and Phrases: Consistency, user interfaces

---

ABOUT THE AUTHOR:

JONATHAN GRUDIN is a visiting member of the faculty of the Computer Science Department of Aarhus University. He is currently on leave from the Microelectronics and Computer Technology Corporation, where he is a member of the technical staff of the Human Interface Laboratory doing research on user interface design as a group process. His interest in user interface consistency was motivated by past work on large software development projects, as well as by earlier research in several areas described in this article. Author's Present Address: MCC Human Interface Laboratory, 3500 West Balcones Center Drive, Austin, TX 78759. grudin@mcc.com.