CHUNKING AND PHRASING AND THE DESIGN OF HUMAN-COMPUTER DIALOGUES[1]

William Buxton

Computer Systems Research Institute
University of Toronto
Toronto, Ontario
Canada, M5S 1A4
(416)-978-6320
buxton.dgp.toronto.edu

INTRODUCTION

It is no secret that the user interface of most computer systems could be improved. Systems are often intimidating, prone to error, and require a high investment in effort before productive work can be undertaken. A desire to make systems easier to use is a good starting point, but we can't get very far without some theory of how to do so.

"Easier to use" is easy to say, but it suggests little about *how* to reduce errors and frustration and promote faster learning. In order to make some headway in this direction, we might best reformulate the problem as "How can we accelerate the process whereby novices begin to perform like experts?". Underlying this formulation is an assumption that there is a qualitative difference between how experts and novices achieve particular goals. This assumption is supported by much of the recent literature in problem solving and the acquisition of cognitive skills (e.g., Anderson, 1980).

Experts and novices differ in the coarseness of granularity with which they view the constituent elements of a particular problem or task. Novices are *attentive* to low-level details. For example, *operational* details such as finding a particular character on the keyboard or remembering the name of a command involve *problem solving*. The result is that valuable cognitive resources are diverted from the central problem at hand.

With experts, these low-level details can be performed *automatically*. Hence, the size of the chunks of the problem to which they are attentive are much larger. The *skills* that permit these tasks to be performed automatically, however, must be highly learned, usually through repetition (Newell & Rosenbloom, 1980). The acquisition of skills, therefore, can be characterized by developing an ability to perform ever-larger chunks of a problem automatically.

---

[1] Citation: Buxton, W. (1986). Chunking and phrasing and the design of human-computer dialogues, *Proceedings of the IFIP World Computer Congress*, Dublin, Ireland, 475-480.

We can now return to our reformulation of the problem at hand, "How can we accelerate the process whereby novices begin to perform like experts?". Our premise is that there should be as close a match as possible between the structure of how we think about problems and the language or representation that we use in solving them. In what follows we argue that this can be achieved by engineering the *pragmatics* of the human-computer dialogue (Buxton, 1983) to reinforce the chunking that we believe would used by an expert working in the domain. Another way of stating this is that the dialogue structure, especially the pragmatics, can be engineered so as to maximize *compatibility* (Fitts & Seeger, 1953; John, Rosenbloom & Newell, 1985) with the problem domain.

SYNTAX: TWO APPROACHES

The design of the syntax has a major effect on the quality of the user interface of an interactive system. It affects learnability, the frequency and nature of user errors, the retention of skills (as with non-regular users) and the speed of task performance. A major problem for users is the cognitive load imposed by remembering the tokens of a command and their order (see, for example, Barnard, Hammond, Mortan, Long & Clark, 1981).

One approach that designers have taken to avoid such problems is to limit the number of arguments to a command. The user interface of the Macintosh computer, for example, limits operators to having only one explicit argument. This causes problems, however, for operations such as *move* which require both a direct and indirect object. To get around this, applications such as *MacWrite* (Apple, 1984) replace the single command *move* with two lower-level commands *cut* and *paste*. While the new primitives have a simpler syntax, the user's mental model must be restructured to map the concept move onto these two new primitives. Rather than simplifying the user interface, therefore, it is possible that the single-operand-per-verb strategy simply redistributes the cognitive loading.

An alternative design strategy exists. If *move*, for example, is the primitive that most closely corresponds to the user's model, then the design problem is to use it while minimizing the burden of remembering the arguments and their ordering. Proof-reader's symbols offer one approach to doing so. An example is shown in Figure 1.

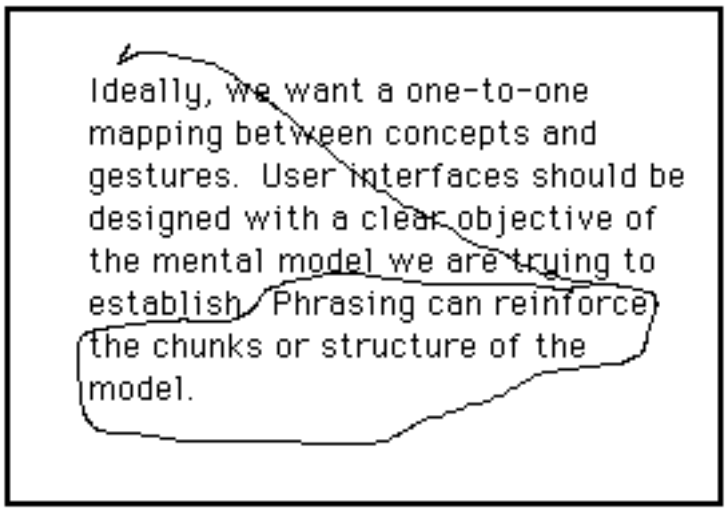

Figure 1: Proof-Reader's Symbol Specifying "Move." Contrast the directness of this with the "cut-and-paste" strategy utilized by *MacWrite* (Apple, 1984).

There are at least three points worth noting about this example, especially in contrast with the "cut-and-paste" strategy for specifying the same operation:

- the entire transaction, verb, direct object, and indirect object are all specified in a single gesture;
- there will never be an error in syntax since the ordering is implicit in the gesture;
- the operation is specified using existing skills and does not require restructuring of existing mental models.

PHRASING AND GESTURE

We can think about the components of the move command in the previous example as being woven together by a thread of continuity similar to that binds together a musical phrase. The "statement" is initiated in a state of neutrality, is articulated by a continuous gesture, and upon *closure*, returns to neutral state where another phrase can be introduced by either party. As in music, the phrase is characterized by tension (in this case muscular) and the neutral state delimiting the start and finish by relaxation.

One of our main arguments is that we can use tension and closure to develop a phrase structure to our human-computer dialogues which reinforces the chunking that we are trying to establish.

In the "body-language" of haptic input, kinesthetics and muscular tension are the raw materials of establishing a phrase structure. With the gesture comes heightened arousal and performance (Yerkes & Dodson, 1908), and in the periods of relaxation, a clear indication that it is aright to be interrupted, or move on to the next step.

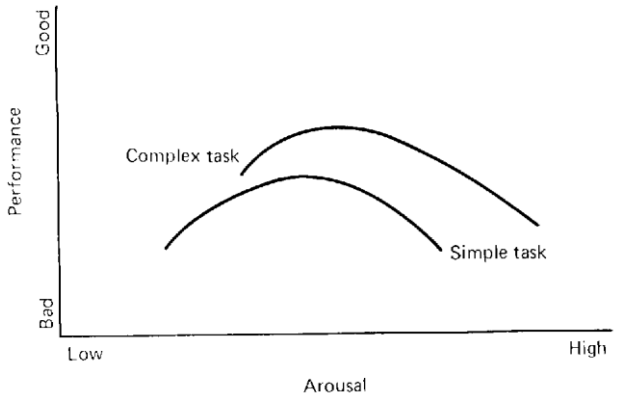

Figure 2: Yerkes-Dodson law relating performance to arousal (From Kantowitz & Sorkin, 1983, p. 606)

COMPOUND TASKS

Problems that we saw previously in the syntax of a single command also appear at another level of the human-computer dialogue. In actual applications, many of the transactions which we perform consist of *compound tasks*. Selecting an electrical component and positioning it in a circuit board layout would be an example of a *selection/positioning* task (Buxton, 1982). Similarly, identifying a word by finding it in a document and then selecting it would be an example of a *navigation/selection* task (Buxton & Myers, 1986). In many such cases, we would argue that the user models the compound task as a single entity. In such cases, having to address the sub-tasks independently may result in an additional burden comparable to using *cut* and *paste* instead of *move*. Furthermore, we claim that the use of phrasing through kinesthetic gesture can be used to overcome this problem.

Pop-up menus provide a good example to illustrate our point. In general, one would consider making a selection from a pop-up menu as being a single task. However, on closer examination, it is seen to consist of three sub-tasks:

- *invoke the menu* : by depressing the mouse button;
- *navigate to selection* : by moving mouse while button is depressed;
- *make selection and return* : release mouse button.

In this case, the "glue" that ties the three sub-tasks together is the tension of holding the mouse button down throughout the transaction. By designing the dialogue in this way, errors of syntax and mode errors are virtually impossible to make since the concluding action (articulated by the mouse button being released) is the unique and natural consequence to the initial action (depressing the mouse button). Furthermore, the tension of the finger holding down the button gives constant feedback that we are in a temporary state, or *mode*. (There is a slight irony to this, since it is precisely in so-called "modeless" interfaces that pop-up menus are most commonly found.)

PHRASING AND COGNITIVE SKILLS

In their 1983 study, Card, Moran and Newell discussed how experts collapsed low-level text editing tasks into cognitive "subroutines" that they termed "routine cognitive skills". Anderson (1982) describes the acquisition of such skills as being based upon the compilation and proceduralization of knowledge about the underlying sub-tasks. We believe that phrasing can be used to organize these sub-tasks to accelerate this process.

PRAGMATICS AND THE COMPONENTS OF INPUT

If Card, Moran and Newell's routine cognitive skills are compilations of lower-level primitives, one could try to determine the basic building blocks. One possible answer comes from Foley, Wallace and Chan (1984). They tried to characterize human input to computer systems from the user's perspective. In so doing, they came up with six basic primitives:

- *Select* an item in 1, 2, or 3D;
- *Position* an item in 1, 2, or 3D;
- *Orient* (rotate) an item in 1, 2, or 3D;
- *Path* : specify a path, such as in inking in a paint program;
- *Quantify* : specify a numerical value;
- *Text* : enter text, as in word processing

On closer examination, however, we see that these primitives are not necessarily all at the same level. Let us use the *position* primitive as an example.

If we use a mouse or a tablet, positioning an object in 2D can be viewed as a single task. However, the moment that we change transducers and use a QWERTY keyboard, specifying the same coordinates involves two primitives, namely *quantify X* and *quantify Y*.

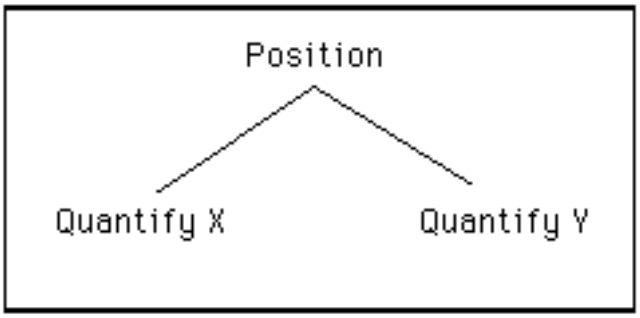

Figure 3. Position as an Aggregate of 2 Quantify Tasks

We see from this example that even Foley, Wallace and Chan's six primitives have a deep structure. Whether the sub-tasks are consciously perceived, however, is very much influenced by the gesture (and capturing transducer) used. When appropriate, a single gesture (pointing) can be used to articulate a single concept (position).

We can build further upon the previous example. Let us look at a simple system for transcribing common music notation (Buxton, Sniderman, Reeves, Patel & Baecker,

1979). Notes are entered using a simple short-hand notation. Using a stylus and digitizing tablet, the user points at where a note is to appear and enters one of the shorthand symbols shown in Figure 4.

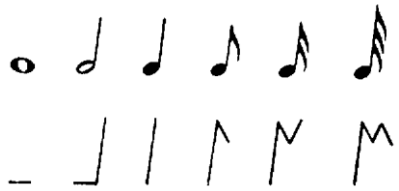

Figure 4. Short-Hand Symbols for Transcribing Musical Notation. (From Buxton, Sniderman, Reeves, Patel & Baecker, 1979.)

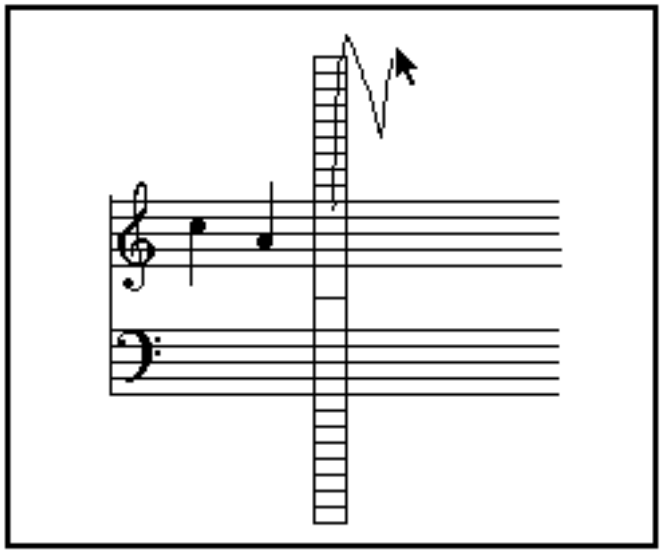

Figure 5. Entering a 16th Note Using a Single Gesture.

Using this system to enter a 16th note is shown in Figure 5.

The underlying structure of adding notes using this technique is shown in Figure 6. We see that adding a note, like positioning, is actually made up of a number of sub-tasks. However, when implemented as described, these sub-tasks all collapse into the single primitive *add note*.

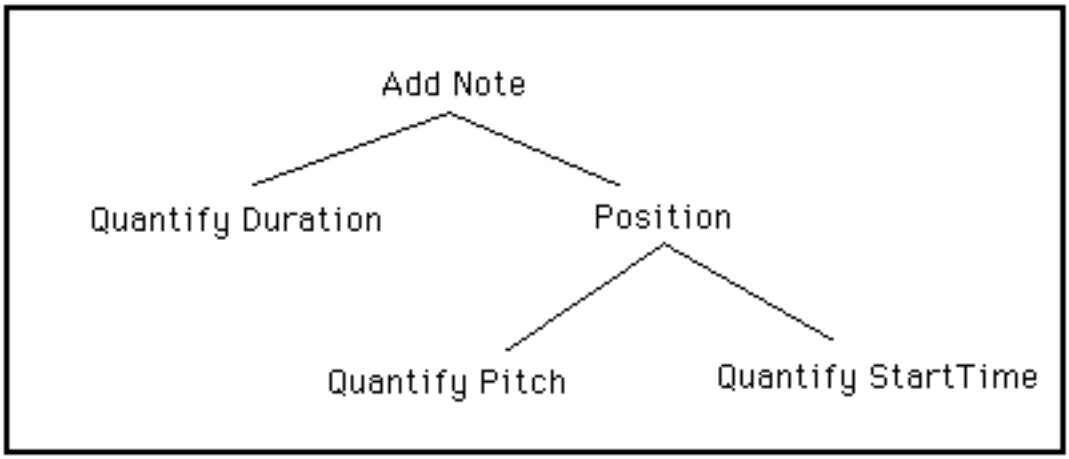

Figure 6. Task Hierarchy in Add Note Task

GRAMMARS, PRAGMATICS AND COMPLEXITY

If the sub-tasks of a higher-level concept like "Add Note" can collapse into a single gesture, is there any model that helps predict the savings? One approach, based on recent studies, attempts to obtain a measure of the difficulty of a user interface by analyzing its underlying grammar. This work was pioneered by Reisner (1981), and further developed by Green & Payne (1984) and Green, Payne, Gilmore and Mepham (1984).

In her original work, Reisner developed a set of heuristics which she used to analyze the grammar of the interaction language of a particular system. From this analysis she would derive a value which gave a measure of the system's learnability and proneness to error. The heuristics that she used were based upon:

- Number of productions
- Number of terminals
- Length of productions

We can apply such an analysis to the grammar of our Add Note primitive, shown below (non-terminals start with upper case, terminals start with lower case):

AddNote:= quantifyDuration PositionPitchTime
PositionPitchTime := quantifyPitch quantifyStartTime

We can apply an approximation of Reisner's heuristics on this grammar in which we assume that the weight of each production and each terminal is 1 unit. Since we have two productions and three terminals, the total weight is therefore 5.

However, if we use the character recognition technique described above, we would argue (from experience) that the *real* weight of the entire transaction is closer to the weight contributed by a single terminal, namely weight 1. Our explanation for this is that the user need not be attentive to any of the operational details of the component sub-tasks. The complete concept can be expressed in a single fluid compatible gesture.

GESTURE AND PRAGMATICS

The notion of physical gesture is central to virtually all of the examples discussed. In each case, the key to using a particular gesture is having the appropriate transducer. Conversely, the main limiting factor, restricting the range of available gestures in our repertoire is the sorry state of current practice in input. The lack of pressure sensitive devices (such as mouse buttons to control line thickness), foot controls, and two-handed input are just a few obvious examples.

To this point all of our examples have involved sequential bindings. However, as changing gears with a manual transmission illustrates, the binding among related tasks can be in parallel and across limbs. This is demonstrated in a recent study by Buxton & Myers (1986).

CONCLUSIONS

We have argued that user interface pragmatics can be designed to accelerate the acquisition of expert operational skills. The key is gesture-based phrasing to chunk the dialogue into units meaningful to the application. Any concept or transaction that can be described in a single word or phrase should be able to be articulated by a single gesture. This desired one-to-one correspondence between concept and gesture leads towards interfaces which are more *compatible* with the user's model.

The work described is based on practice and experience rather than formal experimentation. It is preliminary, and a great deal of research remains to be done. However, the examples discussed are sufficiently persuasive to warrant an examination of current design practice.

REFERENCES/BIBLIOGRAPHY

Anderson, J.R. (Ed.) (1980). *Cognitive Skills and their Acquisition*, Hillsdale, N.J.: Lawrence Erlbaum & Associates.

Anderson, J.R. (1982). Acquisition of Cognitive Skill, *Psychological Review*, **89(4),** 369-406.

Apple (1984), *MacWrite User's Manual*, Apple Computer Inc., Cupertino, CA.

Barnard, P.J., Hammond, N.V., Mortan, J., Long, J.B. & Clark, I.A. (1981). Consistency and Compatibility in Human-Computer Dialogue, *IJMMS*, **15(1),** 87-134.

Buxton, W. (1982). An Informal Study of Selection-Positioning Tasks, *Proceedings of Graphics Interface '82*, 323-328.

Buxton, W. (1983), Lexical and Pragmatic Considerations of Input Structure, *Computer Graphics*, **17(1)**, 31-37.

Buxton, W. & Myers, B. (1986). A Study in Two-Handed Input, *Proceedings of CHI '86*, to appear.

Buxton, W., Sniderman, R., Reeves, W., Patel, S. & Baecker, R. (1979). The Evolution of the SSSP Score Editing Tools. *Computer Music Journal* , **3(4),** 14-25.

Card, S., Moran, T. & Newell, A. (1983), *The Psychology of Human-Computer Interaction*, Hillsdale, N.J.: Lawrence Erlbaum Associates.

Fitts, P.M. & Seeger, C.M. (1953), S-R compatability: Spatial characteristics of stimulus and response codes. *Journal of Experimental Psychology,* **46,** 199-210.

Foley, J., Wallace, V.L. & Chan, P.(1984), The Human Factors of Computer Graphics Interaction Techniques, *IEEE CG&A*, **4(11),** 13-48.

Green, T.R.G. & Payne, S.J. (1984). Organization and Learnability in Computer Languages, *IJMMS* , **21(1)**, 7-18.

Green, T.R.G., Payne, S.J., Gilmore, D.J. & Mepham, M. (1984). Predicting Expert Slips, *Proceedings of Interact '84*, Vol. 1, 92-98.

John, Rosenbloom, P.S. & Newell, A. (1985). A Theory of Stimulus-Response Compatability Applied to Human-Computer Interaction, *Proceedings of CHI'85*, 213-220.

Kantowitz, B.H. & Sorkin, R.D. (1983). *Human Factors: Understanding People-System Relationships*, New York: John Wiley & Sons.

Newell, A. & Rosenbloom, P.S. (1980). Mechanisms of Skill Acquisition and the Law of Practice, in Anderson, J.R. (1980). *op cit.*

Reisner, P. (1981). Formal Grammar and Human Factors Design of an Interactive Graphics System, *IEEE Transactions on Software Engineering*, **7 (2)**, 229-240.

Yerkes, R.M. & Dodson, J.D. (1908). The Relative Strength of Stimulus to Rapidity of Habit-Formation, *Journal of Comparative and Neurological Psychology*, **18**, 459-482.