Lexical and Pragmatic Considerations of Input Structures

William Buxton
Computer Systems Research Group
University of Toronto
Toronto, Ontario
Canada M5S 1A4

Introduction

Increased access to computer-based tools has made only too clear the deficiencies in our ability to produce effective user interfaces [1]. Many of our current problems are rooted in our lack of sufficiently powerful theories and methodologies. User interface design remains more of a creative art than a hard science.

Following an age-old technique, the point of departure for much recent work has been to attempt to impose some structure on the problem domain. Perhaps the most significant difference between this work and earlier efforts is the weight placed on considerations falling outside the scope of conventional computer science. The traditional problem-reduction paradigm is being replaced by a holistic approach which views the problem as an integration of issues from computer science, electrical engineering, industrial design, cognitive psychology, psychophysics, linguistics, and kinesthetics.

In the main body of this paper, we examine some of the taxonomies which have been proposed and illustrate how they can serve as useful structures for relating studies in user interface problems. In so doing, we attempt to augment the power of these structures by developing their ability to take into account the effect of gestural and positional factors on the overall effect of the user interface.

Two Taxonomies

One structure for viewing the problem domain of the user interface is provided by Foley and Van Dam [12]. They describe the space in terms of the following four layers:

- conceptual
- semantic
- syntactic
- lexical

The *conceptual* level incorporates the main concepts of the system as seen by the user. Therefore, Foley and Van Dam see it as being equivalent to the *user model.* The *semantic* level incorporates the functionality of the system: what can be expressed. The *syntactic* level defines the grammatical structure of the tokens used to articulate a semantic concept. Finally, the *lexical* component defines the structure of these tokens.

One of the benefits of such a taxonomy is that it can serve as the basis for systems analysis in the design process. It also helps us categorize various user interface studies so as to avoid "apples and bananas" type of comparisons. For example, the studies of Ledgard, Whiteside, Singer and Seymour [16] and Barnard, Hammond, Morton and Long [3] both address issues at the syntactic level. They can, therefore, be compared (which is quite interesting since they give highly contradictory results[1]). On the other hand, by recognizing the "keystroke" model of Card, Moran and Newell [7] as addressing the lexical level, we have a good way of understanding its limitations and comparing it to related studies (such as Embley, Lan, Leinbaugh and Nagy, [8]), or relating it to studies which address different levels (such as the two studies in syntax mentioned above).

While the taxonomy presented by Foley and Van Dam has proven to be a useful tool, our opinion is that it has one major shortcoming. That is, the grain of the lexical level is too coarse to permit the full benefit of the model to be derived. As defined, the authors lump together issues as diverse as:

- how tokens are spelt (for example "add" *vs* "append" *vs* "a" *vs* some graphical icon)

[1] Barnard *et al* invalidate Ledgard *et al's* main thesis that the syntax of natural language is necessarily the best suited for command languages. They demonstrate cases where fixed-field format is less prone to user error than the direct object — indirect object syntax of natural language. A major problem of the paper of Ledgard *et al* is that they did not test many of the interesting cases and then drew conclusions that went beyond what their results supported.

- where items are placed spatially on the display (both in terms of the layout and number of windows, and the layout of data within those windows)
- where devices are placed in the work station
- the type of physical gesture (as determined by the transducer employed) used to articulate a token (pointing with a joystick *vs* a lightpen *vs* a tablet *vs* a mouse, for example)

These issues are sufficiently different to warrant separate treatment. Grouping them under a single heading has the danger of generating confusion comparable to that which could result if no difference was made between the semantic and syntactic levels. Therefore, taking our cue from work in language understanding research in the AI community, we chose to subdivide Foley and Van Dam's lexical level into the following two components:

- lexical: issues having to do with spelling of tokens (*i.e.*, the ordering of lexemes and the nature of the alphabet used — symbolic or iconic, for example).
- pragmatic: issues of gesture, space and devices.

To illustrate the distinction, in the Keystroke model the number of key pushes would be a function of the *lexical* structure while the homing time and pointing time would be a function of *pragmatics.*

Factoring out these two levels helps us focus on the fact that the issues affecting each are different, as is their influence on the overall effect of the user interface. This is illustrated in examples which are presented later in this paper.

It should be pointed out that our isolation of what we have called pragmatic issues is not especially original. We see a similar view in the Command Language Grammar of Moran [18], which is the second main taxonomy which we present. Moran represents the domain of the user interface in terms of three components, each of which is sub-divided into two levels. These are as follows:

- Conceptual Component
  - task level
  - semantic level
- Communication Component
  - syntactic level
  - interaction level
- Physical Component
  - spatial level
  - device level

The *task level* encompasses the set of tasks which the user brings to the system and for which it is intended to serve as a tool. The *semantic level* lays out the conceptual entities of the system and the conceptual operations upon them. As with the Foley and Van Dam model, the *syntactic level* then incorporates the structure of the language within which the semantic level is embedded. The *interaction level* relates the user's physical actions to the conventions of the interactions in the dialogue. The *spatial level* then encompasses issues related to how information is laid out on the display, while the *device level* covers issues such as what types of devices are used and their properties (for example, the effect on user performance if the locator used is a mouse *vs* an isometric joystick *vs* step-keys). (A representative discussion of such issues can be found in Card, English and Burr, [5].)

One subtle but important emphasis in Moran's paper is on the point that it is the effect of the user interface *as a whole* (that is, all levels combined) which constitutes the user's model. The other main difference of his taxonomy, when compared to that of Foley and Van Dam, is his emphasis on the importance of the physical component. A shortcoming, however, lies in the absence of a slot which encapsulates the lexical level as we have defined it above. Like the lexical level (as defined by Foley and Van Dam), the interaction level of Moran appears a little too broad in scope when compared to the other levels in the taxonomy.

Pragmatics

In examining the two studies discussed above, one quickly recognizes that the effect of the pragmatic level on the user interface, and therefore on the user model, is given very little attention. Moran, for example, points out that the physical component exists and that it is important, but does not discuss it further. Foley and Van Dam bury these issues within the lexical level. Our main thesis is that since the primary level of contact with an interactive system is at the level of pragmatics, this level *has one of the strongest effects on the user's perception of the system.* Consequently, the models which we adopt in order to specify, design, implement, compare and evaluate interactive systems *must* be sufficiently rich to capture and communicate the system's properties at this level. This is clearly not the case with most models, and this should be cause for concern. To illustrate this, let us examine a few case studies which relate the effect of pragmatics to:

- pencil-and-paper tests of query languages
- ease of use with respect to action language grammars
- device independence

Pencil-and-Paper Tests

As an aid to the design of effective data base query languages, Reisner [19] has proposed the use of pencil-and-paper tests. Subjects were taught a query language in a class-room environment and then tested as to their

ability to formulate and understand queries. Different control groups were taught different languages. By comparing the test results of the different groups, Reisner drew conclusions as to the relative "goodness" of structure and ease of learning of the different languages. She then made the argument that the technique could be used to find weaknesses in new languages before they are implemented, thereby shortening their development cycle.

While the paper makes some important points, it has a serious defect in that it does not point out the limitations of the technique. The approach does tell us something about the cognitive burden involved in the learning of a query language. But it does not tell us everything. In particular, the technique is totally incapable of taking into account the effect that the means and medium of doing something has on our ability to remember how to do it. To paraphrase McLuhan, the medium does affect the message.

Issues of syntax are not independent of pragmatics, but pencil-and-paper tests cannot take such dependencies into account. For example, consider the role of "muscle memory" in recalling how to perform various tasks. The strength of its influence can be seen in my ability to type quite effectively, even though I am incapable of telling you where the various characters are on my QWERTY keyboard, or in my ability to open a lock whose combination I cannot recite. Yet, this effect will never show up in a pencil-and-paper test. Another example is seen in the technique's inability to take into account the contribution that appropriate feedback and help mechanisms can provide in developing mnemonics and other memory and learning aids.

We are not trying to claim that such pencil-and-paper tests are not of use (although Barnard *et al*, [3], point out some important dangers in using such techniques). We are simply trying to illustrate some of their limitations, and demonstrate that lack of adequate emphasis on pragmatics can result in readers (and authors) drawing false or misleading conclusions from their work. Furthermore, we conjecture that if pragmatics were isolated as a separate level in a taxonomy such as that of Foley and Van Dam, they would be less likely to be ignored.

Complexity and Chunking

In another study, Reisner [20] makes an important contribution by showing how the analysis of the grammar of the "action language" of an interactive system can provide valuable metrics for predicting the ease of use and proneness to error of that system. Thus, an important tool for system design, analysis and comparison is introduced.

The basis of the technique is that the complexity of the grammar is a good metric for the cognitive burden of learning and using the system. Grammar complexity is measured in terms of number of productions and production length. There is a problem, however, which limits our ability to reap the full benefits of the technique. This has to do with the technique's current inability to take into account what we call *chunking*. By this we mean the phenomenon where two or more actions fuse together into a single gesture (in a manner analogous to the formation of a compound word in language). In many cases, the cognitive burden of the resulting aggregate may be the equivalent of a single token. In terms of formal language theory, a non-terminal *when effected by an appropriate compound gesture* may carry the cognitive burden of a single terminal.

Such chunking may be either sequential, parallel or both. Sequentially, it should be recognized that some actions have different degrees of *closure* than others. For example, take two events, each of which is to be triggered by the change of state of a switch. If a foot-switch similar to the high/low beam switch in some cars is used, the down action of a down/up gesture triggers each event. The point to note is that there is no kinesthetic connection between the gesture that triggers one event and that which triggers the other. Each action is complete in itself and, as with driving a car, the operator is free to initiate other actions before changing the state of the switch again.

On the other hand, the same binary function could be controlled by a foot pedal which functions like the sustain pedal of a piano. In this case, one state change occurs on depression, a second on release. Here, the point to recognize is that the second action is a direct consequent of its predecessor. The syntax is implicit, and the cognitive burden of remembering what to do after the first action is minimal.

There are many cases where this type of kinesthetic connectivity can be bound to a sequence of tokens which are logically connected. One example given by Buxton [4] is in selecting an item from a graphics menu and "dragging" it into position in a work space. A button-down action (while pointing at an item) "picks it up." For as long as the button is depressed, the item tracks the motion of the pointing device. When the button is released, the item is anchored in its current position. Hence, the interface is designed to force the user to follow proper syntax: select then position. There is no possibility for syntactic error, and cognitive resources are not consumed in trying to remember "what do I do next?". Thus, by recognizing and exploiting such cases, interfaces can be constructed which are "natural" and easy to learn.

There is a similar type of chunking which can take place when two or more gestures are articulated at one time. Again we can take an example from driving a car, where in changing gears the actions on the clutch, accelerator and gear-shift reinforce one another and are coordinated into a single gesture. Choosing appropriate gestures for such coordinated actions can accelerate their bonding into what the user thinks of as a single act,

thereby freeing up cognitive resources to be applied to more important tasks. What we are arguing here is that by matching appropriate gestures with tasks, we can help render complex skills routine and gain benefits similar to those seen at different level in Card, Moran and Newell [6].

In summary, there are three main points which we wish to make with this example:

- there is an important interplay between the syntactic-lexical levels and the pragmatic level
- that this interplay can be exploited to reduce the cognitive burden of learning and using a system
- that this cannot be accomplished without a better understanding of pragmatic issues such as chunking and closure.

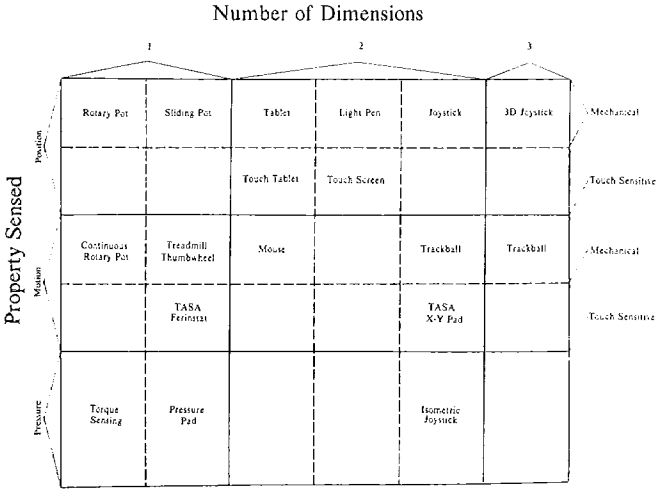
Pragmatics and Device Independence

We began by declaring the importance of being able to incorporate pragmatic issues into the models which we use to specify, design, compare and evaluate systems. The examples which followed then illustrated some of the reasons for this belief. When we view the CORE proposal [13, 14] from this perspective, however, we see several problems. The basis of how the CORE system approaches input is to deal with user actions in terms of abstractions, or logical devices (such as "locators" and "valuators"). The intention is to facilitate software portability. If all "locators," for example, utilized a common protocol, then user A (who only had a mouse) could easily implement software developed by B (who only had a tablet). From the application programmer's perspective, this is a valuable feature. However, for the purposes of specifying systems from the user's point of view, these abstractions are of very limited benefit. As Baecker [2] has pointed out, the effectiveness of a particular user interface is often due to the use of a *particular* device, and that effectiveness will be lost if that device were replaced by some other of the same logical class. For example, we have a system [10] whose interface depends on the simultaneous manipulation of four joysticks. Now in spite of tablets and joysticks both being "locator" devices, it is clear that they are not interchangeable in this situation. We cannot simultaneously manipulate four tablets. Thus, for the full potential of device independence to be realized, such pragmatic considerations must be incorporated into our overall specification model so that appropriate equivalencies can be determined in a methodological way. (That is, in specifying a generic device, we must also include the required pragmatic attributes. But to do so, we must develop a taxonomy of such attributes, just as we have developed a taxonomy of virtual devices.)

A Taxonomy of Devices

In view of the preceding discussion, we have attempted to develop a taxonomy which helps isolate relevant characteristics of input devices. The tableau shown in Figure 1 summarizes this effort in a two dimensional representation. The remainder of this section presents the details and motivation for this tableau's organization.

Figure 1. Tableau of Continuous Input Devices



To begin with, the tableau deals only with continuous hand-controlled devices. (Pedals, for example, are not included for simplicity's sake.) Therefore the first (but implicit) questions in our structure are:

- continuous *vs* discrete?
- agent of control (hand, foot, voice, ...)?

The table is divided into a matrix whose rows and columns delimit

- what is being sensed (position, motion or pressure), and
- the number of dimensions being sensed (1, 2 or 3),

respectively. These primary partitions of the matrix are delimited by solid lines. Hence, both the rotary and sliding potentiometer fall into the box associated with one-dimensional position-sensitive devices (top left-hand corner).

Note that the primary rows and columns of the matrix are sub-divided, as indicated by the dotted lines. The sub-columns exist to isolate devices whose control motion is roughly similar. These groupings can be seen in examining the two-dimensional devices. Here the tableau implies that tablets and mice utilize similar types of hand control and that this control is different from that shared in using a light-pen or touch-screen. Furthermore, it is shown that joysticks and trackballs share a common control motion which is, in turn, different than the other sub-classes of two-dimensional devices.

The rows for *position* and *motion* sensing devices are subdivided in order to differentiate between transducers which sense potential *via* mechanical *vs* touch-sensitive means. Thus, we see that the light-pen and touch-screen are closely related, except that the light-pen employs a mechanical transducer. Similarly, we see that trackball and TASA touch-pad[2] provide comparable signals from comparable gestures (the 4" by 4" dimensions of the TASA device compare to a 3 1/2" diameter trackball).

The tableau is useful for many purposes by virtue of the structure which it imposes on the domain of input devices. First, it helps in finding appropriate equivalences. This is important in terms of dealing with some of the problems which arose in our discussion of device independence. For example, we saw a case where four tablets would not be suitable for replacing four joysticks. By using the tableau, we see that four trackballs will probably do.

The tableau makes it easy to relate different devices in terms of metaphor. For example, a tablet is to a mouse what a joystick is to a trackball. Furthermore, if the taxonomy defined by the tableau can suggest new transducers in a manner analogous to the periodic table of Mendeleev predicting new elements, then we can have more confidence in its underlying premises. We make this claim for the tableau and cite the "torque sensing" one-dimensional pressure-sensitive transducer as an example. To our knowledge, no such device exists commercially. Nevertheless it is a potentially useful device, an approximation of which has been demonstrated by Herot and Weinzaphel [15].

Finally, the tableau is useful in helping quantify the generality of various physical devices. In cases where the work station is limited to one or two input devices, then it is often in the user's interest to choose the least constraining devices. For this reason, many people claim that tablets are the preferred device since they can emulate many of the other transducers (as is demonstrated by Evans, Tanner and Wein, [9]). The tableau is useful in determining the degree of this generality by "filling in" the squares which can be adequately covered by the tablet.

Before leaving the topic of the tableau, it is worth commenting on why a primary criterion for grouping devices was whether they were sensitive to position, motion or pressure. The reason is that what is sensed has a *very* strong effect on the nature of the dialogues that the system can support with any degree of fluency. As an example, let us compare how the user interface of an instrumentation console can be affected by the choice of whether motion or position sensitive transducers are used. For such consoles, one design philosophy follows the traditional model that for every function there should be a device. One of the rationales behind this approach is to avoid the use of "modes" which result when a single device must serve for more than one function. Another philosophy takes the point of view that the number of devices required in a console need only be in the order of the control bandwidth of the human operator. Here, the rationale is that careful design can minimize the "mode" problem, and that the resulting simple consoles are more cost-effective and less prone to breakdown (since they have fewer devices).

One consequence of the second philosophy is that the same transducer must be made to control different functions, or parameters, at different times. This context switching introduces something known as the *nulling problem*. The point which we are going to make is that this problem can be completely avoided if the transducer in question is motion rather than position sensitive. Let us see why.

Imagine that you have a sliding potentiometer which controls parameter A. Both the potentiometer and the parameter are at their minimum values. You then raise A to its maximum value by pushing up the position of the potentiometer's handle. You now want to change the value of parameter B. Before you can do so using the same potentiometer, the handle of the potentiometer must be repositioned to a position corresponding to the current value of parameter B. The necessity of having to perform this normalizing function is the nulling problem.

Contrast the difficulty of performing the above interaction using a position-sensitive device with the ease of doing so using one which senses motion. If a thumb-wheel or a treadmill-like device was used, the moment that the transducer is connected to the parameter it can be used to "push" the value up or "pull" it down. Furthermore, the same transducer can be used to simultaneously change the value of a group of parameters, all of whose instantaneous values are different.

Horizontal vs Vertical Strata

The above example brings up one important point: the different levels of the taxonomies of Foley and Van Dam or of Moran are not orthogonal. By describing the user interface in terms of a horizontal structure, it is very easy to fall into the trap of believing that the effect of modifications at one level will be isolated. This is clearly not true as the above example demonstrated: the choice of transducer type had a strong effect on syntax.

The example is not isolated. In fact, just as strong an argument could be made for adopting a model based on a vertical structure as the horizontal ones which we have discussed. Models based on interaction techniques such as those described in Martin [17] and Foley, Wallace and Chan [11] are examples. With them, the primary gestalt is the transaction, or interaction. The user model is described in terms of the set and style of

[2] The TASA X-Y 360 is a 4" by 4" touch sensitive device which gives 60 units of delta modulation in 4 inches of travel. The device is available from TASA, 2346 Walsh Ave., Santa Clara CA, 95051.

the interactions which take place over time. Syntactic, lexical and pragmatic questions become sub-issues.

Neither the horizontal or vertical view is "correct." The point is that *both* must be kept in mind during the design process. A major challenge is to adapt our models so that this is done in a well structured way. That we still have problems in doing so can be seen in Moran's taxonomy. Much of the difficulty in understanding the model is due to problems in his approach in integrating vertically oriented concepts (the interaction level) into an otherwise horizontal structure.

In spite of such difficulties, both views must be considered. This is an important cautionary bell to ring given the current trend towards delegating personal responsibilities according to horizontal stratification. The design of a system's data-base, for example, has a very strong effect on the semantics of the interactions that can be supported. If the computing environment is selected by one person, the data-base managed by another, the semantics or functional capability by another, and the "user interface" by yet another, there is an inherent danger that the decisions of one will adversely affect another. This is not to say that such an organizational structure cannot work. It is just imperative that we be aware of the pitfalls so that they can be avoided. Decisions made at all levels affect one another and *all* decisions potentially have an effect on the user model.

Summary and Conclusions

Two taxonomies for describing the problem domain of the user interface were described. In the discussion it was pointed out that the outer levels of the strata, those concerning lexical, spatial, and physical issues were neglected. The notion of pragmatics was introduced in order to facilitate focusing attention on these issues. Several examples were then examined which illustrated why this was important. In so doing, it was seen that the power of various existing models could be extended if we had a better understanding of pragmatic issues. As a step towards such an understanding, a taxonomy of hand controlled continuous input devices was introduced. It was seen that this taxonomy made some contribution towards addressing problems which arose in the case studies. It was also seen, however, that issues at this outer level of devices had a potentially strong effect on the other levels of the system. Hence, the danger of over-concentration on horizontal stratification was pointed out.

The work reported has made some contribution towards an understanding of the effect of issues which we have called pragmatics. It is, however, a very small step. While there is a great deal of work still to be done right at the device level, perhaps the biggest challenge is to develop a better understanding of the interplay among the different levels in the strata of a system. When we have developed a methodology which allows us to determine the gesture that best suits the expression of a particular concept, then we will be able to build the user interfaces which today are only a dream.

Acknowledgements

The ideas presented in this paper have developed over a period of time and owe much to discussions with our students and colleagues. In particular, a great debt is owed to Ron Baecker who was responsible for helping formulate many of the ideas presented. In addition, we would like to acknowledge the contribution of Alain Fournier, Russel Kirsch, Eugene Fiume and Ralph Hill in the intellectual development of the paper, and the help of Monica Delange in the preparation of the manuscript. Finally, we gratefully acknowledge the financial support of the National Sciences and Engineering Research Council of Canada.

References

1. Baecker, R. Human-computer interactive systems: a state-of-the-art review. In P. Kolers, E. Wrolftad & H. Bouma, Eds., *Processing of Visible Language II*, New York: Plenum, (1980), 423–444.
2. Baecker, R. Towards an effective characterization of graphical interaction. In R. A. Guedj, P. Ten Hagen, F. Hopgood, H. Tucker & D. Duce, Eds., *Methodology of Interaction*, Amsterdam: North-Holland, (1980), 127–148.
3. Barnard, P., Hammond, N., Morton, J., and Long, J. Consistency and compatibility in human-computer dialogue. *International Journal of Man-Machine Studies 15*, (1981), 87–134.
4. Buxton, W. An informal study of selection-positioning tasks. *Proceedings of Graphics Interface '82*, Toronto, (1982), 323–328.
5. Card, S., English, W., and Burr, B. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT. *Ergonomics 8*, (1978), 601–613.
6. Card, S., Moran, T., and Newell, A. Computer text editing: an information-processing analysis of a routine cognitive skill. *Cognitive Psychology 12*, (1980),32–74.
7. Card, S., Moran, T., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Communications of the ACM 23*, 7 (1980), 396–410.
8. Embley, D., Lan, M., Leinbaugh, D., and Nagy, G. A procedure for predicting program editor performance from the user's point of view. *International Journal of Man-Machine Studies 10*, (1978), 639–650.

9. Evans, K., Tanner, P., and Wein, M. Tablet-based valuators that provide one, two, or three degrees of freedom. *Computer Graphics 15*, 3 (1981), 91–97.
10. Fedorkow, G., Buxton, W., and Smith, K. C. A computer controlled sound distribution system for the performance of electroacoustic music. *Computer Music Journal 2*, 3 (1978), 33–42.
11. Foley, J., Wallace, V., and Chan, P. The human factors of interaction techniques. Technical Report GWU-IIST-81-03, Washington: The George Washington University, Institute for Information Science and Technology, (1981).
12. Foley, J. and Van Dam, A. *Fundamentals of Interactive Computer Graphics*. Reading, MA: Addison Wesley, (1982).
13. GSPC. Status Report of the Graphics Standards Planning Committee. *Computer Graphics 11*, (1977).
14. GSPC. Status Report of the Graphics Standards Committee. *Computer Graphics 13*, 3 (1979).
15. Herot, C. and Weinzaphel, G. One-point touch input of vector information for computer displays. *Computer Graphics 12*, 3 (1978), 210–216.
16. Ledgard, H., Whiteside, J., Singer, A., and Seymour, W. The natural language of interactive systems. *Communications of the ACM 23*, 10 (1980), 556–563.
17. Martin, J. *Design of Man-Computer Dialogues*. Engelwood Cliffs, NJ: Prentice-Hall, (1973).
18. Moran, T. The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies 15*, (1981), 3–50.
19. Reisner, P. Use of psychological experimentation as an aid to development of a query language. *IEEE Transactions on Software Engineering 3*, 3 (1977), 218–229.
20. Reisner, P. Formal grammar and human factors design of an interactive graphics system. *IEEE Transactions on Software Engineering 7*, 2 (1981), 229–240.