7
The Application of User Knowledge to Interface Design

JAMES E. MCDONALD
ROGER W. SCHVANEVELDT

In this chapter we discuss a methodology appropriate for interface *design*, which we define as the process whereby the designer plans and specifies the user interface. The interface designer's objective is to make devices easier and more enjoyable to use, and to make it possible for people to perform tasks which they might not otherwise be able to accomplish. With this objective in mind, we will propose a guiding interface design principle and offer a methodology for applying this principle to different interfaces. We would like to stress at the outset that ours is a methodology under development and that it needs additional empirical support. This approach to interface design is worth considering because it is empirically based, potentially formal, and in concert with current thinking in cognitive science. It is, of course, up to the reader to decide whether or not the arguments and data presented are sufficient to justify such optimism. In the following sections we first discuss some common approaches to interface design in order to place our methodology in perspective. We then present the theoretical motivations for our approach, along with a related discussion of scaling and knowledge acquisition techniques, followed by a review of several applications that illustrate key aspects of our methodology. The last of these studies is an ongoing investigation of UNIX users aimed at improving the on-line documentation system.

1. Interface Design Methods

The most common method used to design interfaces assigns the task of building user interfaces to the engineers or computer scientists responsible for the rest of the system. In many cases the interface is not given any special status, in that it is not treated as a separate component, but simply viewed as another facet of the system hardware and/or software. We and others have argued that this approach is unsatisfactory for both

designers and users (Norman, 1982; Schvaneveldt, McDonald, & Cooke, 1985). The approach is unsatisfactory for designers because they must engage in activities for which they are not specifically trained and for which no detailed specifications exist. It is unsatisfactory for users because the resulting interfaces are often idiosyncratic, based on the preferences of designers, and arcane, because the designer's and user's perspectives **are** different (Manheimer, Stone, & McDonald, 1983). Another consequence of this approach is that those responsible for designing interfaces are often inclined to place cost and efficiency above usability when the inevitable cost vs. usability trade-offs must be made.

1.1 Interface Standards and Guidelines

One solution to the problems described above might be to develop interface *standards* and *guidelines* (e.g., the German DIN standards and IBM's UIA guidelines). Such guidelines tend to be loosely organized collections of recommendations based on facts, experimental results, and opinions. Unfortunately, the recommendations in typical guidelines are based on too few facts and experimental results and far too many opinions. Although well motivated, the soundness of the guideline approach must be questioned on several grounds. First, even when supported by data, many of the recommendations made in guidelines are not valid for specific applications. When recommendations are general (i.e., derived from basic research), they are suspect because particular applications are likely to include factors which invalidate them. On the other hand, when the recommendations in guidelines are derived from studies of particular applications (i.e., based on applied research), unique characteristics of these applications make general conclusions impossible and, again, may invalidate the recommendations.

The main supporters of guidelines are labor unions, manufacturers, and engineers, but there are some human-factors practitioners who favor them as well. The supporters seem to be motivated by the belief that, in the absence of knowledge about how to optimally design interfaces, some benefit can be obtained by designing interfaces in a *consistent* fashion. Human-computer interfaces do exhibit surprising variability, even in their most basic characteristics. A notorious example is the

placement and meaning of auxiliary keys on keyboards (e.g., shift, return, control, and programmable function keys). Other puzzling inconsistencies, difficult to understand without some knowledge of their evolution, include differences in command naming and syntax. Many computer users are unpleasantly surprised when they attempt to transfer their knowledge from one application to a similar one. For example, someone familiar with a text editor such as *vi*, will have considerable difficulty applying his or her knowledge to a different text editor, such as *emacs*, even when using the same terminal hardware and the same operating system (i.e., UNIX). Even greater difficulties may be encountered if the transfer of knowledge involves different hardware and/or a new operating system (e.g., DOS). Although such difficulties may be explicable from the developers' perspective, they really cannot be justified from the users' standpoint.

It would be a mistake, however, to conclude that standardization is a good solution to the most important interface design problems. More often than not, applications are hard to use because of bad interface design, and standardization would only exacerbate these problems. If, as many believe, *all* current interfaces are less than optimal, then standardization will simply doom us to inferior interfaces for years to come. Emerson observed that "a *foolish* consistency is the hobgoblin of little minds..." It would indeed be foolish to insist on consistent interface design before learning more about good interface design.

A final problem with guidelines is that they do not necessarily accomplish one of their main objectives; they do not necessarily make things easier for designers. Guidelines require interpretation, often by designers unfamiliar with psychological methods or jargon who are therefore ill-equipped for the task. Thus, the guideline approach may still result in inconsistently designed interfaces and actually increase the burden on interface designers by requiring them to consult large collections of ambiguous recommendations.

1.2 Creative Insights

The best interface designs, if critical acclaim is any measure of success, have resulted from the efforts of one or more creative individuals. When such efforts succeed, they appear to result from fundamental insights into the relationships between particular problems and common solutions (e.g., desktops, spreadsheets). At its best, this approach produces highly creative interfaces which retain salient characteristics of the prototype systems while improving on their efficiency and/or power. In many cases it seems that once a metaphor is found, the details of the interface follow quite naturally. Thus, if one uses the metaphor of a paper ledger for an "electronic" spreadsheet, then the notion of a matrix of "cells" as the interface logically follows. Our analysis is not meant to trivialize such achievements. On the contrary, we are among those who believe that the best human-computer interfaces have thus far resulted from clever ideas and not formal methodologies. However, much of the credit for such successes must go to "flashes of insight" into similarities between apparently different systems and how these similarities might be exploited. Furthermore, this approach seems to work well for some applications (e.g., spreadsheets) and not others (e.g., operating systems). It is simply not a general approach to interface design.

We have thus far dismissed guidelines as unsupported by data and of little value for specific applications, and creative insights as unreliable. What are the alternatives? We contend that the strengths of scientific disciplines, such as cognitive psychology, lie in their use of theories and methodologies. Empirical methodologies have the dual characteristics of *generality*, meaning they can be applied in a wide range of situations, and *validity*, meaning the answers produced by their application are legitimate for the situations to which they are applied. Thus, we propose a methodological approach to interface design.

1.3 Representing Models in the Interface

Although not directly related to any particular psychological theory, the interface design methodology we propose is most compatible with information processing theories that stress the importance of knowledge. In its present form it does not constitute a theory of user psychology or even a comprehensive set of guidelines. However, our methodology does seem suited to a wide range of applications and its basic premises can be clearly and succinctly stated:

1. Users develop conceptual models of devices (systems) through use of or from experience with related systems or tasks.
2. Experienced users' conceptual models can be characterized as memory schemata that can be elicited and represented using an empirical methodology.
3. Interfaces that reflect experienced user knowledge will facilitate learning and increase productivity for users at all levels of expertise.

A major theme of our work is that effective human-computer interaction depends on the communication of the structure and organization of computer systems via the interface. The user needs a model of the computer system that accurately represents its structure. Our research is concerned with the development and evaluation of methods for eliciting, representing, and communicating appropriate models of systems.

1.3.1 User's Models of Systems and Tasks

Models are frequently employed as explanatory constructs in an effort to clarify the process of designing and using devices. Norman (1986) distinguished two types of conceptual models, designer and user models, and contended that the user interface serves as a physical model of the system (i.e., the *system image*). Designers' models consist of those plans that designers attempt to embody in systems they build, whereas users' models are mental representations of the systems formed by users through experience. The user interface, consisting of all aspects of the system that the user comes into contact with -- "either physically, perceptually, or conceptually" (Moran, 1981) -- serves as a physical bridge between these two types of conceptual models. We have been concerned with methods for obtaining and representing users' models of systems and tasks. Our goal is to incorporate such models into the user-system interface (the system image). In short, we propose to base designers' models on users' models.

A number of theories have been offered about the nature of users' mental models of devices. Young (1983) discussed eight types of device models ranging from strong analogy (metaphor) to commonality. Metaphors are perhaps the most common type of model proposed, and they have been exploited in several systems. The frequent use of metaphors in the user-computer interface (e.g., the desktop metaphor) stems from the belief that metaphors provide useful, ready-made models for users of new systems. More importantly for users without previous task knowledge, metaphors may provide new applications with coherent frameworks.

The degree to which interfaces rely on metaphor varies, with some computer-based applications strongly exploiting existing task knowledge. Continuing our previous example, spreadsheet programs are a rather direct form of metaphor. However, this analogy only carries the user so far. Electronic spreadsheets have several capabilities that are only indirectly related to the use of paper spreadsheets. For example, the capabilities of placing formulae in cells, rather than simple values, and moving cells from one location to another are quite indirectly related to the corresponding qualities of paper spreadsheets. Similarly, the use of the typewriter metaphor in word-processor training is of questionable value. Once again, a suitable metaphor exists because the designer's model of the system was based on an existing device (the typewriter). However, beyond communicating to the novice that she can type on the keyboard and letters will appear on the CRT as if it were paper, little is actually gained. This metaphor may also help the user understand why letters disappear when the backspace key is pressed, if the user is familiar with "correcting" typewriters, but may lead to confusion as to why the left cursor key doesn't perform similarly. In short, there is much new knowledge that must be acquired in learning to use a word processor, and an individual who continues to use one like a typewriter will probably fail to see the value in it. To suggest that a new device is like a familiar one because they both share a set of basic characteristics implies that they share other characteristics as well, possibly limitations.

Another difficulty with using metaphors for interface design is that there is currently no methodology for finding them. In fact, it is unlikely that suitable metaphors exist for complex systems (cf. Norman, 1986). When

metaphors are found, they are usually suitable for simple systems or parts of complex systems, and collections of suitable analogies do not suitably represent systems. Thus, to explain that the heart is like a pump may prove useful in conveying important information about how the heart functions. It might also be appropriate to compare the hand to a pair of pliers and the eye to a camera. However, it is highly unlikely that the pump-pliers-camera model conveys anything salient about how the human body works. Thus, metaphors may not be extensible and explanations at one level of abstraction may not hold at other levels (cf. Halasz & Moran, 1982). The methodology we propose provides coherent frameworks for interface design that are inherently extensible by capitalizing on users' task-related knowledge.

1.3.2 Empirical Coherence Models

We contend that users' conceptual models, or system-related knowledge structures, can be characterized as "schemata" in which parts of the system are associated because they are functionally related or because they have other common characteristics (Young, 1983). During the development of these conceptual models, individual facts about the system are either made to fit into the framework provided by the schema or are excluded and forgotten. Knowledge about systems or tasks can be obtained from users, or potential users, with the methodology we propose. Furthermore, this knowledge can be represented, using various scaling techniques, and serves as the basis for interface design. As we will show in this chapter, there are reasonably well-developed techniques for eliciting users' system- or task-related knowledge, as well as for representing such knowledge in the form of coherence models.

As a physical model of the system, the user interface should only represent those aspects of the system that are relevant to users. Other aspects, such as the particular way in which functions are implemented, are not relevant and should not be represented. Ideally, an interface should project a consistent whole that facilitates the accomplishment of tasks using the system. Relationships among aspects of the system can be emphasized by: (a) organizing interface components such that strongly associated components are physically close together, whereas weakly associated components are physically far apart, (b) using

different colors to identify members of groups or categories, (c) graphically "linking" related components together, (d) restricting access to components according to associations thereby creating "modes," or (e) using similar names or shapes (icons) for related components.

In short, the user interface should provide a complete and coherent view of the structure of the system from the user's perspective. System designers should be concerned with developing systems around coherent structures, and interface designers should be concerned with communicating these structures to users. In addition to other relationships, the user interface should convey: (a) the types of data structures embodied in the system or application program, (b) the way in which the data structures depend upon each other, and (c) the data needed and produced by each program. One challenge for research in this area is to identify the appropriate levels of abstraction for conveying these aspects of systems to users and to develop effective means for communicating them. We hypothesize that user interfaces that embody coherence models will facilitate the development of users' mental models and that such models offer advantages over those based on metaphor. In this chapter we focus on the use of scaling techniques to derive coherence models.

2. Proximity Scaling Techniques

Psychological scaling techniques provide several potentially useful methods for measuring and representing users' mental models. These representations can then be used as the basis for coherence models of systems. The techniques can be subdivided into two major classes, those that yield continuous models of mental structure and those that yield discrete models.

The methods to be discussed all rely on some measure of psychological distance (or dissimilarity) among the entities to be scaled. The methods take pairwise distances as input and produce spatial or graph-theoretic representations as output. The pairwise distances can be obtained from several sources such as psychological judgments, frequency of co-occurrence, confusability, correlations, temporal distance, or spatial

distance. Generally there are two goals shared by most psychometric methods: (1) to reduce a large number of pairwise distances to a simpler representation, and (2) to give some insight into the organization underlying the pairwise distances. Let's examine some of these methods in some detail.

2.1 Continuous Models

Continuous, or spatial, models represent entities as points in a multidimensional space. These models usually provide some global information about the entities represented in that the dimensions of the space often correspond to abstract dimensions underlying the variations among the entities. Multidimensional scaling (MDS) is a term used for a variety of specific techniques that represent the scaled entities as points in multidimensional space (Kruskal, 1964, 1977; Shepard, 1962a, 1962b, 1963). Essentially the techniques attempt to find a spatial layout of the points that best corresponds to the given pairwise distances. This generally requires some iterative procedure (e.g., the method of steepest descent) that may lead to inappropriate representations if local minima are encountered. There are methods for protecting against such problems, however. Shepard (1974) is an excellent source of information about potential problems associated with MDS and ways of dealing with them.

The various MDS techniques differ in the exact method used to iterate over various spatial layouts and the criteria used to determine the fit between the spatial representation and the original pairwise distance estimates. Nonmetric MDS attempts to fit the order of the distances estimates, and thus, only requires ordinal measurement. Three-way MDS makes use of several sets of distance estimates (usually from different people) and yields information about the weights attached to each dimension of the space by each set of estimates.

There are several advantages to be gained from MDS analysis. MDS can represent large amounts of data in a form that is more amenable to interpretation. Dimensions of the space can represent important aspects underlying the mental representation of the scaled entities. In some of the applications we will discuss later, MDS provides a means for defining

a spatial layout which is based on users' knowledge of concepts in a domain.

There are also some characteristics of MDS that fail to capture important aspects of the psychological organization of certain domains. If the domain consists of a very heterogeneous collection of concepts, a spatial representation may require several dimensions to capture the many ways in which the concepts vary. But many dimensions can be difficult to interpret and difficult to represent. There are also some problems that naturally follow from spatial representations. Tversky (1977) discusses the observation that the psychological interpretation of the relations between two entities may be asymmetrical. For example, Cuba seems to be more similar to Russia than Russia is to Cuba. Such asymmetries are not compatible with spatial representations, although some efforts have been made to cope with asymmetric data within the MDS framework (Constantine & Gower, 1978; Harshman, Green, Wind, & Lundy, 1982; Krumhansl, 1978).

As MDS attempts to find a spatial representation to account for the pairwise distances given as input, each of the distances plays an equal role in constraining the solution. In one sense, the equal weighting of the distances may be seen as a strength of the method. The entire data set determines the representation. However, when MDS is used to scale psychological similarity or psychological relatedness, equal weighting of the judgments may not be appropriate. It may be more important to capture the relations among the more related concepts. Equal weighting of related and unrelated concepts may lead to distortions in the representation of the related concepts to accommodate those that are not related. Because relations are represented by distance in spatial representations, this problem is difficult to resolve.

2.2 Discrete Models

Several psychometric methods derive from the mathematical theory of graphs, structures consisting of *nodes* and *links* connecting some pairs of nodes (Carre, 1979; Christofides, 1975; Gibbons, 1985; Harary, 1969). A *graph* can be displayed by a diagram in which nodes are shown as points, and links are indicated by lines connecting appropriate pairs of

points. In general, a direction is associated with a link so that one can distinguish its initial and terminal endpoints (nodes). Directed links are called *arcs*. In a symmetric graph, for every arc there is another arc that connects the same pair of nodes in the opposite direction. Such graphs may be referred to as undirected because a pair of arcs can be represented by a single *edge* without direction.

A *path* is a sequence of links such that the initial endpoint of each link in the sequence (except the first) is the same as the terminal endpoint of the preceding link. The initial endpoint of the first link and the terminal endpoint of the last link of a path are, respectively, the initial and terminal endpoints of the path. A path can also be specified by the sequence of nodes which it visits. A path with the same initial and terminal endpoints is a *cycle*. A path (or cycle) is *simple* if it does not traverse any arc of a graph more than once. A path (or cycle) is *elementary* if all the initial endpoints (or all the terminal endpoints) of the arcs in the path are distinct. We will use the terms path and cycle to refer to elementary paths and cycles, respectively.

In applications of graphs, each node represents an entity, and the links represent pairwise relations between entities. Because a set of nodes can be connected by links in many possible ways, a wide variety of structures can be represented by graphs.

Trees are the basis of such psychometric methods as hierarchical cluster analysis (Johnson, 1967), weighted free trees (Cunningham, 1978), and additive similarity trees (Sattath & Tversky, 1977). A *tree* is a graph with no cycles. An undirected connected tree has exactly *n-1* edges where *n* is the number of nodes. In connected trees there is exactly one path connecting two nodes. Links may have weights (distances or costs) associated with them, in which case the graph is known as a *network*. In a network, the length of a path can be computed by summing the weights associated with the links in the path. The geodetic distance between two nodes is the length of the minimum length path connecting the nodes. The minimal spanning sub-tree (Kruskal, 1956) of a network consists of a subset of the links in the network such that the sub-graph is a tree and the sum of the link weights is minimal over the set of all possible subtrees.

Hierarchical cluster analysis provides a set of nested (hierarchical) groupings of the entities which should correspond to meaningful categories. Different hierarchical clustering methods use different definitions of the distance between a category (once formed) and the other entities and categories. The single link method uses the minimum of the distances between the entities in a category and the entities in other categories. The complete link method uses the maximum distance. A less common variation uses the average distance between entities in different categories. As we will see later, there are some close connections between hierarchical clustering and minimal spanning trees. The value of hierarchical cluster analysis lies in its potential for revealing the underlying categorical structure for a set of entities. One problem often encountered stems from the necessity for clusters to be nested, which means that an entity can only belong to certain clusters.

Additive clustering (Shepard & Arabie, 1979) is a method for producing overlapping clusters so that an entity may belong to more than one cluster. The clusters are not necessarily nested so that non-hierarchical characteristics can be seen. Such a representation violates the constraints on a tree structure and, thus, corresponds to a more general graph. The theory underlying additive clustering assumes that the entities have associated sets of features and that the clusters correspond to shared features among the entities. One value of the method lies in its ability to suggest the underlying features.

Other methods derive from the assumption that the appropriate representation for the entities under study is a network (Dearholt, Schvaneveldt, & Durso, 1985; Feger & Bien, 1982; Hutchinson, 1981; Schvaneveldt & Durso, 1981; Schvaneveldt, Durso, & Dearholt, 1985). These methods assume that the pairwise distances given as input represent an underlying network. The output is the network corresponding to the distances.

2.2.1 Pathfinder

The method we have investigated most closely is realized in an algorithm (Pathfinder) which is based on the idea that a link is present in the output network if and only if that link is a minimum length path between the

nodes connected by the link in the (complete) network corresponding to the input distances. Links are given weights corresponding to the pairwise distances given as input, and the length of a path is a function of the weights on links in the path. Generalizing beyond the usual definition of path length as the arithmetic sum of the link weights, Pathfinder defines the length of a path, L(P), consisting of k links to be:

$$L(P)=(w_1^r+w_2^r+...+w_k^r)^{1/r}, \text{ where } w_i \text{ is the } i^{th} \text{ link weight and } r \geq 1.$$

This adaptation of the Minkowski *r*-metric to the measurement of path length allows Pathfinder to accommodate different assumptions about the level of measurement associated with the distance estimates. Ordinal level measurement requires *r* to be infinity in which case L(P) is equivalent to the maximum weight in the path. Ratio-level measurement allows any value of $r \geq 1$, and the particular value selected can be determined by the desired density (number of links) of the network (see below) and/or the interpretability of the solution.

Another parameter used by the Pathfinder algorithm is *q*, the maximum number of links in paths searched in defining the minimum path lengths for all pairs of nodes. The q-parameter is useful from two different perspectives. First, there may be some limit on the number of links that could be meaningfully involved in connections between nodes in a particular domain. This limit can be imposed with *q*. Second, there is a computational advantage that follows from limiting the number of links in paths which can be significant when working with large numbers of nodes. Meaningful values of *q* range from 2 to the number of nodes minus one (*n-1*).

Let us call a network generated from the Pathfinder algorithm a PFNET(*r, q*), indicating that the particular PFNET was generated using a particular value of the *r*-metric and a particular value of *q*. There are systematic inclusion relationships between PFNETs generated with different values of *r* and *q*. We define inclusion of networks as follows: PFNET' is included in PFNET if and only if PFNET' and PFNET have the same nodes and every link in PFNET' is also a link in PFNET. Then PFNET(r_1, q_1) is included in PFNET(r_2, q_2) if and only if $r_1 \geq r_2$ and $q_1 \geq q_2$.

These inclusion properties mean that PFNET(r=∞, q=n−1) is the least dense of the alternative PFNETs, and it is included in all other PFNETs. As r or q decrease, additional links may be added to the PFNET, increasing the density of the network. PFNET(r=∞, q=n−1) is the union of the minimum-cost spanning trees for all possible PFNETs generated from a particular data set. PFNET(r=∞, q=n−1) will be the unique minimum-cost spanning tree when there is such a unique tree. The PFNET(r=∞, q=n−1) allows a construction of the single-link hierarchical clustering solution (minimum method). Essentially this construction involves grouping nodes according to the links between nodes starting with the minimum link weights and proceeding in order through all of the links. It is not possible to derive the PFNET from the hierarchical scaling scheme (HCS) solution. The HCS solution does not provide sufficient information to recover all of the PFNET links.

Networks provide for a natural separation of structural and quantitative information. Structure is determined by the particular set of links in a network and quantitative information is contained in the weights associated with links. Many of our applications of Pathfinder have focused on the structure of PFNETs without regard for the link weights (even though link weights in the form of pairwise distance estimates are used by Pathfinder to construct the PFNET).

PFNETs often have informative structures and substructures such as trees, chains of links, cycles, star structures, and many others. These substructures may have meaningful psychological interpretations (cf. Schvaneveldt, Durso, & Dearholt, 1985). We have also analyzed the link structures of novices and experts (computer programmers and Air Force fighter pilots) to identify systematic differences in their conceptual organizations (Cooke, 1983; Schvaneveldt, Durso, Goldsmith, Breen, Cooke, Tucker, & DeMaio, 1985). In addition we have found that the information contained in the link weights of PFNETs predicts recall orders in free recall (Cooke, Durso, & Schvaneveldt, 1986). The link weights are also used in a measure of node centrality (the degree similarity product) by Dearholt, Schvaneveldt, and Durso (1985).

Some of the properties of PFNETs that contrast with other methods for representing the structure in psychological distance data are:

1. PFNETs can directly model asymmetries in psychological distance because two nodes can be connected by 0, 1, or 2 links. If two links connect the nodes, they can have different weights.

2. PFNETs emphasize pairwise relations between entities. The structure of PFNETs is mainly determined by the closeness of more similar entities and is relatively unaffected by the distance of distant entities. MDS, in contrast, usually weights all pairwise distances equally in determining the spatial representation of the entities. It is possible to vary the weight of each data point in an MDS solution, but this feature is rarely used in practice.

3. PFNETs are capable of representing structures for relatively heterogeneous sets of entities. The PFNETs will capture local structure among related entities and are capable of producing disconnected networks when some of the distance estimates are infinite. When heterogeneous sets of entities are connected, there will often be only a single connection from an entity in one homogeneous subset to an entity in another homogeneous subset.

4. PFNETs are capable of preserving the "nearest neighbor" relations in a set of distance estimates which is not always possible with spatial representations (cf. Tversky & Hutchinson, 1986).

3. Methodological Considerations

One of the first steps in applying our methodology is obtaining the estimates of distance necessary for scaling analysis. All of the techniques we discuss require as input a matrix of distance estimates for all pairs of items in the set. In general, such matrices may be symmetrical (i.e., the distance from *A* to *B* is equal to the distance from *B* to *A*) or asymmetrical (i.e., the distance from *A* to *B* is not necessarily equal to the distance from *B* to *A*) although only Pathfinder makes use of asymmetries.

3.1 Eliciting Concepts

In order to develop effective models of the systems as viewed by users, all important actions, objects, concepts, tasks, etc., must be included in the set of items. For some applications this is a deceptively straightforward process. If one is designing a menu system containing *n* items, then the *n* items themselves will undoubtedly be in the set. However, it is seldom clear what constitutes a sufficient set of concepts for many domains, and even when an obvious set of concepts does exist, applications may require models with multiple levels of abstraction (e.g., higher level categories), and these abstractions may not be readily available. We will discuss one approach to defining category abstractions in describing our work on the UNIX *man* system.

Recently, Cooke (1987) conducted a study in which she compared four techniques (concept listing, interview, step listing, and chapter listing) on the number and types of concepts elicited. She identified seven categories of concepts: (1) explanations, (2) general rules, (3) conditional rules, (4) pure concepts, (5) procedures, (6) facts, and (7) other. The interview and chapter listing techniques elicited the largest number of concepts, although they also produced the largest amount of redundancy. The four "idea" elicitation techniques also varied in the types of knowledge elicited (e.g., the step technique elicited procedures and general rules, whereas the chapter technique elicited concepts). Cooke (1987) listed several factors that should be considered in selecting a particular technique, such as (1) the type of knowledge desired, (2) the detail or complexity desired, (3) the importance of eliciting as much knowledge as possible, (4) the importance of irrelevancy and redundancy, and (5) the man hours available to perform the task. She also concluded that, if possible, several different techniques should be employed in order to insure more complete coverage of the domain.

3.2 Sources of Knowledge

It is generally held that novice and experienced users have different schemas for the domain in which expertise is measured. However, it is not clear how their schemas differ. It may be that experienced and

novice users share a "core" schema, but that experienced users have more elaborate knowledge structures with extensions to the basic core (cf. Rosch & Mervis, 1975). Alternatively, experience may cause even the core schema to be reshaped, making the experienced user's view of the system quite incompatible with that of the novice.

Although there may be occasions when using novice schemas for interface design is desirable, expert schemas are generally preferable, particularly when novices are expected to become more expert. Theoretically, novice schemas are simpler and potentially easier for other novice users to learn, but they are also less stable. This characteristic of novice schemas, compared to those of experts, is exhibited in the high degree of variability obtained from judgments of relatedness. Furthermore, experts' schemas can be simplified if simpler models are required, while still preserving important aspects. This approach also results in models that are extensible and capable of being adapted to changing levels of expertise. Deciding to use experts is not enough, however, because there may be many kinds of expertise within a particular domain (e.g., general knowledge about a broad range of functions versus a great deal of specific knowledge about a narrow range). In any case, it may be useful to compare novice and experienced users' schemas in order to construct models which emphasize these differences, because they may represent novice user misconceptions (Cooke, 1983). It is also important to remember that a design engineer's view of a system is likely to be different than an experienced user's model (Manheimer et al., 1983). A final note: it is important to remember that subjects *will* provide distance estimates if asked to do so, even if they are unfamiliar with the concepts. It is a good idea to determine each subject's familiarity with the concepts to be rated in order to avoid incorporating such "noise" in the proximity matrices.

3.3 Obtaining Estimates of Distance

3.3.1 Instructions

Once a set of items to be rated has been selected, several problems associated with obtaining estimates of distance remain. One of the first decisions to be made concerns the instructions to give raters. General relatedness instructions are commonly used under the assumption that raters will choose the most relevant dimensions on which to compare the items. We have observed, however, that this lack of guidance can lead to difficulties when the data are scaled (e.g., uninterpretable dimensions using MDS), or even later when the results of the scaling analyses are applied to interface design. For example, if raters are asked to judge the relatedness of a set of items that occur together in various combinations (e.g., task sequences) and that also have featural similarities (e.g., similar syntax of usage), then raters may base their judgments on different dimensions for different pairs of items, rather than weighing each pair on the same set of dimensions.

3.3.2 Context

This problem is a consequence of the aforementioned fact that it is not always clear what constitutes the set of items to be judged. In short, because estimates of relatedness may be affected by the context or frame supplied by the rating environment (Tversky, 1977; Murphy & Medin, 1985), the inclusion or omission of particular items may affect judgments of similarity for the other items. Although interface applications may provide a reasonably well-defined set of items, such applications are not exempt from this concern. For example, if only basic commands are included in the set of items to be rated, then higher-level concepts must be inferred from the scaling solutions or obtained from users' judgments (i.e., having subjects name groups or clusters of functions). Including these concepts along with the set of functions to be rated, however, may influence the ratings for the original items. Iterative cycles of rating, scaling, and expanding the set of concepts may be required.

3.4 Data Collection Methods

There are a number of ways of obtaining the estimates that comprise a proximity matrix. We will discuss several common techniques, although others (e.g., repertory grid) are also widely used. None of the methods discussed is ideal in that each has certain advantages and disadvantages that may interact with particular scaling techniques.

3.4.1 Listing

Simply, subjects are asked to generate relevant domain concepts in a relatively unconstrained fashion. The assumption is that subjects will generate new concepts closely associated with those already produced. Because of their "free associational" nature, lists can also be used to obtain estimates of psychological distances, in addition to eliciting domain concepts. Distance can be computed in several ways, although we have most often calculated simple conditional probabilities (i.e., the probability that item *B* follows item *A* across subjects). If concept-elicitation times are obtained (i.e., inter-item intervals), they can be used to identify groups of items. Once groups of concepts are identified, proximity matrices can be constructed using techniques analogous to those for sorting.

Unfortunately, the listing procedure does not guarantee that estimates of distance will be obtained for all concepts from each subject. Furthermore, because different sets of items are generally obtained from each subject, the process of combining individual lists into a composite proximity matrix may be complex. A final strength, at least for Pathfinder analysis, is that asymmetrical distance estimates are a natural consequence of using this technique.

3.4.2 Paired Comparison

In its simplest form, each subject is required to supply an estimate of similarity, or relatedness, for all $n(n-1)/2$ pairs of items (i.e., the combination of n items taken two at a time). This technique offers several advantages. First, all of the necessary distance estimates are obtained

from each of the raters. Furthermore, the individual judgments required are relatively simple and do not require raters to introspect about the overall organization of the set of items. Therefore, it is possible to obtain relatively unbiased estimates of distances between pairs of items or, if the application requires it, to obtain estimates that are biased for specific dimensions (McDonald, Dayton, & McDonald, 1986). For example, general instructions to base judgments on relatedness may be useful for discovering important underlying dimensions, whereas instructions to base judgments on how often items co-occur may be useful for producing matrices that reflect procedural knowledge.

The major problem with this methodology is that it may not be suitable for real-world (large) applications, at least in its standard form. For example, in order to obtain distance estimates for 100 items (a relatively small, real-world set), each subject would have to rate 4,950 pairs. From experience we know that it takes subjects at least 5 seconds to rate a pair of items. Thus, rating this many pairs would take approximately 7 hours, if raters worked diligently at their task. The basic UNIX *man* system of 219 commands, discussed below, would require 39 hours per subject! Such times appear even more impractical when one considers that several valuable "experts" are usually required. This is a particularly troubling dilemma, because large systems are most likely to benefit from organization. One way of overcoming this problem is to have each subject rate only pairs from a subset of the items. Once again, however, judgments can be influenced by the context supplied by the set of items to be rated and raters may use different criteria for their judgments depending on the context. Thus, this procedure may introduce considerable noise into the combined proximity matrix and requires more raters.

A related concern is that the method of paired comparison is relatively inefficient, particularly for Pathfinder analysis. Within the context (or frame) of the items to be rated, most pairs are judged as "not related." Although raters are capable of making fine discriminations among related pairs, they are less able to do so for those that are unrelated. Unrelated items are simply unrelated, not more-or-less so. Because Pathfinder focuses on small distances and generally discounts large ones, fine discriminations between unrelated items would be of little value even if they could be obtained.

As mentioned above, even when judges rate all possible pairs of items, considerable error may be introduced into the ratings if the raters are not familiar with all of the items. It is important to remember that proximity matrices are comprised of distance *estimates*. These estimates (scores) form sampling distributions that may have high variability. Scaling analyses, particularly discrete methods (e.g., hierarchical clustering schemes and Pathfinder), are not equipped to handle variability and tend to treat all differences as meaningful. When combined proximity matrices are formed by averaging individual subjects' ratings, information about such variability is lost. As a consequence, the presence or absence of particular links (Pathfinder) or clusters (hierarchical cluster analysis) may be the result of error in the data, rather than meaningful differences. A potential solution to this problem is to establish confidence intervals for rating data and use these intervals, rather than means, in the scaling computations. Another possibility is to use the scaling solution to guide subsequent data collection, thereby empirically distinguishing true from artifactual results. We will describe this approach in more detail in the discussion of the UNIX *man* study.

3.4.3 Sorting

This method requires judges to sort items into piles based on logical relationships. It is a much more efficient technique than the method of paired comparison, thus particularly suitable for large sets of items. Furthermore, it is not as susceptible to shifts in the criteria (dimensions) used by raters, because all items must be considered as a set. However, it does require subjects to generate an overall organization for the set of items, and this task may not be as simple as making pair-wise judgments.

If the usual procedure is followed (i.e., items are not duplicated), then this method produces data matrices which individually satisfy the ultrametric and triangle inequalities (Miller, 1969). However, violations of these inequalities are common when individual matrices are combined to form a composite matrix. Furthermore, if duplicates are allowed, even individual proximity matrices may contain violations of the *triangle inequality*:

$$D_{jk} \leq D_{ij} + D_{ik}$$

or the more stringent *ultrametric inequality*:

$$D_{jk} \leq \max [D_{ij}, D_{ik}].$$

The consequences of such violations are most clearly seen for hierarchical cluster analysis. As discussed in the section on psychometric scaling techniques, most hierarchical clustering schemes are of two types: the minimum methods (also referred to as "connectedness" or "single linkage" methods) that tend to emphasize smaller distances between clusters, and maximum methods (also referred to as "diameter" and "complete linkage" methods) that tend to suppress them. In the unlikely event that the ultrametric inequality holds without exception, both maximum and minimum hierarchical clustering schemes will produce the same results. However, if there are violations of the ultrametric inequality in the data, then the maximum and minimum methods will produce different solutions. Such differences may be relatively unimportant, although minimum and maximum solutions can be compared in order to assess the extent of the problem (Johnson, 1967). Substantial differences between the minimum and maximum hierarchical clustering solutions may indicate that the systems being represented are not hierarchical in nature.

The combination of the standard sorting procedure and hierarchical cluster analysis is particularly likely to suppress non-hierarchical relationships. Sorting imposes a "hierarchical filter" on the data acquisition process, limiting the extent to which nonhierarchical relationships can be expressed by judges. Because items must be partitioned into nonoverlapping sets, it is impossible for a subject to express the judgment that item x belongs in both groups *A* and *B* (let alone *A*, *B*, and *C*). By combining the judgments of several subjects, such nonhierarchical relationships may emerge (e.g., one subject might place item *x* in pile *A*, whereas another might place item *x* in pile *B*), although the strengths of these relationships may be artificially reduced. Furthermore, this result actually relies on judges having different, or ill-formed, conceptual models. Although it is theoretically possible to overcome the problem of hierarchical filtering by employing duplicates, thus allowing judges to directly express the belief that a particular item belongs in two different

piles, it has been our experience that raters hesitate to use duplicates, probably because of the extra effort involved (see the UNIX sorting study).

3.4.4 Controlled Association

A promising alternative to paired comparison and sorting is the controlled association technique proposed by Miyamoto, Oi, Abe, Katsuya, and Nakayama (1986). In this technique, each of the *n* items is individually presented to the judge whose task it is to select the most associated concepts from the remaining *n-1* items. Although not as efficient as sorting, it is potentially more efficient than paired comparison, because the number of comparisons does not necessarily increase as the square of the number of items. Furthermore, this technique produces considerably more information than sorting and allows judges to express asymmetrical as well as nonhierarchical relationships.

3.4.5 Event Recording

The event record technique is similar to listing, except that users are monitored while interacting with the system under investigation, or the actions of subjects are recorded while performing particular tasks, and the resulting event records, rather than lists, form the basis for distance estimation. As with lists, event records can be easily converted to conditional probability matrices. A major advantage of this technique is that the data are obtained automatically and judges, in the usual sense, are not required. Furthermore, event records contain task information that may not be obtainable from judgments.

The problems inherent in the listing procedure are also problems for the event-record technique. In particular, distance estimates for all pairs of items are not necessarily obtained. This shortcoming, however, may also be considered a strength, because estimates of command usage (i.e., frequency of occurrence) are automatically obtained, helping to define the concept set.

4. Applications of Psychometric Techniques

In the following section we describe several studies that take the general approach of acquiring users' task-related cognitive structures and reflecting them in the design of the user interface. One of the most obvious ways in which basing the system image on users' conceptual models can improve usability is by facilitating the process of locating necessary information. Simply put, the time users spend searching for information can be reduced by putting things where users expect them to be.

4.1 Menu Organization

McDonald, Stone, Liebelt, and Karat (1982) proposed a methodology for designing cognitive aspects of the human-computer interface which is roughly analogous to using anthropometric data (i.e., human body measurements) to design physical aspects of the interface. The method consists of obtaining measures of the cognitive characteristics of potential system users and using these data to organize information at the interface. McDonald et al. reported a study in which subjects learned to access word-processing (WP) terms organized in various ways. They measured performance on a standard paired-associate learning task in which subjects were required to associate a particular response (a sequence of menu selections) with a particular stimulus (the target word-processing function).

During the first phase of this study experienced WP operators were required to rate the similarity of thirty-four WP functions. The resulting proximity data were submitted to hierarchical cluster analysis, and the 16 functions that most strongly clustered in the hierarchical cluster analysis (four functions in each of four clusters) were selected to serve as the terminal nodes in a three-level hierarchy.

During the second phase of the experiment, McDonald et al. had a separate group of 24 subjects learn to access the subset of 16 WP functions selected during the knowledge acquisition phase. On each trial the subjects' task was to access a target word-processing function by selecting the correct menu options from among a series of alternatives. At the

two highest levels of the hierarchy, subjects selected between two alternatives (e.g., *A* or *B* followed by *C* or *D*). At the lowest level of the hierarchy they selected among four alternatives (e.g., *E, F, G,* or *H*). Thus, as an example, the sequence of menu selection necessary to access a particular target (e.g., *Delete*) might be *ACF*. The intermediate nodes in the hierarchy were assigned arbitrarily selected consonants, rather than meaningful category labels, in an effort to assess the effects of organization independent of the goodness of category labels. The 16 WP functions were either assigned to terminal nodes in the hierarchy based on the results of the cluster analysis (categorically) or randomly. Subjects made fewer errors learning to access the WP functions when they were categorically organized compared to the random condition.

In an effort to further clarify the effects of menu organization on user performance, McDonald, Stone, and Liebelt (1983) compared categorical, alphabetical, and random organizations in a visual search task using a single-panel 64 item menu. Five different within-panel organizations were compared, from completely alphabetical to completely random. In three of the organizations the 64 items were organized by category (food, animals, minerals, and cities) into four columns of 16 items. Within these four categories the items were either organized categorically, alphabetically, or randomly. Similarity rating and cluster analysis were used to derive the categorical organizations.

This study also examined the effects that the *type* of target had on menu-selection performance by providing either explicit targets (e.g., "lemon") or single-line definitions (e.g., "a small, oblong, pale-yellow citrus fruit"). Presumably the task of locating targets based on vague definitions is more realistic and provides another measure of the effects of organization. "Real world" users seldom search for explicit targets in menus -- if they know exactly what they are looking for, then they probably know where it is. For example, if a user must search through a menu to find the command for removing a file, then it is unlikely that the name of the command is known (i.e., it might be *Delete, Remove, Erase, Drop,* etc.). If the user knows that the command is *Delete*, then searching the menu probably is not necessary at all. Learning that the name of the command for removing files is *Delete* usually involves using the system. Because there is ample evidence that the effects of menu organization disappear

with practice (Card, 1982), using the explicit-target search task is a bit like having experienced users locate familiar items in rearranged menus. While this technique may be useful for assessing the degree to which a particular menu organization corresponds to the user's conceptual organization (a reasonable objective on occasion), it may not provide a valid measure of the effects of organization under realistic conditions.

The results of the McDonald et al. (1983) experiment showed a clear advantage for categorical menus over pure alphabetical and random organizations, especially when subjects were uncertain about the target. The advantage of the categorical-categorical organization (i.e., items clustered categorically within categorically organized columns) was especially pronounced during the first block of trials when subjects were shown definitions rather than explicit targets. As predicted, the effect of organization virtually disappeared by the last block of trials (block five, 64 trials/block), replicating the earlier results of Card (1982).

It is tempting to conclude from the preceding that interface organization only affects user performance during learning. In this regard, McDonald et al. (1983) argued that the structure of the interface may also influence the development of users' conceptual models and, consequently, there may be persistent effects of organization when users are required to perform complex tasks. Furthermore, there is some evidence for persistent effects of organization on relatively simple tasks, as shown by the following study.

McDonald, Dayton, and McDonald (1986) conducted a study in which the keys (menu items) on a simulated fast-food cash register keyboard were either organized according to MDS solutions or customized by individual users. During the scaling phase of this experiment, two groups of twelve subjects rated all 276 pairs of 24 fast-food items. One group rated the *similarity* of the pairs whereas the second group rated the pairs in terms of how likely they were to be ordered as part of a single customer's meal (judged *frequency of co-occurrence*). In the second phase of the experiment three groups of eight subjects used different keyboard layouts to enter one- to four-item food orders.

The Similar Keyboard (SK) group used a menu layout based on a nonmetric MDS analysis of the similarity ratings, whereas the Co-occurrence

Keyboard (CK) group used a keyboard layout based on an identical MDS analysis performed on the frequency of co-occurrence ratings. Because the purpose of the MDS analyses was to provide guidance in laying out the keyboards, two-dimensional solutions were used, and no effort was made to determine the optimal dimensionality for the MDS analyses. Thus, the average stress values (Kruskal's stress formula 1) for both solutions remained relatively high, although approximately equal (SK = .336, CK = .358). The keyboards were produced by mapping keys onto points in the MDS solution and reducing the inter-key spacing until all of the keys fit within the size limitations of the keyboard. Because of the nature of this procedure, only ordinal properties of the MDS solutions were retained[1]. Subjects in the Personalized Keyboard (PK) group were told to place the cash-register keys anywhere they wished, with the stipulation that they allow adequate spacing between them.

During the validation phase of this experiment, each subject participated in three, one hour sessions on consecutive days during which they were required to enter 24 single-item orders and 216 multiple-item food orders (72 each of the two-, three-, and four-item orders). Multiple-item food orders were classified as either complementary (e.g., *Hamburger, French Fries, Coke*), similar (e.g., *Cola, Root Beer, Iced Tea*), or odd (e.g., *Eggs, Sundae, Iced Tea*). The subjects' task was to select all of the items in a given food order by touching the corresponding keys on the keyboard. Each trial was terminated when the subject touched the *Total* key.

The results showed strong, early effects of organization which, as in the previous menu-selection studies, could largely be attributed to visual search. Subjects using the SK keyboard performed best with similar orders whereas subjects using the CK keyboard performed best with complementary orders. Contrary to expectations, subjects in the personalized keyboard group had the poorest performance on the first day. For purposes of this discussion, the most important finding was that the pattern of results for multiple-item orders in the SK and CK conditions remained essentially the same after practice. After the subjects learned

[1]We are currently investigating the use of a simulated annealing procedure for keyboard layout

the locations of the keys on the keyboards, thus eliminating the need for visual search, the effects of organization persisted. This persistence was probably due to the physical movement required in selecting items in multiple-item orders. Thus, visual search time was replaced by inter-item movement time as the source of the organization effect.

As mentioned above, subjects using personalized keyboards took more time to enter food orders than did subjects in the SK and CK conditions, particularly on the first day. Interestingly, by the third day of practice their performance approached that of subjects in the SK group, although subjects in the CK group remained superior. When the PK keyboards were compared with the SK and CK layouts, it became evident that the personalized keyboards were variations of the SK scheme. This observation was supported by correlational analyses of inter-key distances; seven of the eight subjects' keyboards in the PK group resulted in significant positive correlations with the SK layout, whereas only two keyboards resulted in significant positive correlations with the CK layout (one subject's keyboard was positively correlated with both the SK and CK layouts). Furthermore, when asked to choose between the keyboard layout based on judged similarity or the layout based on frequency of co-occurrence, people generally preferred the similar keyboard. It is important to note, however, that most fast-food orders are complementary, rather than similar or odd, and that a keyboard based on frequency of co-occurrence should be superior to one based on similarity and also better than keyboards based on individual preference.

In another application, Tullis (1985) described the development of a menu-based interface for an existing command-based operating system. He began by obtaining psychological distance estimates among existing system functions from individuals familiar with the command-based operating system. He had subjects sort function-labeled index cards into "logically related" groups. He then combined these data into a composite proximity matrix and then obtained a representation of the users' model of the system using hierarchical cluster analysis (maximum method).

Tullis was primarily interested in subjects' ability to access functions in "broad" vs. "deep" menu hierarchies. He selected two different cut-off levels in the same hierarchical cluster analysis for this comparison. The

deep version of the menu-access system was restricted to 15 options per menu panel, arranged in a single column. In the broad version, the number of options on individual panels was extended by using two or three columns, up to a maximum of 45 options per panel. Subjects took essentially the same amount of time to accomplish tasks with both versions of the menu, but they made fewer errors with the broad hierarchy. He concluded that, in general, the better menu hierarchy is the one that requires the fewest steps (i.e., the "broader" menu).

The broad and deep organizations were both based on the results of the same hierarchical cluster analysis. Thus, the small effects reported for the depth/breadth manipulation should not be surprising, because all of the clusters corresponded to *potentially* meaningful categories. Earlier research into the depth versus breadth issue (Miller, 1981; Snowberry, Parkinson, & Sisson, 1983) largely ignored the question of category goodness by making the dubious assumption that a set of items can be assigned to the terminal nodes of different hierarchies with arbitrarily many intermediate nodes (category labels). However, a similar problem occurs when one assumes that all of the clusters in a hierarchical cluster analysis of composite data are meaningful.

Roske-Hofstrand and Paap (1986) showed that pilots could access the information pages of a Control-Display Unit (CDU) more efficiently when the menu structure was based on a Pathfinder network derived from judgments of pilots compared to a menu structure developed by a design team. The CDU is a menu-based system used by pilots to facilitate the planning and monitoring of flight activities. This study used the method of paired comparison to obtain similarity ratings for a subset of 34 of the CDU panels from four experienced pilots. The resulting proximity matrix was submitted to Pathfinder analysis to obtain a PFNET($r=\infty$, $q=n-1$) for ordinal data.

The Pathfinder network was used to determine the menu options that appeared at the bottom of each information panel. Two sets of "dominating nodes" (sets with the minimum number of nodes necessary to reach every other node in the network by traversing a single link) were selected to appear on the primary index page. These dominating-node sets differed in their level of redundancy, i.e., the number of extra links

from the index pages to the lower-level pages. In the more redundant network, lower-level pages (nodes) could more often be reached from several of the index pages. Two other organizations were compared, a hierarchy and an organization described in the original specifications for the CDU.

The different organizations in the Roske-Hofstrand and Paap (1986) study can be thought of as consisting of three different kinds of panels. The primary index page, listing the "special" index pages corresponding to the dominating nodes, appeared at the highest level of each organization. The dominating-node panels, in turn, listed the lowest level panels that could be reached directly and, in the nonhierarchical organizations, other dominating node panels as well. The four organizations were compared in an experiment in which sixteen pilots were asked questions that could only be answered by accessing particular CDU panels. Four pilots were assigned to work with each of the four menu prototypes: high redundancy (22 extra links), low redundancy (10 extra links), hierarchical (0 extra links), and the organization based on the specifications of the original CDU design team. The results showed that the high-redundancy group had significantly faster task times than the other groups, supporting the use of cognitive networks for organizing menu panels. It is interesting to note that the advantage for the high-redundancy organization persisted with practice and that subjects in this group still performed faster than subjects in the other conditions during the last block of trials.

The studies discussed in this section have demonstrated that menus organized according to users' empirically-derived cognitive structures are superior to other alternatives (e.g., alphabetical, random, and subjective organizations). The general methodology used in these studies involves (1) obtaining judgments of relatedness for all pairs of menu items, (2) scaling the proximity matrix in order to obtain a representation of the set of items, and (3) mapping the obtained representation onto menu layout (logical or physical). When simple tasks requiring visual search have been used, the effects of organization generally disappear with practice, although persistent effects of organization have been obtained for more complex tasks, and other effects resulting from the development of conceptual models have not been assessed. In the following study we discuss an application in which facilitating the development of effective user models is of primary importance.

4.2 On-Line Documentation

The objective of our work on documentation is to provide structural descriptions of an existing command-based system (UNIX). These structures can be used to provide the kind of context-sensitive help facility typically available on menu-based systems. In addition, structural descriptions of systems might be made available to writers who are in the processes of writing system documentation. Such information should be useful for organizing information within as well as between information panels, and for insuring that important relationships between system elements are documented. Another goal is to allow users to efficiently develop accurate conceptual models of systems by interacting with the documentation.

As a component of the user-system interface, "help" poses a particularly difficult challenge for designers. Using help and documentation is, almost by definition, a problem solving activity. Users are confronted with the tasks of figuring out what they need to know, locating the required information (or discovering that it is not available), and inferring answers to their questions from the information provided. Some common techniques for providing help, such as context sensitivity, already rely on there being a close correspondence between the system and its documentation by assuming that when users request "help" they need help with whatever it is they are currently doing. However, context sensitive help has only been developed for moded systems, typically those with menu-based interfaces, and we propose to extend its application to modeless systems as well. Furthermore, context-sensitive help systems primarily aid users in selecting appropriate entry points into documentation and may be of little or no assistance when users commit a series of errors and are being led down a blind alley. In other words, the assumption underlying context-sensitive help systems is occasionally false, in that the error that eventually causes a user to request help may have occurred earlier in the transaction. Thus, our proposed organizing scheme will enhance the benefits provided by context-sensitive help by guiding users to relevant information after they enter the documentation system as well.

We are particularly interested in providing context sensitivity for command-based systems by developing network models, or state transition diagrams, and tracking user interactions (cf. Jackson & Lefrere, 1984). We hypothesize that when a user is in a particular state (i.e., location in the system network) and requests help, it should be possible to infer his or her goal and provide assistance in the form of suggested paths among help panels. Furthermore, a graphic representation of the system network (e.g., a map with a "you are here" marker) might encourage goal-directed, rather than procedural, behavior by allowing users to indicate where they want to go rather than how to get there (Newell & Simon, 1972). Allowing users to select goals, rather than the procedures necessary to accomplish them, should decrease the complexity of the user's task.

4.2.1 Superman

Sachania (1985) identified the need to improve access to information contained in the UNIX on-line manual system (*man*). He applied Pathfinder to the organization of this collection of information panels. Sachania converted the co-occurrence data inherent in the *man* system into a distance matrix. By summing the number of co-occurrences of commands in the "see also" references on each panel he was able to construct a matrix suitable for Pathfinder analysis. The resulting network solution [PFNET ($r=\infty$, $q=n-1$)] for over 200 UNIX commands formed the basis for his *Superman* system which guides users along paths of related topics. As a consequence of developing his system, Sachania discovered that 28 panels in the *man* system were not referenced by any others, a fact almost certain to be of interest to the designers of the system!

Although not empirically validated, *Superman* appears to be a step in the right direction. There are, however, a number of problems with this particular approach. The most questionable aspect of Sachania's methodology lies in his short-cut technique for constructing the distance matrix. Assuming that the algorithm he used for tallying "see also" references is optimal, the resulting network is, at best, a reflection of the impressions of a small set of documentation designers, rather than users. It is unclear what level of experience the developers of the UNIX *man* system had or what method they used in organizing the panels. The *man* system has evolved, along with UNIX, over a period of years and the backgrounds and qualifications of the developers undoubtedly vary, producing obvious inconsistencies. The 28 unreferenced panels are evidence that the procedure was flawed.

Superman only addresses a small set of the problems users have in locating information. If a user needs information about a command, and already knows the name of the command, then the unaugmented *man* system works reasonably well. If a user needs information about a command but does not know its name, then *Superman* may be of some assistance in locating the desired information (if the user knows the name of a related command). *Superman* is also useful for browsing, when the user is not searching for any particular information but simply exploring the system. However, indexing aids of this kind will most commonly be used to guide users to information via categorical entry points (e.g., editing, programming, filing, etc.). Thus, assistance is needed at a categorical level. A user should be able to request information about "printing" or "communication" functions and be guided to the appropriate subset of commands.

5. Superman II

Our approach to the UNIX interactive documentation guide (*Superman II*) is to develop a system that (1) is based on empirically derived representations of experienced users' conceptual models, (2) has several perspectives (e.g., functional and procedural), (3) has multiple levels of abstraction within each perspective, and (4) provides users who are familiar with other operating systems (e.g., DOS) a "bridge" for transferring their knowledge to UNIX. In order to achieve these objectives, we have employed both sorting and event recording techniques. We have also developed a procedure for eliciting the additional knowledge necessary to identify functional categories and task sequences from the basic representations. Furthermore, we are exploring ways of combining the separate analyses obtained from sorting and event recording to form a more complete coherence model of experienced UNIX users. We will then proceed to conduct a series of studies which will focus on the UNIX *man* system, *Superman*, and *Superman II*.

5.1 The Functional Perspective

Fifteen experienced UNIX users from New Mexico State University voluntarily participated in this phase of the project. A questionnaire administered to the subjects after participation characterized their varying degrees of UNIX experience. Seven were classified as expert or intermediate-to-expert users whereas eight were considered intermediate users (see Table 5-1).

Table 5-1: A summary of the UNIX sorting data showing the number of commands (CMDS) sorted, the number of piles, and the number of duplicates (extra cards) used by each subject

S#	EXPERTISE	PILES	CARDS	CMDS	EXTRA CARDS	PILE
1	I-E	34	139	139	0	4.09
2	I	36	141	137	4	3.92
3	I-E	15	138	138	0	9.20
4	I	54	224	204	20	4.15
5	I	32	116	116	0	3.63
6	I	47	140	135	5	2.98
7	I	34	149	134	15	4.38
8	I	39	113	113	0	2.90
9	E	15	154	154	0	10.27
10	I	22	94	94	0	4.27
11	E	18	220	219	1	12.22
12	E	9	199	199	0	22.11
13	E	32	125	125	0	3.91
14	I	41	190	160	30	4.63
15	I-E	31	125	125	0	4.03
	AVERAGE	30.60	151.13			6.45
	SD	12.64	39.40			5.18

A 5" by 7" index card was prepared for each of the 219 functions documented as part of the Berkeley 4.2 version of the UNIX *man* system running on SUN microsystems minicomputers. The name of each command was printed in large, boldface type on the face of the index card. The name of the command and a short command definition (up to four lines) were printed on the back.

Subjects participated individually in a single session of approximately one hour (range = 20 minutes to 2 hours). The sorting procedure took

place in a large conference room with four adjoining tables. A file box containing the 219 printed cards, along with some blank index cards, was placed in front of the subject. Subjects were instructed to use functional criteria to sort commands which they knew how to use. After the initial sort, they sorted additional familiar commands by referring to the definitions on the back of the cards. Subjects were told not to sort any unfamiliar commands. To eliminate the "hierarchical filtering" associated with sorting, subjects were encouraged to use duplicate cards when they felt a particular command belonged in more than one pile.

A co-occurrence matrix was created for each subject by assigning values to all pairs of cards based on their membership in piles. If two cards were in the same pile they were assigned a distance of 0 in the co-occurrence matrix. If two cards were in different piles, they were assigned a distance of 1. A combined co-occurrence matrix was created by summing the individual matrices. This resulted in a "distance" matrix with values ranging from 0 (all fifteen raters placed the two commands in the same pile) to 15 (no rater placed the two commands in the same pile).

The number of unique commands sorted by individuals ranged from 94 to 219 (Table 5-1). All commands were sorted by at least one individual, but only 44 of the 219 commands were sorted by all 15 raters. Of these 44, 37 were identified as "core" UNIX commands by eliminating those commands not sorted in a pile with one or more of the other 43 by at least half of the raters. A hierarchical cluster analysis of these 37 commands is shown in Figure 5-1. Separate hierarchical cluster (using both the Maximum and Minimum methods) and Pathfinder analyses were performed on the "raw" co-occurrence matrix and a conditional probability matrix in order to determine the most appropriate representation. The conditional probability matrix was constructed by dividing each distance estimate in the co-occurrence matrix by the smaller of the two frequencies for that pair (i.e., the probability of the more frequent command given the less frequent command).

For example, the *cat* command (catenate one file to another file, often the display) was sorted by all fifteen raters. The *comm* command (merge two files and display), on the other hand, was sorted by only seven

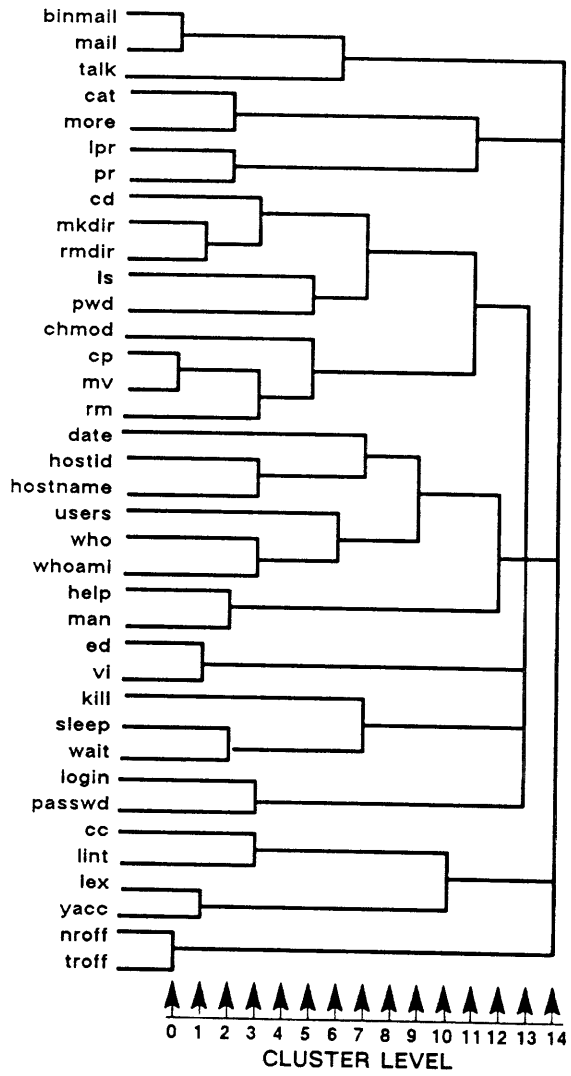

Figure 5-1: Hierarchical cluster analysis (Maximum Method) of the 37 'core' UNIX commands from the sorting study

raters. Of the seven raters who sorted both commands, only 3 put *cat* and *comm* in the same pile. Thus, in the co-occurrence matrix this pair had a distance of 4, whereas in the conditional probability matrix it was assigned a distance of .57 (4/7), equivalent to approximately 8.6 in

co-occurrence units. As can be seen from this example, computing conditional probabilities in this way tends to increase the relative distances between pairs in which one or both items are not judged by all of the raters.

The preliminary Pathfinder analyses (all 219 nodes connected) resulted in a network with 655 links for the "raw" co-occurrence matrix and 417 links for the conditional probability matrix, PFNET(r=∞, q=n–1). The network derived from the conditional probability matrix was not only less complex, but also seemed to capture a more coherent organization as judged by experienced users. Accordingly, this network was used in the analyses to follow.

In *Superman II* we must graphically represent "subnets" (portions of a Pathfinder network) that correspond to high-level concepts. In Figure 5-2 we show one approach to representing subnets that involves the combined use of hierarchical cluster and MDS analyses. First, we obtained clusters using a minimum hierarchical cluster analysis (Johnson, 1967). We then subjected the appropriate subsets of distance estimates for each cluster to a 2-dimensional, nonmetric MDS. Finally, we used the MDS coordinates to guide node (command) layout and added the links specified by the Pathfinder solution. Comparing the hierarchical and network representations of these clusters is informative. In Figure 5-2, for example, not only are the two clusters consisting of directory-level and file-level commands evident in the PFNET, but the "bridge" between these two clusters, from *rmdir* (remove directory) to *rm* (remove file), can be seen as well. It is tempting to speculate that this connection represents more than the simple fact that both of these commands remove things. Most UNIX users will recognize that these commands have a more fundamental relationship as well; in order to remove a directory all of the files in the directory must be removed first.

A similar comparison of hierarchical cluster and Pathfinder analyses is shown in Figure 5-3. In this example we selected the subset of commands used in programming (e.g., compilers, interpreters, etc.). Rather than relying on an MDS analysis to help in laying out the network, we used the constraints inherent in the Pathfinder solution. This procedure is somewhat more subjective than the MDS approach, although many of

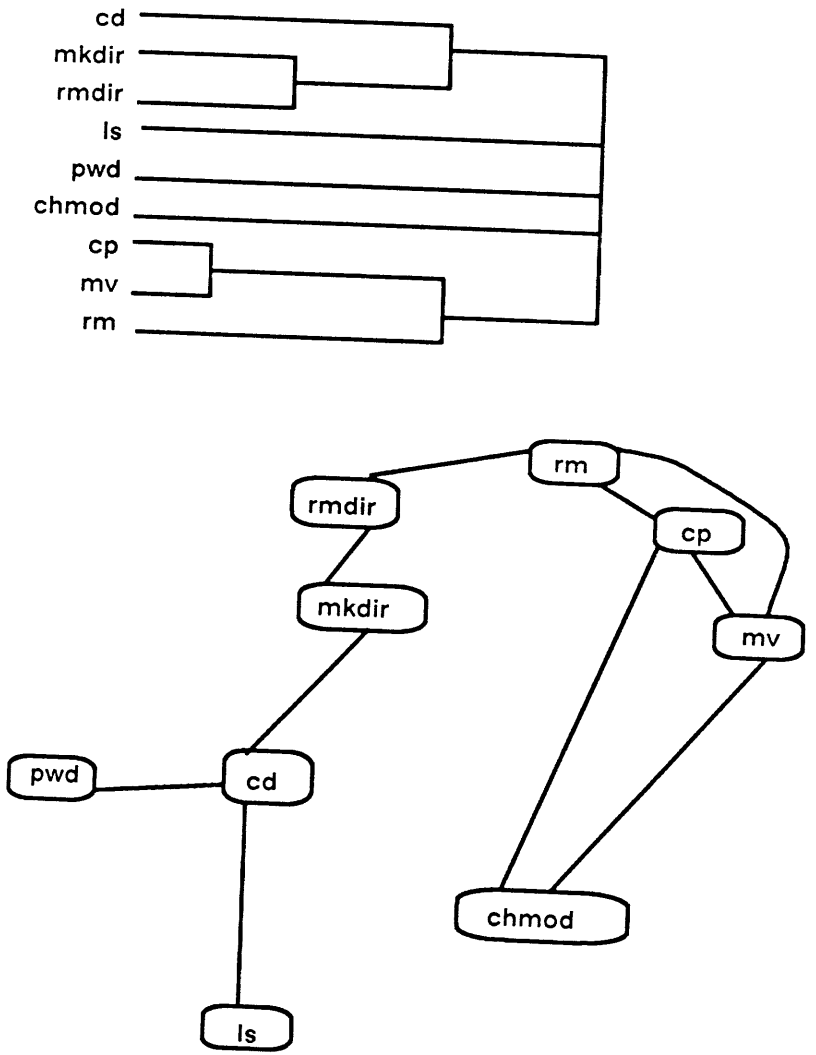

Figure 5-2: A comparison of hierarchical clustering (Minimum Method) and network analysis [PFNET(r=∞, q=n−1)] for UNIX directory commands

the layout rules, such as minimizing link crossings, are relatively unbiased. For those familiar with programming, but unfamiliar with UNIX, the central role of the *C* compiler (*cc*) may seem a bit strange. An

experienced programmer might reasonably expect the assembler (*as*) to occupy such a role, but not a particular compiler. However, those familiar with UNIX will recognize the central importance of *C* to UNIX (e.g., UNIX is written in *C*). Other details in the Pathfinder representation can be observed in this comparison as well, such as the connections between the *C* utilities (*cb, cpp,* and *lint*) and the *C* compiler, and the particular way in which the Pascal functions (*pc, pi, pix,* and *px*) are linked. In both Figures 5-2 and 5-3, the PFNET provides more information than the corresponding hierarchical representation. Indeed, the minimum hierarchical cluster solution can be derived from the PFNET, but not vice versa.

The complete functional perspective requires not only valid categories, but appropriate category labels as well. As we have already discussed, judgments of relatedness contain error and the attempt to identify subnets, described above, is subject to the criticism that some of the clusters correspond to conceptual categories, whereas others are artifacts of the procedures employed. One approach to resolving such questions is to obtain more data. Therefore, we had four of the most experienced UNIX users from the original study judge the goodness of clusters obtained using hierarchical cluster analysis and name the better clusters.

For this phase of the procedure we selected the subset of 152 UNIX commands that were known by at least half of the 15 experienced users. We then performed a hierarchical cluster analyses on the 152 commands using the minimum method. The minimum method was selected because there is a direct correspondence between Pathfinder with r=∞ and q=n−1 that guarantees our ability to map the resulting clusters onto subnets, and the minimum method tends to produce fewer distinct clusters, reducing the amount of rating required. The hierarchical cluster analyses produced a total of 83 distinct clusters, not including the highest-level cluster in which all 152 commands were grouped. Each judge was shown all 83 clusters in a random order and asked to rate the goodness of each cluster on a five point scale, ranging from 1 = Very Bad to 5 = Very Good, and to provide appropriate names for all but the Very Bad clusters. Judges were also allowed to enter a 0 if they did not know one or more of the commands in the cluster. The ratings of the judges were averaged for each of the 83 clusters. There was a general

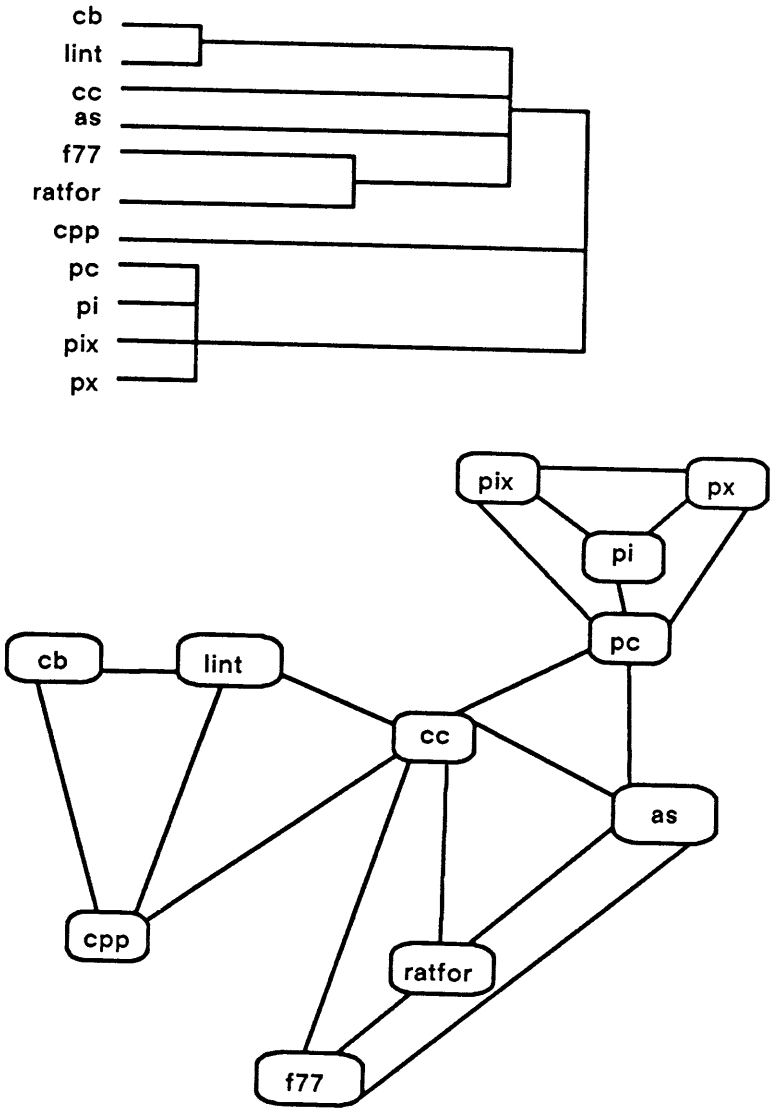

Figure 5-3: A comparison of hierarchical clustering (Minimum Method) and network analysis [PFNET(r=∞, q=n−1)] for UNIX programming functions

tendency for judgments of cluster goodness to decrease as cluster size increased (r = -.50). However, the ratings for clusters ranging in size from two to five commands averaged 4 or better (Good to Very Good).

In the next step, we used the ratings and names supplied by the judges to eliminate artifactual clusters. As an example, Figure 5-4 shows the cluster corresponding to communications functions, with the average ratings of the original nine clusters shown on the top and the reduced version with four categories shown on the bottom. Reduction was accomplished by comparing the names given by the judges for each of the smallest clusters with those given for the clusters above them in the hierarchy, and collapsing smaller clusters into larger ones when the names were the same. Although there is a certain amount of subjectivity involved in deciding that two sets of names are the same, in this case this procedure resulted in considerable simplification and produced very sensible categories. However, some anomalies are also evident, such as the *biff* command not being included in the *Electronic Mail* category, as one might expect, but only the more general *Communication* category. We are currently exploring the possibility of having judges indicate clusters, and name them, by referring directly to network representations, thereby eliminating such artifacts of the hierarchical clustering process.

5.2 The Procedural Perspective

Our approach to developing a procedural perspective was to obtain a number of protocols from nine of the UNIX users who participated in the sorting study. The technique we used required no special effort on the part of the participants. After obtaining permission, we modified users' *.login* files to increase the size of their *history* records so that all of the events that occurred during a session were automatically captured and, at the end of each session, mailed to us. A summary of the event-record data is shown in Table 5-2.

A total of 41,372 commands were obtained. Of these, approximately 61% came from section 1 of the *man* system. The number of unique commands used by individuals ranged from 42 to 133. Interestingly, as shown in Table 5-3, the top ten commands accounted for 66% of all commands issued -- the top three commands accounted for 32%! While these statistics would undoubtedly change somewhat from one group of users to another (e.g., *rsh* is only used when several machines are networked together), they are probably typical of commands usage.

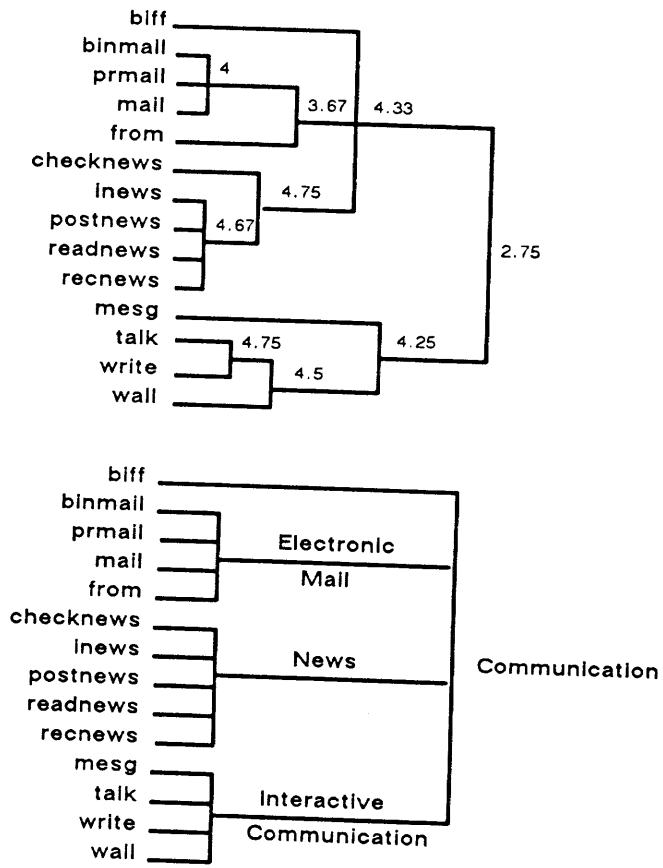

Figure 5-4: A hierarchical cluster analysis (Minimum Method) for Unix communication commands with average 'goodness' ratings (top) and after reduction based on naming (bottom)

The event records from all of the users were combined into a composite frequency of co-occurrence matrix by incrementing the appropriate cells each time a particular command followed another in an event record. Note that this is a relatively crude method for encoding the information in these data because only two-step sequences are considered. The data

Table 5-2: A summary of the UNIX event record data showing the number of commands recorded for each user and the number and percentage of these commands that come from Section 1 of the *man* system

USER	ALL CMDS UNIQUE	TOTAL	SECTION 1 CMDS UNIQUE	TOTAL	PERCENT
1	58	2,124	39	1,197	56%
2	72	1,139	41	685	60%
3	47	1,370	32	814	59%
4	97	2,786	57	1,728	62%
5	42	863	26	592	69%
6	84	9,499	51	4,994	53%
7	122	15,894	75	11,225	71%
8	49	2,196	27	771	35%
9	133	5,501	84	3,178	58%
TOTAL		41,372		25,184	
AVERAGE	78.22	4,597	48.00	2,798	58%

Table 5-3: The ten most frequently used commands from the UNIX event record study

RANK	CMD	FREQ	PERCENT
1	ls	4,719	11.41%
2	cd	4,420	10.68%
3	more	3,961	9.57%
4	|(pipe)	3,372	8.15%
5	mail	2,346	5.67%
6	emacs	2,078	5.02%
7	fg	2,060	4.98%
8	pwd	1,781	4.30%
9	rsh	1,588	3.84%
10	rm	815	1.97%
	TOTAL	27,140	
	ALL CMDS	41,372	65.60%

were then converted to conditional probabilities such that large conditional probabilities produced short distances in the matrix whereas small conditional probabilities produced large distances. This proximity matrix served as the basis for our subsequent analyses.

After converting the frequency of co-occurrence matrix into a conditional probability matrix, we submitted a subset of 49 commands to Pathfinder analysis ($r=\infty, q=n-1$). The network of commands selected consisted of those that occurred in the event records of at least half of the users. The purpose of this analysis was to identify task *sequences*. Figure 5-5 shows a portion of this network that includes all commands within two links (arcs) of *kill*. As an example of the type of information contained in the network, and its potential utility, suppose several programs were executing "simultaneously." Further, suppose that one of these programs was a particularly time-consuming analysis that had to be terminated for some reason (e.g., to reduce processor load). This is one of the few cases where the name of the appropriate UNIX command is fairly easy to remember (i.e., *kill*). However, in order to *kill* a program, you have to know its job number. The name of the command that returns job numbers is not so easy to remember. From the network in Figure 5-5 it is apparent that only one command, ps, frequently precedes *kill* (i.e., there is only one arc leading to *kill*). As you might anticipate from the fact that we are using this example, *ps* (program status) is the command that returns job numbers, along with some other useful information.

5.3 Future Directions

We are currently applying our techniques to other operating systems in an effort to establish a common level of abstraction. We hypothesize that operating systems quite similar to UNIX, such as DOS for the PC, as well as those that appear different, such as the graphic-based Mac system, can be described in terms of a prototypical operating-system model. The development of this model will facilitate the transfer of knowledge from one operating system to any other. Thus, in addition to simply establishing that two commands serve similar functions (e.g., *dir* in DOS and *ls* in UNIX), abstract categorical and task-level relationships can also be identified.

We plan to conduct a series of controlled experiments in which the three documentation systems will be compared (*man, Superman,* and *Superman II*). In the first of these studies, novice users of the UNIX system will be required to locate both specific information (information

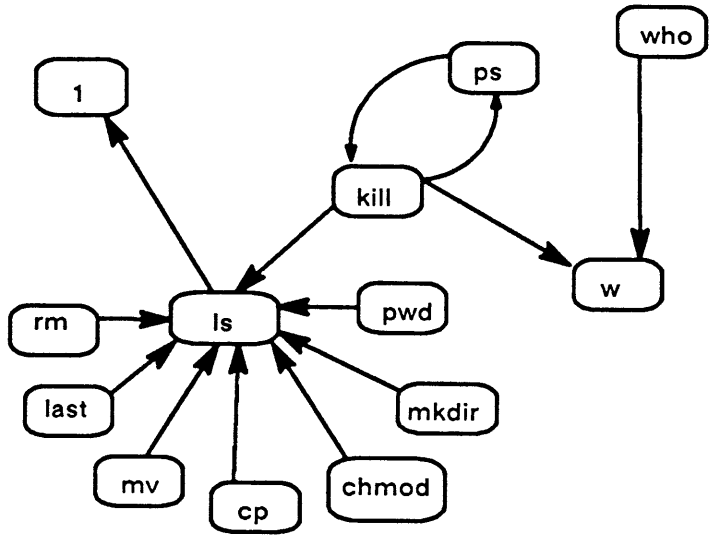

Figure 5-6: A Pathfinder analysis [PFNET($r=\infty$, $q=n-1$)] of UNIX event-record data, including all commands within two links (arc) of *kill*

contained on single panels) as well as information which will require them to integrate information from several panels. In addition, once subjects are familiar with the documentation system, they will be given tasks which require them to locate information previously not retrieved and, finally, to provide an overall description of the UNIX system including all of the commands they can recall and their interrelations. Success or failure in completing tasks and task completion times will serve as the general dependent measures, along with measures which will directly assess the development of cognitive models.

6. Conclusions

In this paper we have presented a methodology for interface design. The basic methodology consists of (1) selecting a set of items to be

analyzed (e.g., concepts, actions, objects), (2) obtaining estimates of distance between items in the set, (3) submitting the resulting proximity matrix to an appropriate scaling analysis (e.g., MDS, cluster analysis, Pathfinder), and (4) using the scaling solution as a "blueprint" for the system image. All of the scaling techniques suggested in this paper have been widely studied and are appropriate for different applications. Network scaling analyses seem particularly flexible and suitable for providing organization to complex systems.

Although a number of studies have employed the methodology we propose, some problems remain. More detailed guidance must be supplied if the methodology is to become widely applicable. In its present state, our methodology allows designers to acquire user knowledge to organize interface elements. Such user's models are potentially more useful, however, and may eventually form the basis for a comprehensive theory of user psychology.

Our main objective is to develop a theory of interface design. It is, of course, not possible to validate a theory. At best, one theory emerges as the most useful on a number of dimensions in comparison to competing theories. The best interface design theory will provide a useful framework for explaining what is known about human-computer interaction, make interesting predictions about what is not known, provide specific mechanisms for translating the theory's principles into design decisions, and have several successful applications to its credit.

We have discussed at some length the most obvious application of our methodology, menu organization, and the benefits one might expect. We are pursuing answers to the question of what other effects might be expected, specifically those that arise from the development of conceptual models. In the process, we are constructing models of users and tasks that can be applied in a variety of ways. By extending the basic knowledge acquisition methodology, it is also possible to use these results to construct data-bases suitable for expert-system development (Butler & Corter, in press; Cooke & McDonald, 1986).

Acknowledgements

The content and presentation of this chapter have benefited substantially from reviews by authors of other chapters of this volume, and from comments by Nancy Cooke and Ken Paap.

References

Butler, K. A., & Corter, J. E. (in press). Use of psychometric tools for knowledge acquisition: A case study. In W. Gale (Ed.), *Artificial intelligence and statistics*. Reading, Mass.: Addison-Wesley.

Card, S. K. (1982). User perceptual mechanisms in the search of computer command menus. *Proceedings of Human Factors in Computer Systems, CHI'82*, 190-196.

Carre, B. (1979). *Graphs and networks*. Oxford: Clarendon Press.

Christofides, N. (1975). *Graph theory: An algorithmic approach*. New York: Academic Press.

Collins, A. M., & Loftus, E. F. (1975). A spreading activation theory of semantic processing. *Psychological Review, 82*, 407-428.

Constantine, A. G., & Gower, J. C. (1978). Graphical representation of asymmetric matrices. *Applied Statistics, 27*, 297-304.

Cooke, N. M. (1983). *Memory structures of expert and novice computer programmers: Recall order vs. similarity ratings*. Unpublished M. A. thesis, New Mexico State University.

Cooke, N. M. (1987). *The elicitation of units of knowledge and relations: Enhancing empirically-derived semantic networks*. Unpublished Ph.D. dissertation, New Mexico State University.

Cooke, N. M., Durso, F. T., & Schvaneveldt, R. W. (1986). Recall and measures of memory organization. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 12*, 538-549.

Cooke, N. M., & McDonald, J. E. (1986). A formal methodology for acquiring and representing expert knowledge. *Proceedings of the IEEE, 74, 10*, 1422-1430.

Cunningham, J. P. (1978). Free trees and bidirectional trees as representations of psychological distance. *Journal of Mathematical Psychology, 17*, 165-188.

Dearholt, D. W., Schvaneveldt, R. W., & Durso, F. T. (1985). *Properties of networks derived from proximities* (Memorandum in Computer and Cognitive Science, MCCS-85-14). New Mexico State University, Computing Research Laboratory.

Feger, H., & Bien, W. (1982). Network unfolding. *Social Networks, 4*, 257-283.

Gibbons, A. (1985). *Algorithmic graph theory.* Cambridge: Cambridge University Press.

Halasz, F., & Moran, T. P. (1982). Analogy considered harmful. *Proceedings of Human Factors in Computer Systems, CHI '82*, 383-386.

Harary, F. (1969). *Graph theory.* Reading, Mass.: Addison-Wesley.

Harshman, R. A., Green, P. E., Wind, Y., & Lundy, M. E. (1982). A model for the analysis of asymmetric data in marketing research. *Marketing Science, 1*, 205-242.

Hutchinson, J. W. (1981). *Network representations of psychological relations.* Unpublished Ph.D. dissertation, Stanford University.

Jackson, P., & Lefrere, P. (1984). On the application of rule-based techniques to the design of advice-giving systems. *International Journal of Man-Machine Studies, 20*, 63-86.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika, 32*, 241-254.

Krumhansl, C. L. (1978). Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density. *Psychological Review, 85*, 445-463.

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society, 7*, 48-50.

Kruskal, J. B. (1964). Nonmetric multidimensional scaling: A numerical method. *Psychometrika, 29*, 115-129.

Kruskal, J. B. (1977). Multidimensional scaling and other methods for discovering structure. In Enslein, Ralston, and Wilf (Eds.), *Statistical methods for digital computers.* New York: Wiley.

Manheimer, J., Stone, J. D., & McDonald, J. E. (1983). *A multidimensional scaling analysis of copier-function similarity rated by copier engineers and office workers* (IBM Technical Report No. TR77.0129).

McDonald, J. E., Dayton, J. T., & McDonald, D. R. (1986). *Adapting menu layout to tasks* (Memorandum in Computer and Cognitive Science, MCCS-86-78). New Mexico State University, Computing Research Laboratory.

McDonald, J. E., Stone, J. D., & Liebelt, L. S. (1983). Searching for items in menus: The effects of organization and type of target. *Proceedings of the 27th Annual Meeting of the Human Factors Society*, 834-837.

McDonald, J. E., Stone, J. D., Liebelt, L. S., & Karat, J. (1982). Evaluating a method for structuring the user-system interface. *Proceedings of the 26th Annual Meeting of the Human Factors Society*, 551-555.

Miller, D. P. (1981). The depth/breadth tradeoff in hierarchical computer menus. *Proceedings of the 25th Annual Meeting of the Human Factors Society*, 296-300.

Miller, G. A. (1969). A psychological method to investigate verbal concepts. *Journal of Mathematical Psychology, 6*, 169-191.

Miyamoto, S., Oi, K., Abe, O., Katsuya, A., & Nakayama, K. (1986). Directed graph representations of association structures: A systematic approach. *IEEE Transactions on Systems, Man and Cybernetics, 16*, 53-61.

Moran, T. P. (1981). An applied psychology of the user. *Computing Surveys, 13*, 1, 1-11.

Murphy, G. L., & Medin, D. L. (1985). The role of theories in conceptual coherence. *Psychological Review, 92*, 289-316.

Newell, A., & Simon, H. A. (1972). *Human Problem Solving.* Englewood Cliffs, N.J.: Prentice-Hall.

Norman, D. A. (1982). Steps toward a cognitive engineering: Design rules based on analyses of human error. *Proceedings of Human Factors in Computer Systems, CHI'82*, 378-382.

Norman, D. A. (1986). Cognitive engineering. In D. Norman and S. Draper (Eds.), *User centered system design.* New Jersey: Lawrence Erlbaum Associates.

Rosch, E., & Mervis, C. B. (1975). Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology, 7*, 573-605.

Roske-Hofstrand, R. J., & Paap, K. R. (1986). Cognitive networks as a guide to menu organization: An application in the automated cockpit. *Ergonomics, 29*, 1301-1311.

Sachania, V. (1985). *Link weighted networks and contextual navigation through a database.* Unpublished M. S. thesis, New Mexico State University.

Sattath, S., & Tversky, A. (1977). Additive similarity trees. *Psychometrika, 42*, 319-345.

Schvaneveldt, R. W., & Durso, F. T. (1981). *Generalized semantic networks.* Paper presented at the meeting of the Psychonomic Society, Philadelphia.

Schvaneveldt, R. W., Durso, F. T., & Dearholt, D. W. (1985). *Pathfinder: Scaling with network structures* (Memorandum in Computer and Cognitive Science, MCCS-85-9). New Mexico State University, Computing Research Laboratory.

Schvaneveldt, R. W., Durso, F. T., Goldsmith, T. E., Breen, T. J., Cooke, N. M., Tucker, R. G., & DeMaio, J. C. (1985). Measuring the structure of expertise. *International Journal of Man-Machine Studies, 23*, 699-728.

Schvaneveldt, R. W., McDonald, J. E., & Cooke, N. M. (1985). *The user interface in computer systems: A research program* (Memorandum in Computer and Cognitive Science, MCCS-85-10). New Mexico State University, Computing Research Laboratory.

Shepard, R. N. (1962a). Analysis of proximities: Multi-dimensional scaling with an unknown distance function. I. *Psychometrika, 27*, 125-140.

Shepard, R. N. (1962b). Analysis of proximities: Multi-dimensional scaling with an unknown distance function. II. *Psychometrika, 27*, 219-246.

Shepard, R. N. (1963). Analysis of proximities as a technique for the study of information processing in man. *Human Factors, 5*, 33-48.

Shepard, R. N. (1974). Representation of structure in similarity data: Problems and prospects. *Psychometrika, 39*, 373-421.

Shepard, R. N., & Arabie, P. (1979). Additive clustering: Representation of similarities as combinations of discrete overlapping properties. *Psychological Review, 86*, 87-123.

Snowberry, K., Parkinson, S. R., & Sisson, N. (1983). Computer display menus. *Ergonomics, 26*, 7, 699-712.

Tullis, T. S. (1985). Designing a menu-based interface to an operating system. *Proceedings of Human Factors in Computing Systems (CHI'85) Conference.*

Tversky, A. (1977). Features of similarity. *Psychological Review, 84*, 327-352.

Tversky, A., & Hutchinson, J. W. (1986). Nearest neighbor analysis of psychological spaces. *Psychological Review, 91*, 3-22.

Young, R. M. (1983). Surrogates and mappings: Two kinds of conceptual models for interactive devices. In D. Gentner & A. Stevens (Eds.), *Mental models*. New Jersey: Lawrence Erlbaum Associates, pp. 35-52.