A Study to Develop and Evaluate a Taxonomic Model of Behavioral Techniques for Representing User Interface Designs

by

Joseph Dwight Chase

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

APPROVED:

H. Rex Hartson, Chair

Deborah Hix

Robert S. Schulman

Sallie M. Henry

John Lewis

Sherry Perdue Casali

April, 1994

Blacksburg, Virginia

A Study to Develop and Evaluate a Taxonomic Model of Behavioral Representation Techniques

by

Joseph Dwight Chase

Chairman: Dr. H. Rex Hartson
Department: Computer Science

Abstract

A user-centered approach to interactive system development requires a way to represent the behavior of a user interacting with an interface. While a number of behavioral representation techniques exist, not all provide the capabilities necessary to support the interaction development process. The original goal of this research was to modify and extend the User Action Notation (UAN), a user- and task-centered behavioral representation technique. In order to facilitate and evaluate the improvement in the UAN, we developed and evaluated a taxonomic model of behavioral representation techniques. The development and evaluation of our model followed the epistemological cycle of observation, theorization, and evaluation. The model provides a framework for discussing, analyzing, extending, and comparing existing behavioral representation techniques, as well as being a springboard for developing and evaluating new techniques.

Acknowledgments

I would like to thank my committee chairman, Dr. H. Rex Hartson, for his guidance, support, and patience throughout this research. I would also like to thank the other members of my committee, Dr. Hix, Dr. Henry, Dr. Casali, and Dr. Lewis, for their continued support and mentoring throughout this process. I would especially like to thank Dr. Schulman for his statistical innovation and his patience with my lack of statistical innovation.

I would like to thank the National Science Foundation, the Department of Computer Science at Virginia Tech, and the Department of Computer Science at Radford University for providing the financial support to allow me to pursue my goals.

Finally, I would like to thank my family, my wife Melissa for her support, tolerance, and understanding through these years, my parents, Del and Jan Chase for their encouragement and support throughout my education, Melissa's parents, Don and Wanda Wirt for their support and the occasional distraction from the research, and to all the brothers, sisters-in-law, brothers-in-law, and offspring, Tim, Dan, Lisa, Young, Butch, Denise, Chris, Miles, Sarah, Adam, and Hannah for providing much needed support and distraction through the years.

Table of Contents

1. INTRODUCTION

1.1 Problem Statement

Improving the productivity of users of interactive software is a primary goal of human-computer interaction research. "In 1988, approximately 20.33 million personal computers (not counting terminals and workstations or home PCs) were in use by Americans in the workplace (Census, 1990). If they are used an average of only two hours per workday, that is over 40 million personhours per day, or over 10 billion personhours per year. At, say, $10 per hour for an average computer user's wage, a productivity enhancement of only 1% would result in an annual savings of more than $1 billion" (Hartson, 1990).

One of the most effective ways of improving the productivity of users of interactive software is by improving the *usability* of the user interface. Usability is defined as ease of learning, speed of user task performance, user error rate, subjective user satisfaction, and user retention over time (Shneiderman, 1992). Usability includes issues such as effectiveness, efficiency, and naturalness of the user interface as well as the user's reaction to that interface (Hix & Hartson, 1993).

It has been shown repeatedly that traditional software engineering methods do not necessarily lead to high usability when applied to the development of user interfaces (Gould & Lewis, 1985; Rosson, Maass, & Kellogg, 1987; Hartson & Hix, 1989). This result is reasonable since the focus of these methods is the software, not the user. To achieve high usability, the interface development process should be user-centered; i.e., it should focus on the user's tasks, needs, and behavior while interacting with the system. This view, referred to as the *behavioral view* (Hix, et al., 1993). has led to a variety of techniques for representing the design of a user interface in terms of the

behavior of the user, independently of user interface software and hardware considerations. These behavioral representation techniques will be discussed in detail in Chapter 2.

The original charge of this research was to improve and extend one of these techniques, the User Action Notation (UAN) (see Section 2.2.9), through observation of, and interaction with, users of the technique. However, the lack of a practical or theoretical yardstick for determining "improvement" in the technique redirected the research effort to consider development of such a yardstick.

A taxonomic model of behavioral representation techniques, for the purpose of developing and evaluating new behavioral representation techniques, as well as analyzing, extending, and comparing existing techniques, was then developed and evaluated. This model was subsequently used to improve and extend the UAN.

1.2 Approach

1.2.1 A Scientific Process

In order to develop a taxonomic model of behavioral representation techniques, it was first necessary to define the way in which we are using the term *model*. While there are many good definitions from a variety of fields that connote a surrogate for that which is being modeled (e.g., for simulation), the definition most appropriate to our needs characterizes a model as a scientific metaphor for the purpose of the systemization and classification of attributes of a group or class of things (Kaplan, 1964). The word taxonomic was added to emphasize that this model is for the purpose of classification of the attributes of behavioral representation techniques. We will use the word taxonomy interchangeably with the phrase taxonomic model.

Creation of models using the traditional epistemological cycle of

observation, theorization, and *evaluation* is a common theme in many research fields including human factors. Unfortunately, in human-computer interaction, many models are proposed having left out one or more of these three steps. For example, Simon's (Simon, 1988) trade-off space of analytical models, discussed in Section 2.3, was developed only through analysis of existing techniques and was never evaluated. Thus the three steps needed to properly develop a taxonomic model of behavioral representation techniques were:

- *observation*—study of behavioral representation techniques and their use, as well as analysis of related existing models found in the literature (Chapter 3);

- *theorization*—development of a taxonomy of behavioral techniques based on those observations (Chapter 3); and

- *evaluation*—evaluation of the proposed taxonomy of behavioral representation techniques (Chapter 5).

1.2.2 Developing a Taxonomic Model of Behavioral Representation Techniques

While the concept of an empirically derived taxonomy is relatively new to the area of behavioral representation techniques, there is experience from related areas on which to draw. For example, in the area of input devices and device selection, researchers have found it useful to build a taxonomy of devices in terms of their attributes and classes of attributes (Mackinlay, Card, & Robertson, 1990; Bleser, 1991). Similarly, in this research, the behavioral representation techniques to be modeled have a number of attributes based on the needs of their users. With the goal of identifying and classifying these attributes, users of the UAN were asked to contribute a list of their needs for a behavioral representation technique including which activities in the development process used the technique, what

it was used to represent, and the quality of that representation. Further, existing behavioral representation techniques, discussed in detail in Chapter 2, were analyzed to determine qualities for which they were designed. Our analysis of the use of the UAN and of existing behavioral representation techniques was the first step, *observation*, toward developing and evaluating a taxonomic model of behavioral representation techniques. The second step, *theorization*, involved the development of our taxonomic model of behavioral representation techniques based on the data collected in our study of the UAN and our analysis of existing behavioral representation techniques. This taxonomy consists of three dimensions:

- *scope of process activities*—activities within the interface development process that can use the technique. These activities (attributes) include task analysis, design, and so on.

- *content of interaction*—interface or interaction components being represented using the technique. These components (attributes) include user definition, cognitive processes, and so on,

- *quality factors of the technique*—characteristics of the representation. These characteristics (attributes) include ease of use, expressiveness, and so on.

Each axis of this taxonomy is made up of a number of discrete attributes which will be further discussed in Section 3.3. A more detailed discussion of the model development process can be found in Section 3.1.

1.2.3 Evaluating the Taxonomic Model of Behavioral Representation Techniques

Once our taxonomy of behavioral representation techniques had been developed, the third and final step in the development and evaluation of our model was *evaluation.* The first criterion examined was that of *completeness,* defined to be the framework's inclusion of all aspects of behavioral representation techniques. As with the related models listed earlier (Mackinlay, et al., 1990; Bleser, 1991), it was virtually impossible to test the completeness of the taxonomy's representation of behavioral representation techniques. The most one could hope to do is show that the model is complete with respect to currently available behavioral representation techniques. However, a strong argument can be made that the taxonomy should provide ample coverage of these techniques since it was derived from an analysis of them.

Evaluation of the device models (Mackinlay, et al., 1990; Bleser, 1991) centered on the criterion of *utility,* meaning that the models provided some useful function in working with input devices. However, another important criterion described in the literature is that of *reliability*, meaning that the taxonomy must produce similar results across those who use it to compare and evaluate behavioral representation techniques. Evaluation of the taxonomy under these two criteria is described below in Sections 1.2.3.1 and 1.2.3.2 and then again in more detail in Sections 3.2.1 and 3.2.2.

1.2.3.1 Demonstrating the Reliability of the Taxonomic Model

As described previously, the purpose of our taxonomic model is to support development and evaluation of new behavioral representation techniques, as well as analysis, extension, and comparison of existing techniques. Thus, this taxonomy supports two

basic functions:

- *classifying* — mapping problems or critical incidents observed in the use of a specific behavioral representation technique to a missing or failing activity (from the Scope dimension), component (Content dimension), and characteristic (Quality dimension); and

- *rating* — analyzing an existing behavioral representation technique for its ability to cover all activities, components, and characteristics identified in the taxonomy.

Moran (1980) raised the need for the first of these two functions in describing the need to categorize and analyze hundreds of "nits" or critical incidents involving interface development at Xerox. Jacob (1982) raised the need for the second of these functions in discussing the need to analyze and compare new and existing user models and design representations.

The taxonomy was evaluated for *reliability* by having a group of behavioral representation technique users perform classification and rating tasks and then performing a statistical test of the similarity of their results. Detailed descriptions of these experiments and the results of these experiments are included in Chapter 5.

1.2.3.2 Demonstrating the Utility of the Taxonomic Model (Modifying, Extending, and Evaluating the UAN)

The other criterion for evaluation of the taxonomy was to show that it could provide useful function in some measurable way. To do this, the taxonomic model of behavioral representation techniques was applied to the problem of UAN development and extension. If a new version of the UAN based on the taxonomic model could be shown to be a significant improvement over the old version, then the

conclusion could be drawn that the taxonomic model of behavioral representation techniques was, indeed, useful.

In order to facilitate this experiment, the old version of the UAN was documented, then the taxonomic model was applied to create and document a new version of the UAN. Next, two groups of relatively equal behavioral representation technique users were selected, giving one group the old version of the UAN and the other group the new version of the UAN. Both groups then wrote design representations for the same user interface. The improvement in the UAN was then statistically tested in two ways: the user's perception of the version of the UAN that they used as recorded by questionnaire results and the quality of the representations of the interface as determined by a panel of human-computer interaction experts. Anecdotal data, such as time to complete the description, frustration level of participants, etc., were also collected. Chapter 4 provides a discussion of the use of the model and the changes to the UAN. A more detailed discussion of the evaluation of the UAN and the model is included in Chapter 5.

1.3 Contribution

The contribution of this research to the field of human-computer interaction is at three different levels:

- Improvement in the UAN
- Development and Evaluation of a taxonomic model of behavioral representation techniques
- Exploration and demonstration of the epistemological cycle applied in human-computer interaction.

This three level structure of both the contribution and the approach of this research is illustrated in Figure 1-1.

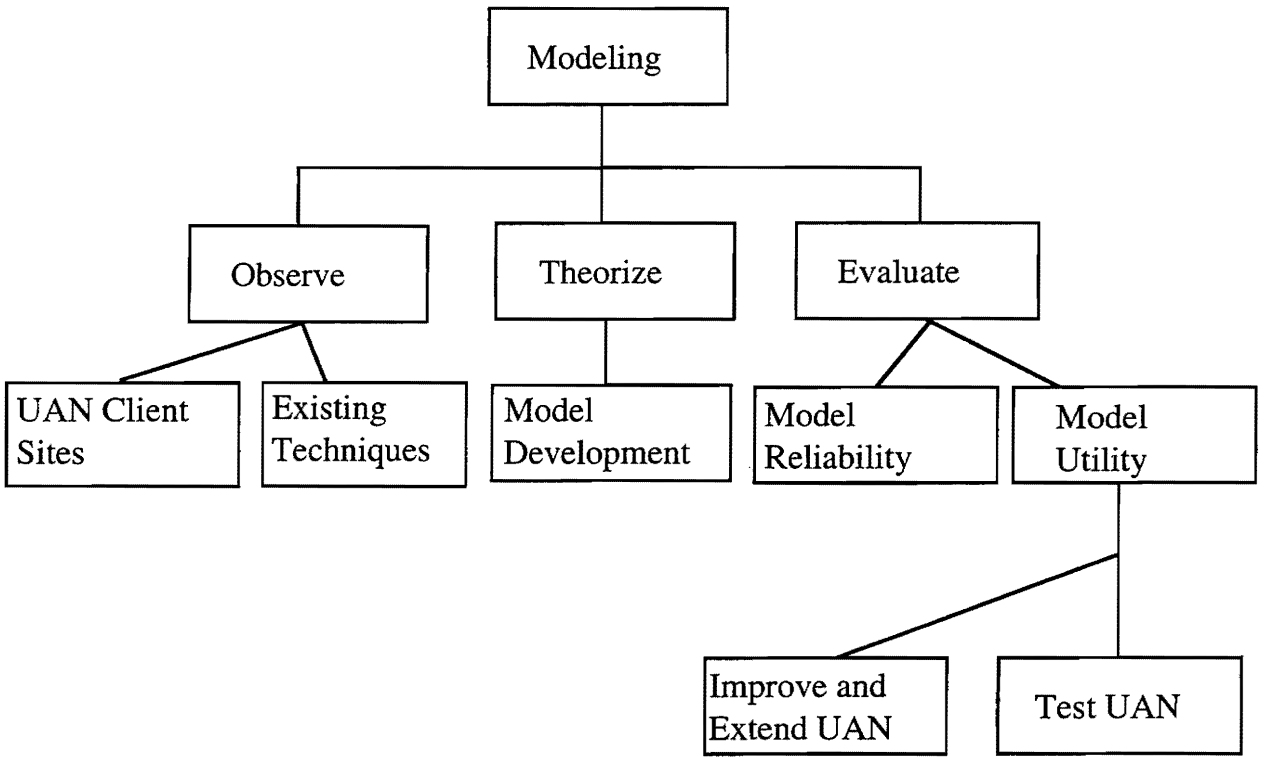


Figure 1-1. Levels of research contribution and approach

2. BACKGROUND AND RELATED WORK

With the exception of one attempt at a theoretical framework (Simon, 1988) which is discussed further in Section 2.3, there has been little done to describe behavioral representation techniques. The bulk of literature to date has focused on the development, analysis, or extension of a single technique, usually for a very specific purpose. In this chapter, we will briefly touch on a variety of constructional techniques for the representation of user interface designs (Section 2.1), we will detail the existing behavioral representation techniques (Section 2.2), we will discuss related models within human-computer interaction (Section 2.3), and we will summarize the need for and the purpose of our taxonomic model of behavioral representation techniques (Section 2.4).

2.1 Constructional Representation Techniques

A variety of techniques have been developed to represent the user interface portion of a system in a constructional manner, i.e. from the system's perspective. These include:

- State-transition diagrams (Jacob, 1985; Wasserman & Shewmake, 1985; Yunten & Hartson, 1985; Jacob, 1986)
- Event handlers (Green, 1985; Hill, 1987)
- Concurrent programming (Cardelli & Pike, 1985; Flecchia & Bergeron, 1987)
- User interface development environments (Foley, Gibbs, Kim, & Kovacevic, 1988)
- Object-oriented programming (Jacob, 1986; Sibert, Hurley, & Bleser, 1988)

The constructional view is necessary and useful in supporting the translation of user interaction design— designing the behavior of the system and the user in their cooperative performance of a user task—

into user interface software design and implementation. However, the behavioral view is necessary to support a focus on the needs of the user within the interaction design process and to capture the behaviors of both user and system detailed in user interaction design. Correspondingly, since our purpose in this research was to examine behavioral representation techniques, we will not expand the discussion of constructional representation techniques.

2.2 Behavioral Representation Techniques

The user interface is not a communication mechanism between application and user, but between designer and user (Nadin, 1988). The objective should be to minimize the difference between the user's mental model and the system image (Norman, 1986). To this end, a number of behavioral or user-centered representation techniques have been developed. Among these are:

- GOMS (Card, Moran, & Newell, 1983)
- Command Language Grammar (CLG) (Moran, 1981)
- keystroke-level model (Card & Moran, 1980)
- Task Action Grammar (TAG) (Payne & Green, 1986)
- Reisner Action Language (Reisner, 1981)
- work by Kieras and Polson (Kieras & Polson, 1985)
- scenarios or story-boarding
- Task Artifact Framework (Carroll, Kellogg, & Rosson, 1991)
- User Action Notation (UAN) (Hartson, Siochi, & Hix, 1990)

These techniques were generally designed for one of two purposes:

- predictive evaluation of existing interfaces, or
- providing a usability focus in the design and development process (Young, 1989).

Some of these techniques focus on one specific aspect of the

interaction and may not provide any representation at all for other aspects. Many of these techniques trade off prediction and/or accuracy in some areas for a lack of either in others (Young, 1989). Each of these techniques is discussed further in the following sections.

2.2.1 GOMS (Goals, Operators, Methods, Selection Rules)

The basic component of a GOMS (Card, et al., 1983) description of an interface is a simple task called a unit task. This unit task, made up of user actions, mental workload information, and prediction information, is found by decomposing the user's primary goal into subgoals creating a goal structure. These goals and subgoals are then accomplished by means of operators or simple user actions. There are two primary criticisms of GOMS (Carroll & Olson, 1988; Wilson, Barnard, Green, & Maclean, 1988; Butler, 1989). First, the concept of the unit task is not well defined. This may lead to different analysts or designers determining different sets of unit tasks for the same design. Second, there is no specified method for creating the goal structure. Again, this may lead to different analysts or designers determining different goal structures for the same interface. The variability of this creative process can lead to inconsistent representation of the user's goal structure, which, in turn, can lead to inaccurate predictions.

The original purpose of GOMS was the analysis of existing interfaces in order to predict user performance. In fact, GOMS has been shown to be an excellent predictor of error-free user performance (Olson & Olson, 1990). GOMS is the only behavioral representation technique with a prediction mechanism based on psychological theory (Johnson, 1988). However, GOMS was not intended as a representation technique in the development process (Carroll, et al., 1988; Wilson, et al., 1988). Further, since GOMS is intended for the evaluation and prediction of user performance as opposed to design generation, it

does not detail any particular method of task analysis or design synthesis (Johnson, 1991). GOMS provides only serial execution of operators and only represents error-free performance (Card, et al., 1983; Wilson, et al., 1988). GOMS does not represent interface feedback or interface state information.

2.2.2 Command Language Grammar (CLG)

The Command Language Grammar (CLG) (Moran, 1981) is a design representation intended to describe the user's conceptual model of the system (Reisner, 1983). Unfortunately, this usually results in CLG being a representation of the designer's mental model of the user's mental model of the system, which is a difficult proposition at best (Johnson, 1988). CLG is primarily a representation to help generate and evaluate alternative designs as opposed to being a design methodology (Reisner, 1983).

CLG has a tree-like task structure very similar to the goal structure of GOMS. This structure is made up of three components: Conceptual, Communications, and Physical. The conceptual component consists of the task level containing user tasks and the semantic level containing concepts and manipulation of objects. The communications component consists of the syntactic level containing the command language structure and the interaction level which contains a finite set of structures from which all user-system dialogues can be described. Finally, the physical component consists of the spatial layout referring to the layout of the screen and other output devices and the device level containing the physical devices with which the user has contact. A CLG description specifies sets of entities, tasks, procedures and methods at each level (Moran, 1981). Each level in a CLG description consists of a complete description of the system at that level of abstraction. This fact makes it apparent that a CLG description of a system can be very thorough, and also very cumbersome.

CLG provides prediction mechanisms for memory load and learning time, but not performance (Johnson, 1988). However, the designers of CLG envision a time predictor as weighted sum of primitive actions (Wilson, et al., 1988). CLG also provides analysis of knowledge in real-world tasks. As with GOMS, CLG is limited to error-free, sequential user performance.

2.2.3 Keystroke-Level Model

The Keystroke-Level Model (Card, et al., 1980) was one of the earliest attempts at performance prediction. It is based on strings of user and system activities which make up user tasks. The model is made up of 5 variables: keystrokes, pointing (with a mouse), homing (finding device), drawing (straight line segments), and mental preparation (based on chunking). As with GOMS and CLG, the Keystroke-Level Model predicts error-free, sequential performance only (Wilson, et al., 1988). Also like GOMS, this model does not represent interface feedback or interface state which are key components of interaction design.

Like GOMS, the Keystroke-Level Model breaks user activity up into simple or unit tasks, in this case based on memory units. The time to accomplish one of these tasks is then given by:

$$t(\text{task}) = t(\text{acquire mentally}) + t(\text{execute physically})$$

where t(acquire mentally) is an empirically derived constant representing the time it takes the user to choose which task to perform and t(execute physically) is the sum of the five variables listed above as given by (Card, et al., 1980):

$$t(\text{execute physically}) = tk + tp + th + td + tm + tr(\text{response})$$

As with GOMS, this model is not directly concerned with design generation, but instead focuses on the evaluation and prediction of user performance and does not detail any particular method of task analysis (Johnson, 1991).

2.2.4 The Task Action Grammar (TAG)

The Task Action Grammar (Payne, et al., 1986) attempts to model the user's understanding of the task language using production rules (Payne, et al., 1986; Wilson, et al., 1988; Green, 1989). The purpose of this model is to represent the user's knowledge of the mappings from user tasks to user actions and to predict learnability by capturing all generalities and similarities among tasks about which the user may be aware.

As with GOMS and the Keystroke-Level Model, TAG decomposes the user's tasks into "simple tasks" which are defined to be any task that a user can routinely perform. Again, as with the previous techniques, this imprecise definition of simple tasks leads to some ambiguity as to how the user's general goals are to be decomposed. Further TAG does not represent temporal relations among tasks, interface feedback, or interface state.

TAG provides prediction mechanisms for consistency by comparing the rules for similar tasks, for completeness by analysis for missing rules, and for complexity by counting the number of rules. However, this latter point is somewhat in dispute in that there is no existing theory to support the idea that more rules translates into higher complexity (Johnson, 1988). TAG also provides a prediction mechanism for learnability by comparing the features encoded in the rules to the user's real-world tasks (Olson, et al., 1990). Again, however, as with many of the previous techniques, this raises the question of the designer's ability to "correctly" identify the user's real-world tasks and their structure. Further, since TAG is not a

design representation technique but rather an analytic interface representation technique, it does not provide any method for task analysis or design synthesis (Johnson, 1991).

2.2.5 The Reisner Action Language

The Reisner Action Language (Reisner, 1981) is a formal, Backus-Naur Form (BNF) grammar based representation of the user interface structure. In this language, terminal symbols represent user actions, both cognitive and physical, and non-terminals represent groups of user actions. There is no representation for temporal relationships between tasks other than sequencing though there is reference in the literature to simultaneous activity. However, this idea is not well defined. Further, as with many of the other techniques, there is no representation of interface feedback or interface state information. Construction of the set of terminals and non-terminals represents a hierarchical structure not unlike that of GOMS, CLG, etc., and as with GOMS and CLG, this introduces the problem of the designer's creative judgment affecting the resulting structure.

Reisner claims that this technique is useful for predicting usability and learnability of an interface as well as complexity. However, as with TAG, these are at least partially measured by counting the number of production rules which has already been stated to have no foundation in theory. Reisner did perform an empirical validation of the model which may provide a foundation for future work in using the number of rules as a basis for prediction. Differences between interface designs are predicted by examining the length of strings in the language. As with TAG, this technique is not intended to be a design representation technique and does not provide the analysis and synthesis methods necessary for interface design and development. In fact, Olson and Olson (Olson, et al., 1990) refer to both techniques as models of knowledge content (required user competence) rather than models of a full system.

2.2.6 Kieras and Polson's Cognitive Complexity Theory

Kieras and Polson's Cognitive Complexity Theory (CCT) (Kieras, et al., 1985) is intended to represent the complexity of a user interface from the user's perspective. This technique represents an interface as the combination or mapping between the user's job-task environment and the interaction device behavior. The user's job-task environment, which is similar to the goal structure of GOMS, is represented by sets of production rules put together very much like a program. Device behavior is represented by Generalized Transition Networks (elaborated state-transition diagrams). The combination of the production rules and the generalized transition networks can then be "run" like a program to simulate the user interacting with the system. This simulation of the user interacting with the system can then be used to provide performance predictions and complexity information.

The addition of device behavior information helps to overcome the shortcomings of GOMS with respect to interface feedback and interface state information. However, because of its similarity to GOMS, this technique inherits the problems associated with GOMS such as the lack of a formal method for constructing the user's goal structure (Bennett, 1986; Butler, 1989). Further, there is some question as to the accuracy of using a set of production rules or a program to simulate the user for prediction purposes. As with most of the previous techniques, CCT is intended for prediction not interface design and development and therefore does not support analysis and synthesis activities necessary for interface development.

2.2.7 Scenarios

Scenarios or story-boarding provide an effective method of conveying an early picture of the appearance of an interface. However, scenarios are only an example of the interface so there are limitations on the level of completeness they can provide (Hix, et al., 1993).

2.2.8 The Task Artifact Framework (TAF)

The Task Artifact Framework (TAF) (Carroll, et al., 1991; Rosson, 1992) is a user-scenario based interactive system design methodology that represents a designed artifact or interface object using the set of scenarios supported by that artifact. Using this technique, a set of user scenarios is iteratively refined until it is of sufficient detail to fully specify most of the interactive system features. These scenarios can then be represented as causal schemas or claims about the artifacts within the scenarios in order to determine the underlying rationale (e.g., why would the user select a particular object? or why would this feature obstruct the user?). It is important to note that artifacts can include features or objects within the development environment as well as in the interface itself, e.g., tools, or development environments that may affect design rationale. One of the current failings of TAF, as stated by Rosson (Rosson, 1992), is that for complex designs, the number and variety of scenarios and claims makes management of a TAF design representation very complex.

TAF alone probably does not constitute a behavioral representation technique but rather a behavioral description methodology. One way of viewing TAF is as the interaction component equivalent of Object-oriented Design (OOD). OOD focuses on defining computational objects that model problem entities as closely as possible. TAF attempts to

approach interface artifacts the same way. In fact, Carroll and Rosson have referred to TAF as "constructivist" (Rosson, 1992).

The main purpose of TAF, that of codifying and generalizing design rationale, is unique among behavioral representation techniques and in fact representation techniques in general. Due to that focus, TAF is weak on formally specifying the details of the interaction component of a system. However, when used in tandem with OOD, the scenarios and claims within TAF eventually converge and evolve into a working system including temporal relationships, interface feedback, and state information.

2.2.9 The User Action Notation (UAN)

The User Action Notation (UAN) is a user- and task-oriented notation that describes the behavior of a user and an interface during their cooperative performance of a task (Hartson, et al., 1990). The primary abstraction of the UAN is a *user task* — a user action or group of temporally related user actions performed to achieve a work goal. A user interface is represented as a quasi-hierarchical structure of tasks that are *asynchronous* — the sequencing within each task is independent of that in the others. User actions, corresponding interface feedback, and state information are presented at the lowest level. Levels of abstraction, in which lower level tasks are combined under a single general task name, are used to hide these details and represent the entire interface. At all levels, user actions and user tasks are ordered and combined using temporal relations such as sequencing, interleaving, and concurrency (Hartson & Gray, 1992). Since textual notations are not always convenient for specifying all components of an interface, the UAN includes screen pictures, or scenarios, and can be supplemented with state-transition diagrams to indicate precisely how the user interacts with the interface.

The UAN is primarily a notation for behavioral representation of an interaction design. However, through empirical work with industrial users of the UAN, we have found that it has a variety of uses across the entire interaction development process. As we continue to collect information on how the UAN is being used, a composite method of organization, representation, and communication has emerged. The UAN is continuing to evolve as a behavioral representation technique as is discussed further in Section 3.2.2.

2.3 Existing Models and Analysis Techniques

Bleser (Bleser, 1991) stated that there are three things missing from interaction design methodologies:

- a view of interaction broader than data,
- a notational mechanism for recording human factors information,
- an explicit description of device selection.

She then defined an explicit description of device selection, based on a taxonomic model of input devices derived from device attributes, as well as a notational mechanism for recording human factors information about input devices.

Mackinlay et al. (Mackinlay, et al., 1990) also found it useful to design a taxonomic model of input devices based on attributes of those devices. Both the Mackinlay and Bleser device models were evaluated by application to a wide variety of devices to show their utility for device description. While these models are not applicable to the area of behavioral representation techniques, their empirical development, based on device attributes, and evaluation does suggest a method of empirically creating and evaluating a taxonomic

model of behavioral representation techniques based on the attributes of those techniques.

As mentioned above, there has been one attempt to create a theoretical model of techniques to represent interface designs from the user's perspective. Simon (Simon, 1988) created what he called a "trade-off" space of user models. The purpose of this model was to create a theoretical view of the pros and cons of any given representation technique. The three dimensions of this model are: Processing Resources (e.g., actions, cognition, perception and sensation), Knowledge Representation (e.g., from high to low, continuously), and Level of Representation (e.g., from actual operations specified to no operations specified, continuously). This space, as shown in Figure 2.3-1, was to help a designer/analyst choose between the available techniques according to the criteria listed above. For example, if the designer/analyst wanted a high degree of knowledge representation without having the operations specified, they might choose TAG since it is in the upper left quadrant of the space.

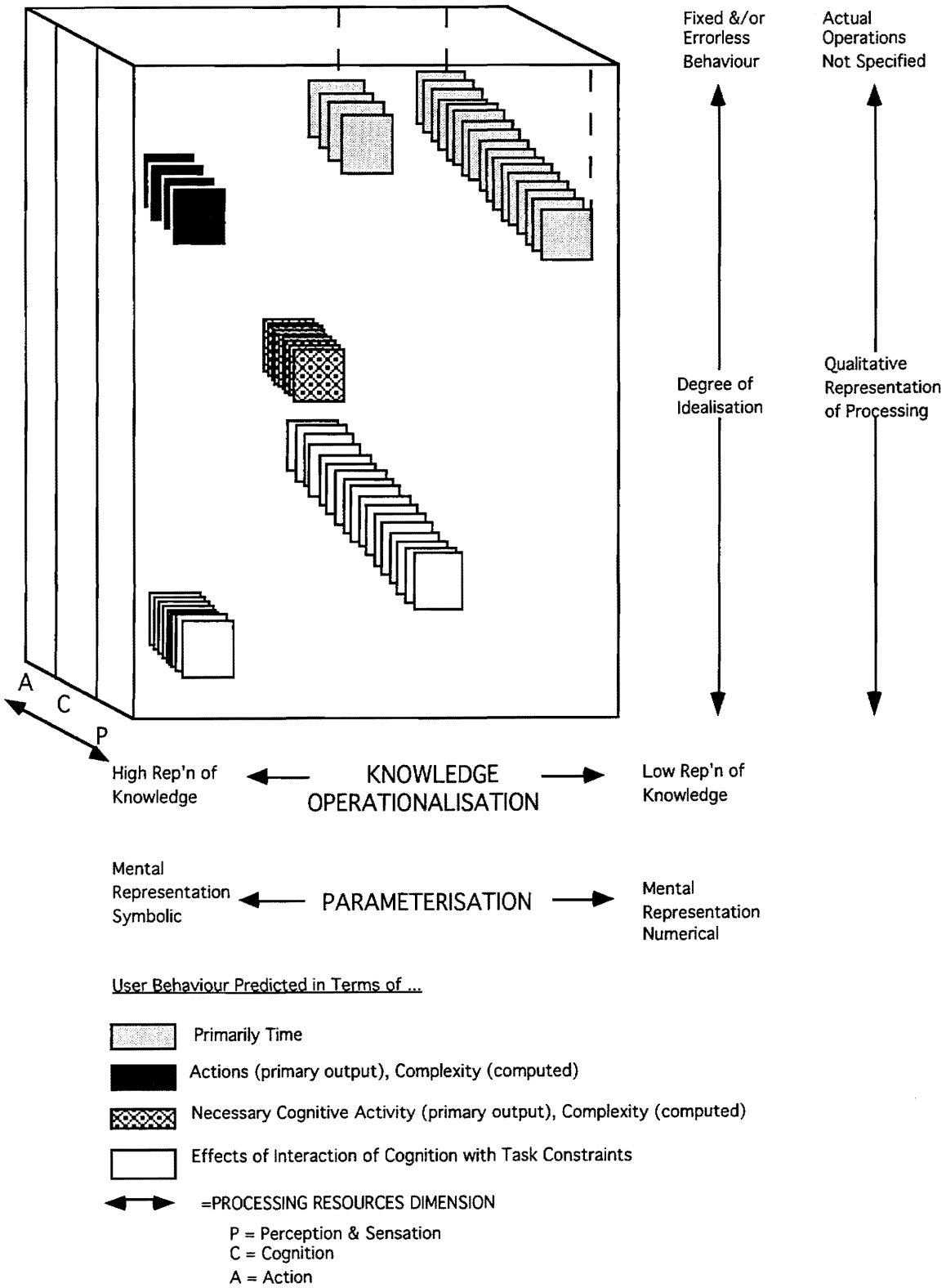


Figure 2.3-1. Simon's "Trade-Off Space" for User Models

While this "trade-off space" raises many interesting questions about the relationships among the various techniques, it does not go very far in terms of answering them. First, the space has not been evaluated in any way. Second, the use of continuous axes for two of the dimensions raises questions about where a given technique should reside. For example, what if a technique is very explicit about cognitive operations but not about physical ones. Should it reside near the top or the bottom of the Level of Representation axis?

2.4 Summary

The lack of a practical or theoretical ideal for behavioral representation techniques has been previously recognized. Young (1989b) stated that there is currently no clear method for the development and evaluation of new techniques for representing human-computer interaction. Certainly, one requirement of these new techniques is that they represent a wider range of behavioral phenomena. Moran (Moran, 1980) found that a comprehensive framework or taxonomic model was needed to bring together a variety of user interface representation and development issues. Jacob (Jacob, 1982) added that any such model must support:

- the development of specification languages for interface design,
- the development of cognitive models of users, and
- the comparison of these models and notations.

It is apparent that what is needed is a taxonomic model of behavioral representation techniques for the purpose of developing and evaluating new behavioral representation techniques, as well as analyzing, extending, and comparing existing techniques.

3. DEVELOPING A TAXONOMIC MODEL OF BEHAVIORAL REPRESENTATION TECHNIQUES

As described in Chapter 1, the first step in the epistemological cycle is *observation*. Our first problem was to determine what to observe. This problem led to a two-pronged solution. First, we observed the use of the User Action Notation (UAN) in a variety of environments and locations as further described in Section 3.1. Second, since this was to be a model of behavioral representation techniques in general, we analyzed existing behavioral representation techniques as described in Section 3.2. Once the observations from these two sources had been collected, we then proceeded with the second step in the epistemological cycle, i.e. *theorization*, by developing a taxonomic model based on those observations. The model, and the rationale for the inclusion of each of its components, is discussed in Section 3.3.

In addition to our observation of behavioral representation techniques, we also studied related models and approaches to their development found in the literature. For example, in the area of input devices and device selection, researchers have found it useful to build a taxonomic model of devices in terms of their attributes and classes of attributes (Mackinlay, et al., 1990; Bleser, 1991). Similarly, in our research, the behavioral representation techniques to be modeled have a number of attributes based on needs of their users. With the goal of identifying and classifying these attributes, industrial and academic users of the UAN were asked to contribute their needs for, and experiences with, behavioral representation techniques.

Further, existing behavioral representation techniques, such as GOMS, CLG, TAG, etc., were analyzed to determine qualities for which they were designed. Results included characteristics such as performance prediction, cognitive modeling, and object and artifact definitions.

This section details the empirical and analytical development of the taxonomic model of behavioral representation techniques. Chapter 4 provides an overview of the use of our model. Chapter 5 discusses the evaluation of the taxonomic model of behavioral representation techniques and Chapter 6 discusses the conclusions and implications to be drawn from the results detailed here.

3.1 Empirical Development of the Taxonomic Model

Empirical development of the taxonomic model was driven by interaction with and observations of users of the UAN. The UAN has been introduced to a large number of industrial and academic interface development sites (approximately 60). The characteristics of these sites are discussed further in Section 3.1.1. Interaction with and observations of these sites centered on critical incidents, both positive and negative, that occurred during the interaction development process using the UAN. Critical incidents are discussed in more detail in Section 3.1.2. Finally the critical incident data were analyzed to derive components of the taxonomy. This analysis process is described in Section 3.1.3.

3.1.1 UAN Client Sites

The UAN is currently or has been in use at more than 60 sites around the world, ranging from local businesses to European universities. It is important to note that while data were collected from most of these sites, we were in regular contact with only a small subset of this group.

Four independent variables were combined to represent the ways that UAN data were collected from sites, including (Hartson, 1990):

- *team size: individual vs. group*—a group of several users interacting while using UAN yielded data about needs for interrole communication and group dynamics, whereas an individual using the UAN provided data about the needs of a single developer role.

- *experimenter location: remote vs. local*—having a local experimenter (i.e., experimenter directly observes interface designer, either in a lab setting or in an industrial setting, producing interface designs using the UAN) yielded rich real-time qualitative data through critical incident identification and verbal protocol taking, whereas a remote interface designer was able to work uninterrupted, providing data on ease of use and usefulness of the UAN.

- *task type: real vs. benchmark*—a real-world task (e.g., interface designers work on an interface from their own development environment) yielded data about using the UAN to represent an actual interface. Benchmark tasks (Appendix E) produced "control" data, allowing identification and factoring out of individual differences among developers, and identification of recurring weaknesses and strengths in the UAN.

- *task environment: field vs. lab*—a field trial, in which the UAN is used in the user's own work environment, again yielded data on real-world use of the UAN, whereas a lab setting, an artificial environment, produced more controlled data because the researcher can manipulate subjects and tasks to study specific areas of interest in the UAN.

There are 16 possible combinations of these four independent variables shown in Table 3.1-1 (Hartson, 1990). Several of these conditions contain unlikely or uninteresting combinations of the four variables such as an individual working on a real task at a remote site in a lab setting. In all, six of these, conditions 2, 4, 6, 10, 12, and 14 are not useful and were not pursued. The other conditions were tested as sites became available. In some cases, one site satisfied more than one condition. This categorization of sites provides a basis for ensuring that we received data from a representative sample of development environments.

Table 3.1-1. Possible Experimental Conditions

Condition	Team Size	Location	Task Type	Environment
1	individual	remote	real	field
2	individual	remote	real	lab
3	individual	remote	benchmark	field
4	individual	remote	benchmark	lab
5	individual	local	real	field
6	individual	local	real	lab
7	individual	local	benchmark	field
8	individual	local	benchmark	lab
9	group	remote	real	field
10	group	remote	real	lab
11	group	remote	benchmark	field
12	group	remote	benchmark	lab
13	group	local	real	field
14	group	local	real	lab
15	group	local	benchmark	field
16	group	local	benchmark	lab

By way of example, seven of the sites that reported data during this research are further described below. For the sake of various non-

disclosure and privacy issues involved, none of these sites are named. Instead they are referenced using a site number. Sites are described individually here, including the experimental condition(s) they cover.

Site 1 was a local industrial site. This company is a medium-sized software developer conveniently located for on-site visits by members of the research group. A project group within this company, developing a security system for applications such as banks and airports, originally approached the UAN as a tool to aid in description and documentation of a system which had, theoretically, already been designed. However, soon after beginning the process of translating the existing design into UAN, it was discovered that the existing design was inconsistent and incomplete. Use of the UAN led to this discovery because of the requirement that both hierarchical and temporal relationships between tasks must be specified. These relationships were not fully specified in the original design document. It was apparent that use of the UAN imposed structure and method that helped reveal these areas of inconsistency and incompleteness (Chase & Hix, 1994). This group fully translated their existing design into the UAN and completed their design using the UAN. At times this team worked on the UAN as a group and at times the work was left to a single individual. Furthermore, while we provided a two to three hour presentation on the UAN and were able to capitalize on the opportunity to observe this site's UAN work in progress and even participate in design review meetings, direct observation was not always possible. This site encompassed experimental conditions 1, 5, 9, and 13.

A full year software engineering course at another university provided site 2. This course involved six senior students who started out working individually on benchmark tasks (Appendix E) involving simple desk-top or operating system level activities. Next, they formed two groups of three for an exercise with real tasks where

each group implemented the other's design. This was a remote site, although we did have the opportunity to visit this site and present a six hour short course on the UAN. We were also able to review their progress by mail and by e-mail and make a follow up visit at the end of the academic year. This site was included in experimental conditions 3 and 11.

A graduate course in human-computer interaction at our own university provided site 3. Students worked with the UAN both individually and in teams. Their tasks in both cases were benchmark tasks (Appendix E). For individual work, the tasks involved complex desk-top or operating system level activities. The tasks for team work were simple tasks designed around new interaction styles (e.g., gestural). These students were exposed to the UAN in the classroom and were presented with a written UAN tutorial (Appendix A). Since this was a local site, it fell under experimental conditions 8 and 16.

Another graduate course in human-computer interaction provided site 4. Students in this course worked with the UAN in teams of three to six. The tasks were both real and benchmark. The benchmark tasks were developed around a simple interactive calendar management system as developed in the new UAN tutorial (Appendix B). These students had two days of classroom instruction with the UAN and were given a dictionary of UAN symbols. This site was local and encompassed experimental conditions 13 and 15.

Site 5 was a local site at which an individual was developing a language tool for the UAN. This individual had technicians available to him for design and implementation assistance. Therefore, some of the work at this site was done on an individual basis while other parts were done as a group. All of these participants had classroom exposure to the UAN as well as a number of sessions with members of the project group. This site covered experimental conditions 5 and 13.

Site 6 was a remote site at which an individual was working as a freelance engineer within a large software development company. This UAN user had completed projects both individually and as part of a group where he was the only UAN user. The tasks involved were typically real-world tasks. This individual was given a variety of UAN publications and had a number of sessions with members of the project group. This site encompassed conditions 1 and 9.

Site 7 was a local site at which an individual was working as the interface designer for a large inter-disciplinary database development project at our own university. This was a real-world task being performed by a single individual. This site was local, allowing our direct observation of, and participation in, the interface development process. This individual had classroom experience with the UAN as well as regular interaction with members of the project group. This site covered condition 5.

3.1.2 Data Collection

Four types of data were collected from each of the client sites. These were:

- *needs analysis data*—these were UAN users' responses to the question "What are your needs and requirements for a behavioral representation technique?" This type of data was also gathered by observing how the UAN was used especially at sites where it was adapted to meet a particular need.

- *written UAN descriptions*—these were UAN descriptions that were analyzed to look for variations in notation, method, style, consistency, etc.

- *qualitative data*—these consisted of UAN users' subjective opinions about various aspects of the technique.

- *method and scope data*—these are from either observation or reporting of how and where the UAN was used.

All four types of data contributed to the taxonomy of behavioral representation techniques. The latter three types of data, written UAN descriptions, qualitative data, and method and scope data, yielded information about critical incidents. For this research, a critical incident was defined to be:

- an encounter with an interface component that the UAN did not represent,

- a difficulty using the existing UAN notation,

- variations in notation or method of using the UAN, or

- an incident in which the UAN provided a notation or a view of the interface that would not have occurred otherwise.

3.1.3 Data Analysis

As needs analysis and critical incident data were collected, they were analyzed to derive components of the taxonomic model. Each piece of data was examined to determine which activity in the interaction development process had produced that piece of data (e.g., task analysis, design, etc.), what component of the interaction was involved (e.g., user actions, interface feedback, etc.), and the

characteristic of the UAN that caused the critical incident or failed to meet the UAN user's needs (e.g., learnability, readability, etc.). Thus, each piece of data was divided into three dimensions:

- *scope of process activities*—activities within the interface development process that can use the technique. These activities (attributes) include task analysis, design, and so on.

- *content of interaction*—variety of different interface or interaction components that can be represented using the technique. These components (attributes) include user class definition, cognitive processes, and so on,

- *quality factors of the technique*—characteristics of the representation. These characteristics (attributes) include ease of use, expressiveness, and so on.

Each piece of data was analyzed along these three dimensions. Any new values for any of the three dimensions were then added to our list of values for that dimension. For example, if an incident had not previously been reported involving the representation of hierarchical relationships among user tasks and one was reported, then this item was added to the list of items to be modeled under the Content dimension. The determination as to whether an item belonged under Scope, Content, or Quality dimension was done by determining whether the item described the representation technique's role in the interaction development process (scope), the technique's coverage of components of the interaction (content), or characteristics of the representation technique itself (quality).

In this way, a comprehensive listing was gathered of which activities were using the UAN, what interaction components were being represented, and what required characteristics there were for that representation. Having been gathered empirically, there was no way of determining the completeness, or lack thereof, of this listing.

3.2 Analytical Development of the Taxonomic Model

Empirical development of the taxonomic model, as described throughout Section 3.1, had two problems: It dealt only with UAN users so there was no representation in the taxonomy for things that the UAN did not address and UAN users did not request, and as mentioned above, there was no way of determining completeness of the taxonomy. Therefore, we found it necessary to analyze other existing behavioral representation techniques to ensure that all their components and uses were included in the taxonomy as well. In doing this, it was important to pay particular attention to the literature surrounding a technique that was not written by the creators of the technique. This search for objective reviewers provided information about what these techniques actually could or could not do as opposed to what the designers of each technique hoped it might be able to do.

Our analysis of GOMS as a representation technique provides an excellent example of this analytical process in that GOMS provides representation for two components that the previous version of the UAN did not: user performance information and mental workload. By finding this information in the surrounding literature, we had now discovered that these were components of the Content dimension (since they are describing characteristics of the interaction) which should be included in our taxonomy of behavioral representation techniques. Further, the literature describes the user performance and mental workload information in GOMS as typically being associated with the activity of interface evaluation (as opposed to interface design representation) which reinforced the presence of the evaluation activity in the Scope dimension (needs analysis data and critical incidents from UAN users had already indicated that the evaluation activity should be present in the Scope dimension).

It is important to note that this analysis was not limited to positive

attributes of techniques. In fact, some of the best information about the various representation techniques came from critical accounts in independent literature reviews. For example, almost all the techniques that generate hierarchical goal/task structures have been criticized for a lack of reliability in how that structure is created (e.g., two or more designers/analysts might produce different goal/task structures for the same interface) (Johnson, 1988). This indicates that an important characteristic or quality of behavioral representation techniques is reliability, a measure of whether different designers/analysts produce similar results.

By surveying all the behavioral representation techniques listed in Chapter 2, the taxonomy was developed to encompass a broad range of behavioral representation techniques. Section 3.3 includes a description of the taxonomic model including the sources of each of the components within the taxonomy.

3.3 The Resulting Taxonomic Model of Behavioral Representation Techniques

Figure 3.3-1 illustrates our taxonomic model of behavioral representation techniques. The three dimensions of the taxonomy, *scope of process activities, content of interactions,* and *quality factors of the technique*, do not represent continuous variables; rather, they demark a three-dimensional space with discrete points along each edge. The taxonomy facilitates characterization of the capabilities, coverage, and appropriate applicability of an existing behavioral representation technique. Further, the taxonomy helps anticipate problems or missing elements within a behavioral representation technique by examining the technique at each point or cell. For example, if a point on the *quality factors of the technique* axis is automated translation of the design representation into a prototype and a point on the *content of interaction* axis is interface feedback,

then the taxonomy suggests examining the technique's capabilities with respect to automated translation of feedback notation. In this way, the taxonomy facilitates examination of all points in the three dimensional space for the behavioral representation technique under investigation. By rating a behavioral representation technique at each point in the taxonomy, areas of possible improvement for that technique will be illustrated by points or subspaces within the taxonomy containing low ratings. In the same way, the taxonomy may suggest areas of needed extension. For example, the UAN previously provided no method for recording user task performance prediction information, a characteristic that has already been established as an important component of the *content of interaction* dimension. This points to performance prediction as a possible area of future extension for the UAN.

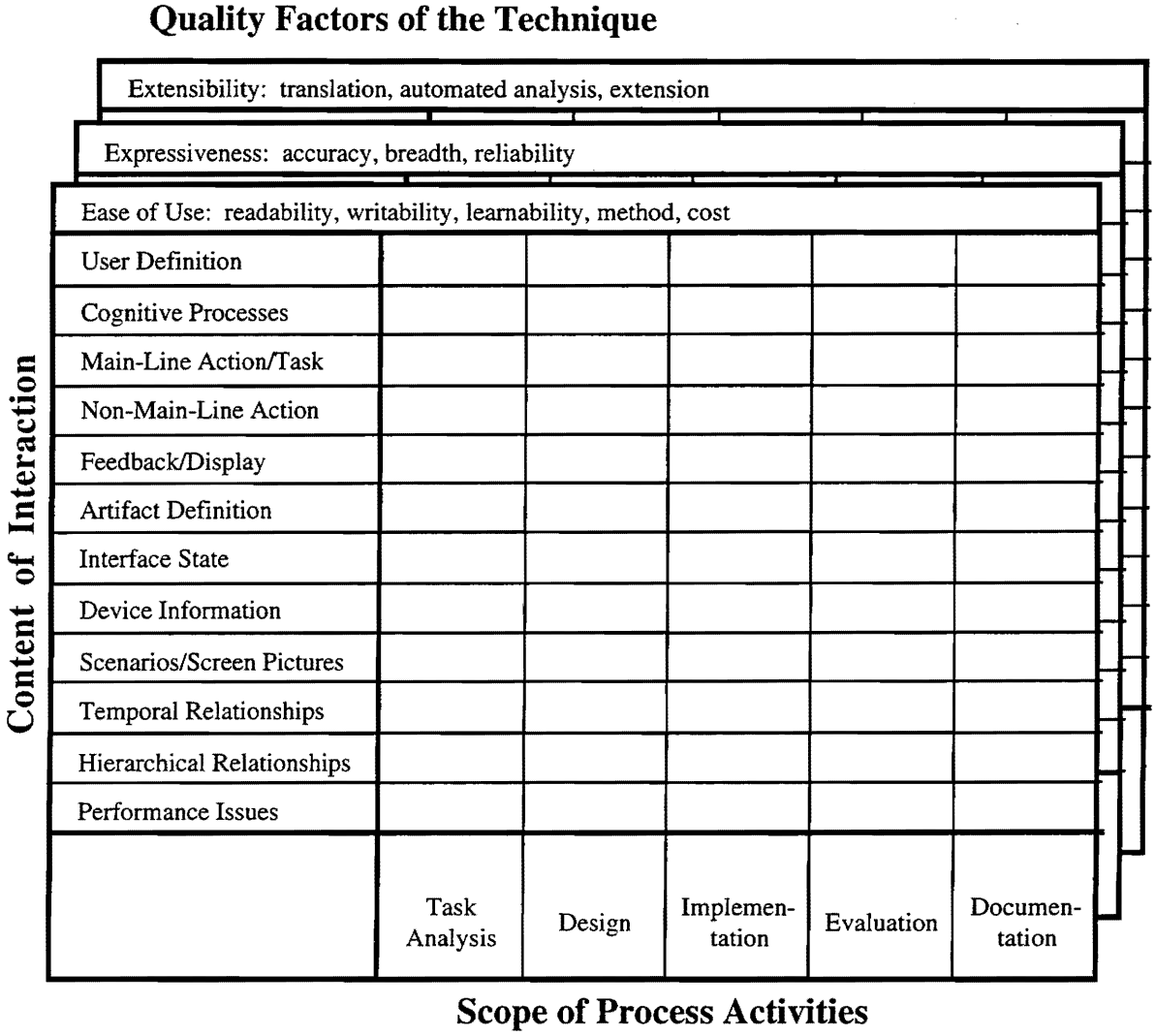


Figure 3.3-1. Taxonomic Model of Behavioral Representation Techniques

Each of the three dimensions of the taxonomic model are further detailed in the following sections.

3.3.1 Scope of Process Activities

We consider the *interaction development process*, the process of developing the content, appearance, and behavior of a user interface, to incorporate a wide variety of activities and roles, beginning at the recognition of a user need for a system to accomplish a specific task and ending at the time that an acceptable system is delivered to the user or customer. Johnson and Johnson (Johnson, 1988) describe activities of *analysis*, *specification*, *development*, and *evaluation* ; Hartson and Hix describe *task/functional/user analyses*, *requirements specification*, *conceptual/formal/detailed design*, *rapid prototyping*, *software production*, *evaluation*, and *deployment* (Hartson, et al., 1989; Hix, et al., 1993). We combined these two views with our observations of and conversations with interactive system developers currently using the UAN to yield five general activities within the interaction development process:

- *Analysis* (including task/functional/user analyses)
- *Design* (including conceptual/formal/detailed and redesign)
- *Implementation* (user interface software design and programming)
- *Evaluation* (formative evaluation, performance and acceptance testing)
- *Documentation*

We have observed that in some cases the documentation specialist is a key member of the interface design and development team. Further, documentation is certainly a part of a system with which a user interacts. Therefore, we include documentation development as a component of interaction development.

Development of interactive systems begins with analysis, and *task analysis* is one of the most important components of this process.

Task analysis is generally defined as the activity of decomposing a user's real-world tasks into ordered sets of subtasks (Olsen, 1988). The UAN, since it is a task-based representation technique and is particularly adept at representing the temporal and hierarchical relationships within and among tasks, provides an excellent starting place for capturing the results of task analysis. However, the UAN was not originally introduced to client sites as anything other than an interface design representation technique. Despite our own experiences that showed the UAN was useful in the task analysis process, early UAN documentation did not suggest this use. Thus, we must admit to being somewhat surprised when early contact with UAN users indicated task analysis as being the starting point for UAN use. Some UAN users discovered on their own that the UAN was very adept at recording task analysis information. Examinations of other behavioral representation techniques suggested that many of them might also be applied in task analysis, especially those such as GOMS, CLG, TAG, etc., which build decomposition-based task hierarchies. However, most of them are lacking in any ability to represent the ordering or temporal relationships among tasks. It is important to note that there are techniques which are not considered behavioral representation techniques that specialize in task analysis (e.g., TAKD (Johnson, 1991)).

Use of the word *design* within the taxonomy refers to interaction design as opposed to interface software design. Users of the UAN reported that the UAN was very useful to them in interaction design. This process of interaction design consists of the conceptual, formal, and detailed design and redesign of the interface, not the interface software. Other formal representation techniques, such as CLG, were, in fact, created for interaction design.

Design and development of the interface software, be it prototype or final version, falls under the heading of *implementation*. Any of the techniques such as UAN or CLG which provide a detailed description

of the expected behavior of the interface are useful for communicating the design to the implementers. We witnessed this communication at several UAN sites, although at one site, it was interesting to note that the design was first translated from the UAN into a more traditional state-transition diagram, constructional notation.

Most of the representation techniques listed in Chapter 2 are techniques for prediction and evaluation of user performance with a given interface. This process, as well as the processes of formative evaluation and acceptance testing, fall under the general heading of *evaluation*. At one UAN site, site 1 described earlier, an analyst sat with UAN task descriptions and a prototype interface for days checking the behavior of the interface against the predicted behavior provided by the UAN task descriptions. This evaluation led to many revisions for the implementers that had strayed from the design.

We must admit to having been somewhat surprised by one of the applications of the UAN in the interface development process. While we had speculated for some time that the UAN might provide a substantial benefit in the development of user *documentation*, we were delighted to see that UAN users began using it in this area on their own and indeed found it useful (Chase, Peretti, Hartson, & Hix, 1993). While examination of the literature regarding other behavioral representation techniques did not produce any evidence of their use in this way, some of them provide many of the same resources that proved to be useful in the UAN (e.g., hierarchical relationship structures, task-based descriptions). However, most of these representation techniques are lacking any representation of temporal relationships, interface feedback, and state information. All these components were found to be key ingredients of the documentation process using the UAN.

It is apparent that once we have drawn these lines of demarcation

between activities and defined these terms, the categories on the Scope axis are discrete.

3.3.2 Content of Interaction

Many of the techniques referred to here as behavioral representation techniques are actually referred to by their creators as user models. As such, many of them include representations for various attributes of the user (e.g., physical, cognitive, etc.) which will come to bear on the interaction. For this reason, a *user definition* is considered a key component of behavioral representation techniques. This definition may include, but is not limited to, physical attributes (e.g., limitations of movement), cognitive abilities (e.g., limitations of memory or attention), experience level (e.g., application specific experience), and knowledge (e.g., task domain specific knowledge). A user definition of this nature has only recently become an issue for the UAN as the technique as been applied in interface experiments involving people with physical disabilities (Chase, Casali, & Hartson, 1992).

One of the primary functions of many of the behavioral representation techniques, such as GOMS and TAG, is the idea of predicting and/or recording the user's mental workload. The purpose of this activity is to recognize tasks within an interaction that require the user to keep track of too much information and therefore degrading performance. For this reason, *cognitive processes* are a component which behavioral representation techniques could or should represent. In fact, the UAN has been augmented with memory cognition action (MCA) tables in order to record the user's mental workload (Sharratt, 1990). In this case, rather than merely adding new symbols to the UAN, new columns were added to record user memory requirements and cognition requirements. User memory requirements are broken down into the following tasks:

- recall LTM—recall from long term memory
- retain WM—store in working memory
- recall WM—recall from working memory
- forget WM—erase from working memory

These tasks are then used to create a memory column. The cognition column is annotated with pseudo-code. The user action column, which is the connector to the basic UAN, may contain either UAN task names referring to previously defined macros or explicit UAN notation. Figure 3.3-2, taken from Sharratt(Sharratt, 1990). shows an example of the memory, cognition, and action columns for finding an application on the Macintosh™ desktop.

TASK: Find an Application on the Macintosh™ Desktop		
MEMORY	**COGNITION**	**ACTION**
recall LTM(AN;HDI)		
retain WM(AN;HDI)		
recall WM(HDI)		
	Look at (desktop) for (HDI)	
		~[(result of look)]
		Mv^v^
forget WM(HDI)		
recall WM(AN)		
	Look at (window) for (AN)	
	Decide:	
	IF (AN present) THEN (end task) END	
recall LTM (scroll bar &		
elevator)		
	Look at (window) for (scroll bar & elevator)	
	Decide:	
	IF (elevator present and not at bottom of scroll bar) THEN	
retain WM(elevator		
position)		task:Scroll Window
(go to Recall WM(AN))	ELSE	task:SearchFolders
	END	
End Task		

Key: AN - Application Name HDI - Hard Disk Icon
WM - Working Memory LTM - Long Term Memory

Figure 3.3-2. Memory, Cognition, and Action Columns of UAN

In this way, the UAN can provide not only a description of tasks the user performs, but also a description of the cognitive processes and memory loading involved in performing those tasks. This information can be very useful in designing user documentation and online help systems.

Any technique for behaviorally recording the design of an interactive system must at the very least describe the behavior or actions of the user. All the techniques listed in Chapter 2 have the capability to represent user actions. Many of them are limited to recording only error-free user actions. Therefore, error-free user actions or *main-line user tasks* (i.e., descriptions of the most direct way to perform the task without excursion or detour) are one component of behavioral representation techniques. However, not all user actions are error-free or main-line. Users make mistakes and users change their minds in the middle of tasks. For this reason, UAN users have asked for a way of representing erroneous behavior, intention shifts, and other non-main-line user actions. Therefore *non-main-line user tasks* are also a component that behavioral representation techniques could or should represent.

One of the components of interaction that the UAN represents that most of the other representation techniques do not is that of *interface feedback/display*. This component includes any perceivable response by the system to a user action as well as perceivable independent system actions. While interface feedback has long been a part of the UAN, UAN users at a site working with virtual reality systems reported that many of the highly interactive systems that they are now producing generate independent system activity which the user may monitor, interrupt, or ignore. Therefore this component includes both feedback and display.

As interactive system developers began using object-oriented approaches to software development such as C++, Smalltalk, etc., one of the recurring complaints was that they would like for their interface design representations to be object-oriented as well in order to cut down on the translation from behavioral to constructional domains. The need for *artifact definitions* was registered several times by UAN users. These object-oriented

definitions were not to replace the UAN task-based descriptions but rather to augment them to aid in translation to the constructional domain. Further, the Task Artifact Framework (TAF) described in Chapter 2 already provides a mechanism for recording a behavioral description of an interface including artifact definitions. For these two reasons, artifact definitions are included as a component of behavioral representation techniques. Of course, constructional representation techniques have always been strong in the area of artifact definitions. However, a full representation of artifact behavior from an object-oriented view is not usually a task-oriented description.

Another component of interaction that the UAN represents, and that is lacking in many other techniques, is that of *interface state.* Many UAN users have reported that they use this information to connect behavioral descriptions of the UAN to constructional representations like state-transition diagrams. For this reason, interface state is included as a component of behavioral representation techniques.

As stated in Chapter 2, there has been much recent work in human-computer interaction in the area of device modeling. This area is becoming increasingly critical as a broad array of new and unique input devices are being introduced. One UAN user working with users with physical disabilities reported the need to record *device information* as part of the interaction description in order to keep track of whether or not an individual with a given disability could manipulate a given device to accomplish the necessary actions. This would mean that this information must include a description of the device as well as a description of the physical actions that typically manipulate the device. Further, this device information might also need to include a description of the functions or actions available with a given device. None of the behavioral representation techniques in this study currently provides a means of recording this information. However, the device models listed earlier (Mackinlay,

et al., 1990; Bleser, 1991) do provide a means of representing this type of information. Providing a representation for device information is becoming an area of greater importance with the introduction of more unusual devices such as virtual reality. Therefore, device information is included as a component of behavioral representation techniques.

One of the earliest forms of behavioral representations of interfaces was the use of descriptions of possible user activity associated with screen pictures. This method, often called scenarios or story boarding, is very good for getting an early picture of the interface. In fact, the Task Artifact Framework (TAF) described in Chapter 2 takes this technique and builds on it to make it a viable technique throughout the interaction development process. One of the many recurring themes in critical incidents collected from UAN users was the need for a way to associate task descriptions directly with screen pictures in order to provide context. Therefore, *scenarios and screen pictures* are considered a component of behavioral representation techniques.

With today's highly interactive systems, it is no longer sufficient to represent an interface as merely a sequence of user actions as do many of the techniques listed Chapter 2. Today's interfaces are highly concurrent, highly interleaved, and very rarely only sequential. The UAN provides a great deal of attention to temporal relationships among tasks and yet quite a number of the critical incidents reported by UAN users had to do with temporal behavior that was difficult or impossible to represent. For these reasons, *temporal relationships* are included as a component of behavioral representation techniques.

Many existing behavioral representation techniques have notational mechanisms for representing the *hierarchical relationships* among user tasks (e.g., GOMS, TAG, CLG, UAN). While these hierarchical

relationships are important throughout the development process, one of the more interesting critical incidents associated with these relationships came from the documentation activity where representation of these relationships in the UAN proved very beneficial to the creation of end-user documentation (Chase, et al., 1993). For these reasons, hierarchical relationships are included as a component of behavioral representation techniques.

One of the primary goals of the human-computer interaction research community is to increase user productivity by improving user performance (i.e., time, learning, error-rate, etc.). Many of the techniques listed in Chapter 2 provide a mechanism for recording information about the predicted and/or actual user performance. While the UAN was not originally intended to address user performance prediction, UAN users continue to report new ways of using the UAN for this purpose (Chase, et al., 1992). Therefore, *performance issues* are included as a component of behavioral representation techniques.

As with the categories on the *scope of process activities* axis, the components on the *content of interaction* axis are relatively discrete.

3.3.3 Quality Factors of the Technique

Unlike the *scope of process activities* and *content of interaction* axes, the *quality factors of the technique* axis did not seem to produce naturally discrete categories. In fact, very early in the empirical work, it looked as if this axis might need to be split into two dimensions or even dropped altogether from the taxonomy if discrete categories could not be found. However, over time and with the increased input from the analysis of other behavioral representation techniques, it became apparent that everything on the *quality factors of the technique* axis fit reasonably well into one of three discrete categories:

- *ease of use*—those things that facilitate use of a given technique such as readability, writability, learnability, methods, and cost;

- *expressiveness*—those things that tend to support a correct representation such as accuracy, breadth, and reliability, (this term might better be represented by the word correctness); and

- *extensibility*—balance between strict formalisms which can be easily translated and analyzed with open notations that are easily extended to encompass new ideas.

The idea of facilitating use is one that was drawn primarily from empirical work with UAN users. Early on, these users complained repeatedly about the difficulty of learning the terse symbolic notation of the UAN as well as the lack of a method for applying the UAN in the various activities of interaction development. However, as we released new documentation which included method information as well as hints for getting started using explanatory notes, we began to get positive critical incident reports about increases in productivity due to the improved readability, learnability, and methodological development in the new version. For example, in the experiments testing the improvement in the UAN, described in Chapter 5, the group using the new version of the UAN finished their task in slightly more than half the time it took the group using the old version of the UAN. The group members attributed their success to the task (i.e. interface development task) oriented nature of the new version of the UAN.

Two basic complaints initiated the category of expressiveness: "This symbol doesn't mean exactly what I want to say" (e.g., the idea of inclusive or—performing one or more of a set of tasks—was very

difficult to express using existing temporal notation in the original version of the UAN) and "There is no way to represent this interaction component with this notation" (e.g., trying to represent interface feedback or state information in one of the representations such as GOMS that does not represent these interaction components). These two comments spawned use of the attributes accuracy and breadth, alluding to the ability of a representation to reflect the interface or design exactly, and the ability to encompass everything that is in the interface design. Additionally, as analysis of other techniques continued, we began to encounter the repetitive criticism that given two designers/analysts, they might both accurately represent the behavior of interface with different task/goal structures. Therefore reliability, the idea that results using a representation technique are repeatable across users of that technique, became another important part of the expressiveness category.

One of the long sought-after goals interaction and interface designers is the idea of automated translation of the design into a prototype. As some behavioral representation techniques have gotten close to that idea, we have discovered a trade-off very similar to that described by Simon in his model (Simon, 1988). As a design representation gets more and more formal, as do the BNF grammar-based techniques, it may get very close to being automatically translatable into a prototype. However, this same formalism produces a notational complexity which makes the technique more difficult to use. Therefore, it is clear that there must be a balance between the goals of formalism for the purpose of automation and more flexible prose-like styles which facilitate use.

4. USING THE TAXONOMIC MODEL OF BEHAVIORAL REPRESENTATION TECHNIQUES

As described previously, the purpose of our taxonomy is to support development and evaluation of new behavioral representation techniques, as well as analysis, extension, and comparison of existing techniques. Thus, this taxonomy supports two basic functions:

- *classifying:* mapping problems or critical incidents observed in the use of a specific behavioral representation technique to a missing or failing activity (from the Scope dimension), component (Content dimension), and characteristic (Quality dimension); and

- *rating:* analyzing an existing behavioral representation technique for its ability to cover all activities, components, and qualities identified in the taxonomy.

4.1 Classifying Critical Incidents

The purpose of the classification of critical incidents is to identify the missing or failing attributes of a behavioral representation technique which caused the critical incident. In this way, modifications or extensions can be identified for the behavioral representation technique to address very specific problems. Additionally, in the case of positive incidents, as described in Chapter 3, the mapping will identify the successful attributes of the representation technique.

As an example of this mapping and analysis, Table 4-1 provides a listing of five selected critical incidents reported for the UAN and how each might be mapped onto our model.

Table 4-1. Sample mapping of critical incidents

Critical Incident	Scope	Content	Quality
An individual attempting to record the cursor manipulation required to operate a given system found no notation available to represent the fact that this particular system required manipulation of two distinct cursors.	design	main-line user actions	expres-siveness
An individual attempting to record the cursor manipulation required to operate a given system found no notation available to represent the condition that the cursor must be within a particular icon before the user may continue.	design	main-line user actions	expres-siveness
An individual attempting to record the cursor manipulation required to operate a given system found no notation available to represent the idea that the user must move iconx until it overlaps with icon y.	design	main-line user actions	expres-siveness
An individual attempting to record information about a system requested a way to record information about the size, shape, color, etc. of objects, and icons on the screen.	design	feedback /display	expres-siveness
An individual attempting to record information about a system found no notation to represent the idea that some objects have multiple handles.	design	main-line user actions	expres-siveness

The examples in Table 4-1 are indicative of the critical incidents most commonly reported about the UAN. These incidents most often involve the activity of design, since that is the most common use for the UAN, they usually involve one of the four typical UAN content attributes, i.e. main-line user actions, feedback/display, state information, or temporal relationships, and they most commonly involve the quality of expressiveness or lack thereof.

The original mapping of an incident to the model is only the first step. Consider the first incident from Table 4-1 as an example. Its original mapping to the activity of design, the component of main-line user actions, and the quality of expressiveness is relatively easy

since it is an incident involving the lack of sufficient representation for some part of the design. However, in looking for a solution to this problem, it is useful to look across each of the rows in the model which intersect at this cell to see if this incident might be indicative of a larger problem. For example, for the *scope* dimension one could check whether any of the other activities along the *scope* dimension would encounter difficulty because something is missing within the notation. If so, then we need to address this problem across all of those activities. Likewise for the *content* dimension, we would check if any of the other components of the *content* dimension would be affected by this lack of sufficient representation.

The *quality* dimension is slightly different. Rather than checking if any of the other attributes of the *quality* dimension are affected by the original problem, we need to address each of the *quality* attributes when making a change to the notation. For example, the first incident in Table 4-1 was mapped to the expressiveness attribute of the *quality* dimension since the UAN was lacking a notation to represent multiple cursors. In adding a notation for multiple cursors, the other two attributes of the *quality* dimension, ease of use and extensibility, should be rechecked so that a new problem is not created while fixing an old one.

4.2 Rating Behavioral Representation Techniques

The purpose of rating behavioral representation techniques using our taxonomy is to identify the capabilities of each technique to represent all of the components, activities, and qualities described in the taxonomy. This rating can then be used in two ways:

- evaluation of the strengths and weaknesses of a particular behavioral representation technique

- comparison of behavioral representation techniques for a specific purpose.

The first of these, technique evaluation, can best be illustrated by looking at an example from the original UAN. Figure 4-1 shows a subset of a sample rating of the original UAN for expressiveness along the *scope* and *content* dimensions. A one in a cell means that the original UAN did not address that issue, a two in a cell means that the original UAN addressed that issue, but either not very well or not completely, and a three in a cell means that the original UAN did a relatively good job of addressing the corresponding issue.

Expressiveness					
User definition	1	1	1	1	1
Cognitive processes	1	1	1	1	1
Main-line user actions	2	3	3	2	2
Non-main-line user actions	1	1	1	1	1
Feedback/Display	2	3	3	2	2
Artifact definitions	1	1	1	1	1
Interface state	2	3	3	2	2
Device information	1	1	1	1	1
Scenarios/Screen pictures	2	2	2	2	2
Temporal relationships	2	2	2	2	2
Hierarchical relationships	1	2	2	1	1
Performance issues	1	1	1	1	1
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-1. Sample rating of the original UAN for expressiveness along the *scope* and *content* dimensions.

For example, the fact that the first row, i.e. user definition, is all ones suggests that the original UAN did not provide any representation for user definition. As the ratings suggest, the original UAN primarily provided representations for main-line user actions, feedback/display, interface state and temporal relationships. Further, these representations were generally for the purpose of communicating the design of the interaction component of software system from designers to implementers. This is illustrated by the difference between the ratings in the design and implementation columns and those in the other columns in the model.

Once the ratings of a particular technique have been completed, it is then possible to analyze the ratings to determine areas of possible modification and extension. For example, the original UAN provided only indirect representation of hierarchical relationships among user tasks. Therefore the ratings for hierarchical relationships are not particularly good. This suggests that this is an area of possible modification and extension for the UAN. However, that it may not be possible or even desirable for any behavioral representation technique to rate highly in every cell. Such a technique may be so broad as to be unusable.

When there are particular attributes which are important to a system designer, there is probably a technique which will provide excellent coverage of those attributes at the expense of attributes of lesser importance. Finding the technique which best suits the needs of the interaction designer is the second purpose for rating behavioral representation techniques using our taxonomy, i.e. comparison of behavioral representation techniques for a specific purpose. By way of example, Figure 4-2 shows the ratings for GOMS for the same subset of the model shown in Figure 4-1 for the original UAN, i.e. expressiveness along the *scope* and *content* dimensions.

Expressiveness					
User definition	1	1	1	1	1
Cognitive processes	2	2	2	2	2
Main-line user actions	2	2	2	2	2
Non-main-line user actions	1	1	1	1	1
Feedback/Display	1	1	1	1	1
Artifact definitions	1	1	1	1	1
Interface state	1	1	1	1	1
Device information	1	1	1	1	1
Scenarios/Screen pictures	1	1	1	1	1
Temporal relationships	1	1	1	1	1
Hierarchical relationships	2	2	2	2	2
Performance issues	2	2	2	3	2
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-2. Sample rating of GOMS for expressiveness along the *scope* and *content* dimensions.

Comparing the ratings for GOMS in Figure 4-2 with the ratings for the original UAN in Figure 4-1, it is clear that GOMS does not provide representation for feedback/display, interface state, and temporal relationships whereas the UAN does. Thus, if these were important attributes to an interaction designer, then the original UAN would be a better choice than GOMS. However, GOMS provides representation for performance prediction whereas the UAN does not making GOMS the better choice if performance prediction is the most important issue to an interaction designer. In this way, any two behavioral representation techniques can be compared over the subset of interest once the ratings have been completed. All of the model ratings of the original UAN, GOMS, and the new UAN, shown in this chapter were compiled by the author working as a single analyst.

The evaluation of the reliability of the model as discussed in Chapter 5 suggests that any group of analysts using the model to rate these three techniques would generate similar results. It is important, however, to caution a user of the taxonomy against taking an average of the ratings across the entire model and then saying that the technique with the higher rating is "better". There is no research to suggest that such a conclusion would be valid or even useful.

4.3 Modifying and Extending the UAN

The modification and extension of the UAN required the use of both of the functions of the taxonomy, i.e. classification of critical incidents and rating of the UAN. Critical incidents alone would not have suggested many extensions to the UAN since the majority of the critical incidents reported from UAN users mapped to cells where the UAN already rated highly. There are two explanations for this occurence. First, the majority of UAN users were using the UAN for its original purpose, i.e. the recording and communication of the design of the interaction component of a software system. Second, they limited themselves to the *content* components which the UAN represented, i.e. main-line user actions, feedback/display, interface state and temporal relationships. In cells where the original UAN rated highly, the critical incidents helped identify areas of possible improvement. Thus, a rating of three in a particular cell does not mean that there is not room for improvement.

4.3.1 Rating the Original UAN

Figure 4-1 above shows the rating of the original UAN for the subset of the model showing the intersection of the expressiveness attribute of the *quality* dimension with the *scope* and *content* dimensions. Figures 4-3 and 4-4 show the rest of the taxonomy ratings for the original UAN.

Ease of Use					
User definition	1	1	1	1	1
Cognitive processes	1	1	1	1	1
Main-line user actions	1	2	2	1	1
Non-main-line user actions	1	1	1	1	1
Feedback/Display	1	2	2	1	1
Artifact definitions	1	1	1	1	1
Interface state	1	2	2	1	1
Device information	1	1	1	1	1
Scenarios/Screen pictures	1	2	2	1	1
Temporal relationships	1	2	2	1	1
Hierarchical relationships	1	1	1	1	1
Performance issues	1	1	1	1	1
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-3. Rating of the original UAN for ease of use along the *scope* and *content* dimensions.

Extensibility					
User definition	1	1	1	1	1
Cognitive processes	1	1	1	1	1
Main-line user actions	2	2	2	2	2
Non-main-line user actions	1	1	1	1	1
Feedback/Display	2	2	2	2	2
Artifact definitions	1	1	1	1	1
Interface state	2	2	2	2	2
Device information	1	1	1	1	1
Scenarios/Screen pictures	2	2	2	2	2
Temporal relationships	2	2	2	2	2
Hierarchical relationships	1	2	2	1	1
Performance issues	1	1	1	1	1
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-4. Rating of the original UAN for extensibility along the *scope* and *content* dimensions.

Looking at the ratings of the original UAN for ease of use in Figure 4-3, the ratings are substantially lower than those for expressiveness in Figure 4-1. This is due to two factors. First, the original UAN provided no prescribed method of use which left many UAN users confused as to how to begin using the technique. In fact, the only examples of UAN use involved the activities of design and implementation meaning that the relatively few users attempting to use the UAN for task analysis, evaluation, and documentation were completely on their own. Second, the UAN is, by design, a terse, symbolic notation which leaves may beginning users struggling to learn the notation while also having to invent their own method of using the UAN.

Comparing the ratings of the original UAN for extensibility, i.e. Figure 4-4, with the ratings of the original UAN for expressiveness, i.e. Figure 4-1, again we see that the ratings for extensibility are slightly lower than those for expressiveness. This is due to the fact that the quality of extensibility includes the ideas of automated translation and tool support. The original UAN is notationally very informal. In fact, there are as many different styles and unique symbols in the UAN as there are users of the UAN. This informality, while a strength in terms of flexibility, is definitely a weakness with regard to automated translation and tool support.

4.3.2 Making Changes to the UAN

With ratings in hand, the next question to be answered was where and how to extend the UAN. The model ratings suggested several areas, e.g. user definitions, performance prediction, hierarchical relationships, artifact definitions, etc., in which the original UAN could be extended. However, as stated earlier, it is probably not necessary or even desirable for a single representation technique to cover all of the attributes in the taxonomy.

In terms of potential improvement in the UAN, we needed to focus on those areas where we could improve a capability that the UAN already provided or add a capability which would strongly complement the existing UAN. The first of these changes was to prescribe a method of use for the UAN for each of the five activities, i.e. task analysis, design implementation, evaluation, and documentation. This method is described in the new UAN Tutorial included as Appendix B. We derived the method of use of the UAN for each activity by examining the various methods of use employed by each of the UAN client sites. This change was intended to help alleviate some of the problems noted in the ease of use ratings for the original UAN.

A second change intended to help the ease of use, or more particularly, the learnability, of the UAN, was the adoption of an explanatory notes column. Figure 4-5 shows an example taken from the UAN Tutorial in Appendix B. The purpose of this column is to provide a prose description of the design to assist beginning UAN users who may not have mastered the symbolic notation.

Task: manage calendar		
User Action	**Feedback**	**Explanatory Notes**
OR(add appointment, view appointment, modify appointment, delete appointment, set alarm)*		The user may add an appointment, view an appointment, modify an appointment, delete an appointment, or set an alarm and may choose from this set zero or more times.

Figure 4-5. Highest level of abstraction for calendar management system

Another change to the UAN which resulted from the taxonomy ratings was the adoption of an explicit hierarchical task structure representation. For example, in Figure 4-5 we see that the **add appointment** task is a descendent of the **manage calendar** task. We know this implicitly because the **add appointment** task is listed within the **manage calendar** task. However, this implicit representation of the task hierarchy creates some navigational problems in that a UAN user can see only one level of the hierarchy at a time and therefore will have difficulty traversing the task structure. Figure 4-6 shows the hierarchical representation of the **manage calendar** task from Figure 4-5. As each of the tasks in the hierarchy are expanded, the hierarchy expands until the UAN user has a complete hierarchical representation of the task structure to accompany the UAN symbolic representation. This use of hierarchical representation is further discussed in the new UAN tutorial in Appendix B.

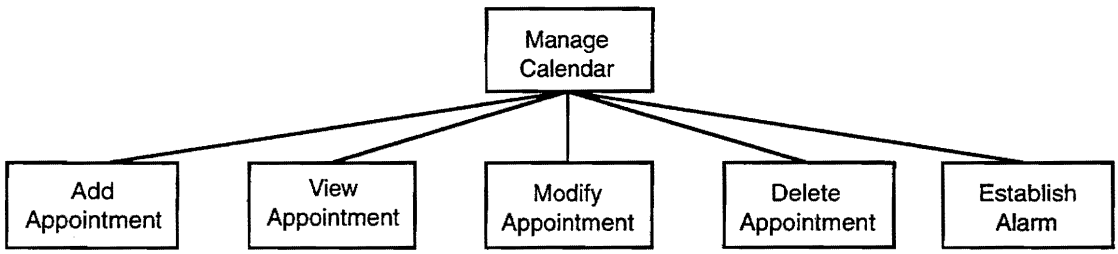


Figure 4-6. Highest level of hierarchy for calendar management system

Scenarios or screen pictures have long been accepted as a complimentary representation with the UAN. However, in looking at the ratings for the UAN and the use or lack of use of scenarios among UAN users, we decided to make the use of screen pictures or scenarios in the UAN more formal. This is especially useful with the changing face of graphical user interfaces in that it is sometimes more effective to draw an object than to describe it. The use of scenarios in the new UAN is fully described in the new UAN tutorial in Appendix B.

One of the reasons that the UAN has been somewhat difficult to learn for novices is the level of detail required to represent the various behaviors of graphical objects when acted on by a user. This problem, coupled with a desire to connect the UAN to object oriented development techniques and languages, prompted us to examine the idea of including artifact definitions in the UAN. In this way, we could localize the description of the behavior of types and classes of objects rather than having to describe that behavior in every task which affects any object of that type.

If an example can be taken from object oriented programming, it is apparent that certain groups or families of artifacts exhibit common behaviors and attributes. With this in mind, it makes sense to use these common elements to abstract away as much behavior as

possible in order to simplify task descriptions. As an example, take the standard description of selecting a file_icon shown in Figure 4-6:

Task: Select(file_icon')		
User Action	**Feedback**	**Interface State**
~[file_icon']Mv M^	file_icon'-!:file_icon'! file_icon!:file_icon-!	selected = file

Figure 4-6. UAN description of the select(file icon) task

This example illustrates the use of the feedback column to describe the behavior of other artifacts in the same group or family as the file_icon being selected. In this example, file icons are described as mutually exclusive for the select task meaning that all other file icons will be unselected and unhighlighted when a new file icon is selected. If a group definition were used to describe this behavior, it might look like this:

Declaration of Class: File_icon
...
Mutually Exclusive under single select
...

The use of this declaration would simplify the above task down to the task shown in Figure 4-7:

Task: Select(file_icon')		
User Action	**Feedback**	**Interface State**
~[file_icon']Mv M^	file_icon'-!:file_icon'!	selected = file

Figure 4-7. UAN description of the select(file icon) task

Then adding a definition of the highlighting to the class definition would simplify the task description even further as shown in Figure 4-8:

Declaration of Class: File_icon

...

Mutually Exclusive under single select

...

! means bold, no effect if already !

Task: Select(file_icon')		
User Action	**Feedback**	**Interface State**
~[file_icon']Mv M^	file_icon'!	selected = file

Figure 4-8. UAN description of the select(file icon) task

Declarations of this type do not initially appear to belong in the behavioral domain. However, where behavior, as described above, is uniform across a set or class of artifacts, these descriptions do indeed address the behavior of the interface. While this is undoubtedly a simplistic example, it gives great evidence of the economy to be gained through the use of artifact, group, and class definitions. This economy may lead to advantages of scalability which has long been a barrier to complete behavioral descriptions of interfaces.

With individual definitions for each artifact type, there exists the possibility that the standard symbols such as !, >, >>, [], etc. may become functionally overloaded in that they may be defined to mean different things for different artifacts. This may be a problem or it may truly be a feature. It will remain for user testing to determine the advantages or disadvantages of this approach. Artifact definitions in the UAN are more fully discussed in the new UAN tutorial in Appendix B.

Another component of the *content* axis that the original UAN did not address is non-main-line user actions. This category refers to actions that the user may take that are not part of a main-line user task or

that may terminate a main-line user task without completing it.

Many of the behavioral representation techniques are limited to error-free or main-line user performance. Certainly, this approach can give a designer the ability to describe how the interface should work for an ideal user under ideal conditions. However, as most anyone who has ever designed and implemented a system can attest, a large portion of the software devoted to the interface will be devoted to handling non-ideal situations.

This leads to somewhat of a conflict of interest. On the one hand, we would like to represent the way that the design should work in the most straightforward, cost-effective way, yet at the same time, we need to represent the way that the interface will work to handle error-conditions, or non-main-line user actions, in very explicit detail. The UAN addresses this conflict by providing two views. The primary view will be that of the error-free, ideal path through the interaction. The secondary view will be a much more detailed description of the same task providing for the possible error conditions. Certainly, most designers will not feel it necessary to write the detailed secondary view for every task. There will be similarities between tasks so that it should be sufficient for a designer to write a detailed secondary view of a representative subset of tasks. As tool support continues to develop for the UAN, it should be possible to switch back and forth between views on the screen as needed. An example of two views in the UAN is shown in Figure 4-9. This example, the open file task from Microsoft Word on the Macintosh, is a relatively small task. However, it shows how a task can balloon in size when all of the possible paths are considered. The primary view shows the expert, error-free performance of the main-line task. The expanded view shows the same task with the user having the ability to quit. For larger tasks, this in-line description of alternatives could get confusing. Therefore, the secondary view is provided with the alternative actions listed in a

separate column. All of the actions which terminate the task are indicated by a double line under the action.

Task: Open File -- primary view

User Action	Feedback	Interface State
Select File		
Open selected file		

Task: Open File -- expanded view

User Action	Feedback	Interface State
Select File or select(cancel)		
Open selected file or select(cancel)		

Task: Open File -- secondary view

User Action	Alternative View	Feedback	Interface State
	select (cancel)		
Select File			
	select(cancel)		
Open selected file			

Figure 4-9. Two views of the UAN description of the open file task.

The changes in the UAN discussed thus far were the result of the rating of the original UAN in the taxonomic model of behavioral representation techniques. However, other changes were made as a direct result of the receipt and mapping of critical incidents reported by UAN users. For example, the ability of the UAN to represent the temporal relationships between user tasks has long been one of its strengths, e.g. Figures 4-1 and 4-2 comparing UAN and GOMS. However, several critical incidents were reported mapping to the

intersection of the activity of design, the component of temporal relationships, and the quality of expressiveness. These incidents led to a complete redesign of the notation within the UAN to represent the temporal relationships between user tasks.

Figure 4-10 provides a summary of all of the ways that tasks can be temporally combined in the UAN through either overlapping or combination.

Task Overlapping

None	A single task: A	A single task: A	A single task: A
Sequenced	Two or more tasks written as AND(A, B)	Two or more tasks written as INCOR(A, B)	Two or more tasks written as OR(A, B)
Order Independent	Two or more tasks written as IND/AND(A, B)	Two or more tasks written as IND/INCOR(A, B)	Two or more tasks written as IND/OR(A, B)
Interleaved	Two or more tasks written as INTER/AND(A, B)	Two or more tasks written as INTER/INCOR(A, B)	Two or more tasks written as INTER/OR(A, B)
Concurrent	Two or more tasks written as CONC/AND(A, B)	Two or more tasks written as CONC/INCOR(A, B)	Two or more tasks written as CONC/OR(A, B)
	And	Inclusive Or	Exclusive Or

Task Combination

Figure 4-10. Temporal Notation Combinations in the UAN

Figure 4-10 shows that the new UAN provides for tasks to be overlapped in one of five ways:

- None— which is the degenerate case of only one task,
- Sequenced— meaning that the tasks will not be overlapped but must be performed in a particular order,
- Order Independent— meaning that the tasks will not be overlapped but may be performed in any order,
- Interleaved— meaning that the user may alternate between tasks,
- Concurrent— meaning that the user may perform all tasks simultaneously,

and combined in one of three ways:

- And— meaning that the user must perform all tasks,
- Inclusive Or— meaning that the user must perform at least one of the tasks but may perform any or all of them,
- Exclusive Or— meaning that the user must perform one and only one task.

Thus, for example, a task description written as INTER/AND(A,B) means that the user may alternate between tasks A and B but must eventually complete both tasks. Again, this topic is further discussed in the new UAN tutorial in Appendix B.

A slightly more advanced topic which arose from critical incident reports was the idea of temporal and causal relationships within a UAN task description. The UAN is temporally ordered top to bottom. This means that any action A, system or user, which is described on a line above another action B, occurs before that action unless temporal

notation is used to specify otherwise. For example, if a task X was described as shown in Figure 4-11, then action A must precede action B.

Task: X		
User Action	**Feedback**	**Interface State**
A B		

Figure 4-11 UAN description of Task X

However, the temporal relationships get a little more complicated as you add in feedback and state information. The relationship between user actions on one hand, and feedback and interface state on the other, on a given line is not strongly temporal; it is primarily causal as shown in Figure 4-12. The designer is specifying that the user action is the cause of the feedback and the change in interface state. This, of course, implies the minimal temporal relationship that feedback and interface state changes cannot occur before a user action listed on the same line. This means that they may either be concurrent with the action (e.g. overlapping it in time as in the case of an icon following cursor movement in a draggin action) or sequenced following the action (e.g. as in the case of highlighting to indicate selection). If the temporal relationship or the time delay is important to the interface designer then it should be specified explicitly by indicating a time delay of $t > n$ in either the feedback or interface state columns or both.

Further, there normally is no temporal relationship between feedback and interface state in the standard UAN notation. There is, however, usually a semantic mapping or binding between the two, e.g. file-icon == file. If the designer has a particular interest in the temporal relationship between feedback and interface state changes, it should be specified explicitly.

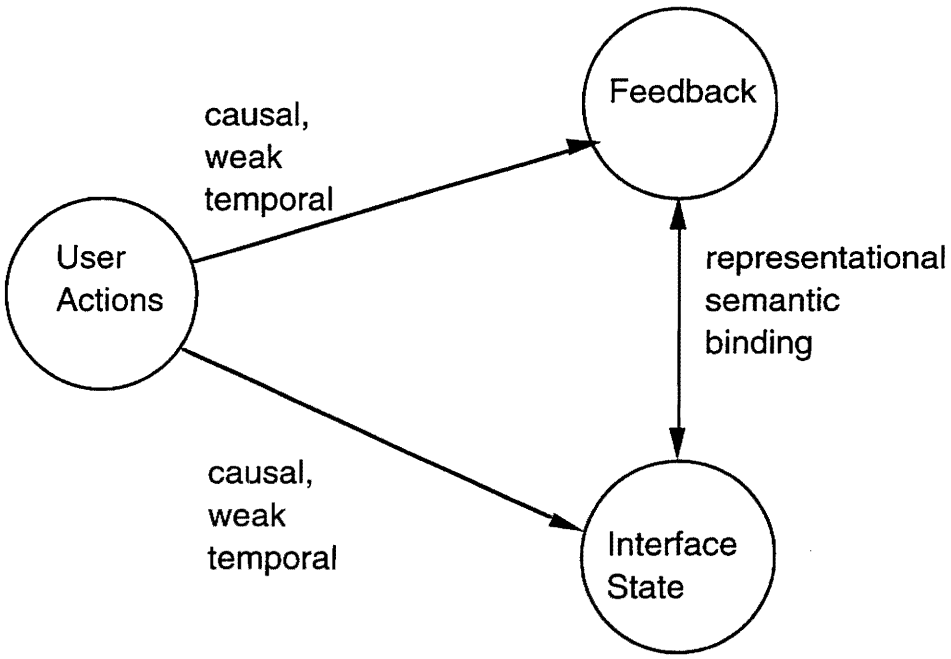


Figure 4-12 Relationships between actions, feedback, and state.

Many of the changes made to the UAN were relatively small changes made to address critical incidents dealing with ease of use and expressiveness. For example, several UAN users reported a desire to use notation to represent iterative looping. Specifically, program lanuage constructs such as do until or while do were added to the UAN.

One user reported that the naming of variables in the original UAN was contrary to the standards in other disciplines. Specifically, the original UAN used x to mean a specific instance of x and x' to mean all x in general. This usage was counter to the accepted use of the prime (') symbol in other disciplines. Thus the use of the prime (') symbol in the UAN has been swapped so that prime (') now means a specific instance. This should facilitate learnability for those UAN users that are already familiar with the use of the prime (') symbol

from other disciplines.

4.3.3 Rating the New UAN

Once the changes to the UAN had been made and documented (Appendix B), the next step was to rate the new version of the UAN using the taxonomy. Figures 4-13, 4-14, and 4-15 show the model ratings of the new UAN. As previously stated, all of the model ratings of the original UAN, GOMS, and the new UAN, shown in this chapter were compiled by the author working as a single analyst. The evaluation of the reliability of the model as discussed in Chapter 5 suggests that any group of analysts using the model to rate these three techniques would generate similar results.

Expressiveness					
User definition	1	1	1	1	1
Cognitive processes	1	1	1	1	1
Main-line user actions	3	3	3	3	3
Non-main-line user actions	3	3	3	3	3
Feedback/Display	3	3	3	3	3
Artifact definitions	2	2	2	2	2
Interface state	3	3	3	3	3
Device information	1	1	1	1	1
Scenarios/Screen pictures	3	3	3	3	3
Temporal relationships	3	3	3	3	3
Hierarchical relationships	3	3	3	3	3
Performance issues	1	1	1	1	1
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-13. Rating of the new UAN for expressiveness along the *scope* and *content* dimensions.

Ease of Use					
User definition	1	1	1	1	1
Cognitive processes	1	1	1	1	1
Main-line user actions	3	3	3	3	3
Non-main-line user actions	3	3	3	3	3
Feedback/Display	3	3	3	3	3
Artifact definitions	2	2	2	2	2
Interface state	3	3	3	3	3
Device information	1	1	1	1	1
Scenarios/Screen pictures	3	3	3	3	3
Temporal relationships	3	3	3	3	3
Hierarchical relationships	3	3	3	3	3
Performance issues	1	1	1	1	1
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-14. Rating of the new UAN for ease of use along the *scope* and *content* dimensions.

Extensibility					
User definition	1	1	1	1	1
Cognitive processes	1	1	1	1	1
Main-line user actions	2	2	2	2	2
Non-main-line user actions	2	2	2	2	2
Feedback/Display	2	2	2	2	2
Artifact definitions	2	2	2	2	2
Interface state	2	2	2	2	2
Device information	1	1	1	1	1
Scenarios/Screen pictures	2	2	2	2	2
Temporal relationships	2	2	2	2	2
Hierarchical relationships	2	2	2	2	2
Performance issues	1	1	1	1	1
	Task Analysis	Design	Imple-mentation	Evaluation	Documen-tation

Figure 4-15. Rating of the new UAN for extensibility along the *scope* and *content* dimensions.

Again, while it is tempting to take an average across all of the cells in the model and calculate a percentage of improvement, that is not the purpose of the model and such a number taken from the ratings of a single analyst would be misleading. The experiments showing the improvement in the new version of the UAN are discussed in Chapter 5.

Comparing these ratings for the new UAN to those for the original UAN, it is apparent that there has been substantial improvement, i.e. higher ratings, in a large number of cells. Most all of this improvement can be attributed to two sources. First, the new UAN addresses three components of the content axis which were not addressed in the old version, i.e. hierarchical relationships, artifact

definitions, and non-main-line user actions. Further the new version greatly expands the role of scenarios in the interaction development process. Second, the new UAN is documented with a prescribed method of use, including examples, for each of the five activities on the *scope* dimension. The ratings of the new UAN handling of artifact definitions are listed as twos instead of the maximum of threes since part of the purpose of these definitions is to provide a link to object oriented development and no testing has as yet been done to test the effectiveness of the new artifact definitions for this purpose. Chapter 5 discusses the evaluation of the new UAN and the evaluation of the model in general.

5. EVALUATING THE TAXONOMIC MODEL OF BEHAVIORAL REPRESENTATION TECHNIQUES

The final and perhaps most important step in the epistemological cycle is evaluation. Without evaluation, our proposed model would be similar to Simon's model in that it would have to be treated as speculation, not theory. Our taxonomy was evaluated over a variety of criteria. The first criterion we examined was that of *completeness*, defined to be the taxonomy's coverage of all aspects of behavioral representation techniques. Developers of the device models mentioned earlier (Mackinlay, et al., 1990; Bleser, 1991) found it virtually impossible to test completeness of their model. We, too, found this to be the case in evaluating our taxonomy's coverage of behavioral representation techniques. However, we postulate that our taxonomy should provide satisfactory coverage of at least existing techniques since it was derived from an analysis of them. There is little doubt that as new interaction styles and representation techniques are developed, our taxonomic model will need to be extended.

Evaluation of the device models centered on the criterion of *utility* (Mackinlay, et al., 1990; Bleser, 1991), a measure of whether the model provides some useful function in working with input devices. However, another important criterion is that of *reliability*, a measure of whether the model produces similar results across those who use it to compare and evaluate behavioral representation techniques. The taxonomy of behavioral representation techniques was evaluated under these two criteria, *utility* because it is the standard to which other similar models have been held and *reliability* because it was the single most common criticism of other models in the literature, as described in the following sections.

5.1 Demonstrating Reliability of the Taxonomic Model

Our taxonomy was evaluated for *reliability* by having a group of interface designers perform both of the tasks described in Chapter 4, *classifying* and *rating*, and then performing a statistical test of the similarity of their results. In the first task, *classifying*, we selected 40 critical incidents commonly reported by UAN users and had five interface designers map these incidents into the taxonomy. For purposes of this mapping, we defined critical incidents to be:

- an encounter with an interface component that the UAN did not represent,
- a difficulty using the existing UAN notation,
- variations in notation or method of using the UAN, or
- an incident in which the UAN provided a notation or a view of the interface that would not have occurred otherwise.

We selected the 40 critical incidents (Appendix C) to provide the broadest possible coverage of the taxonomy, i.e., so that they would not all map to a small number of cells. The five interface designers were selected on the basis of their varying levels of experience with the UAN. Experience level of potential interface designers was determined by a survey asking for number of years working with the UAN, number of completed projects using the UAN, size of completed projects using the UAN, and number of courses in the area of human-computer interaction. We intentionally chose a mix of interface designers from UAN novices to UAN experts in order to demonstrate that the taxonomy would produce reliable results across experience levels. Subjects were shown two example mappings and

were given a brief description of the taxonomy. Each subject then independently mapped each of the 40 incidents by writing the number of the incident in the cell in the taxonomy in which they thought it best fit (Appendix C). For example, if an incident was reported in which two UAN users created different hierarchical task structures during task analysis, then that incident might be mapped to the cell of *task analysis* on the *scope of process activities* axis, *hierarchical relationships* on the *content of interaction* axis, and *expressiveness* (which includes the idea of consistency of the technique) on the *quality factors of the technique* axis.

Reliability of the taxonomy is the extent to which the five interface designers agreed on their mappings of the 40 critical incidents. Of course *some* agreement in mappings would be expected by chance alone: that is, even if the interface designers assigned incidents to cells at random. Cohen's kappa (Cohen, 1960) is a measure of the proportion of agreement above and beyond what would be expected on the basis of chance. Kappa is scaled between 0 and 1, with 0 corresponding to only chance agreement and 1 corresponding to perfect agreement. Kappa is approximately normally distributed and can be used to test the null hypothesis of no agreement beyond the chance level.

While kappa is limited to assessing agreement between two subjects, an extension (Fleiss, 1971) permits measuring agreement among several subjects. As with the two-subject version, the extension also produces a kappa value between 0 and 1, and also allows testing for agreement by reference to the normal distribution. In this study we employed the extended version to measure agreement among our five interface designers. To gauge the reliability of each dimension in the taxonomy individually, separate tests were conducted for each of the three dimensions. Table 5-1 shows the proportion of agreement, kappa, z-value (standard normal scaling of kappa), and p-value for each of the three dimensions. For each test, the p-value

is the chance of obtaining the observed level of agreement by chance alone. The extremely small p-values shown in Table 5-1 provide convincing evidence that each dimension of the taxonomy is reliable.

Table 5-1. Results of analysis of incident mapping

Dimension	Proportion of Agreement	Kappa	z-value	p-value
Scope	0.8725	0.8362	28.56	<0.00001
Content	0.6575	0.6224	36.04	<0.00001
Quality	0.8925	0.8327	20.10	<0.00001

While these results show highly significant levels of agreement among the five subjects, the level of agreement on the Content axis appears lower, since the proportion of agreement and kappa values for the Content axis are well below those of the other two axes. However, since there are 12 levels on the Content axis but only 3 and 5 levels, respectively, on the Scope and Quality axes, it is less likely that all subjects would agree on the best fit of an incident along this axis. As indicated by the larger z-value for the Content axis, the obtained level of agreement provides even more convincing evidence of reliability for this dimension than for either the Scope or Quality axes.

We should note that there are two domains within which we might like our results to be general: the domain of critical incidents and the domain of subjects. For statistical purposes, the sample size of 40 for each test (the number of critical incidents) implies that our taxonomy is reliable for the domain of all critical incidents. Although our five subjects were chosen to span a wide range of experience with the UAN, the statistical results do not speak to generality in this domain; we can only speculate that similar consistency would be obtained among any reasonable set of subjects.

We have shown that each dimension of our model is reliable.

However, looking at our model as a whole, we would expect the probability (including chance) of two analysts or interface designers mapping a single incident to the same cell to be the product of the probability of agreeing on each axis or

$$0.8725 * 0.6575 * 0.8925 = \underline{0.5119}.$$

This points out that while these results are highly statistically significant suggesting that the model is reliable, in practical application behavioral representation technique designers and analysts will need to be cautious when comparing results across individuals.

In the second task, *rating*, used to evaluate reliability of the taxonomy of behavioral representation techniques, we had the same five interface designers use the taxonomy for analysis of the UAN. This process involved going through each cell in the taxonomy and rating the UAN on a scale of 1 to 3, 1 being the lowest rating and 3 being the highest rating, with respect to that cell. Each interface designer was given a copy of the new UAN tutorial (Appendix B). Using the taxonomy cell from the previous mapping example, i.e., the cell of task analysis, hierarchical relationships, and expressiveness, the UAN might be assigned a score of 3 for this cell since it is very effective at representing decomposition of larger tasks into a hierarchical task structure.

While perfect agreement among all interface designers in the rating of a cell would provide the most convincing evidence of reliability, it would be unreasonable to expect different analysts to give exactly the same ratings. Accordingly, we considered two possible criteria for "agreement," each allowing for some minor degree of discrepancy among the five ratings for an intersection. For the first criterion, which we refer to as *3 or more agreement,* three of the five interface designers had to agree exactly on the score and the other two could

differ by at most one from that score, e.g., three analysts give a score of 3, and the other two both give scores of 2. For the second criterion, which we refer to as *4 or more agreement*, four of the five interface designers had to agree exactly on the score and the other could differ by at most one, e.g., four analysts select 1 and the other selects 2. For each criterion we compared the observed proportion of agreements with the proportion that would be expected on the basis of chance alone. The comparison used was a test on single proportion, using a normal approximation to the binomial distribution. The sample size, n, for this comparison was the total number of cells in the taxonomy or 180. The results of these comparisons are shown in Table 5-2.

Table 5-2. Results of analysis of ratings comparisons

Criterion	Probability of Agreement by Chance	Proportion of Agreement	z-value	p-value
3 or more	0.3416	0.7795	12.89	< 0.00001
4 or more	0.0947	0.5641	22.39	< 0.00001

Again, as with the incident mapping task, these results indicate a highly significant level of agreement among the five subjects. While the proportion of agreement is somewhat lower for the *4 or more agreement* criterion, the probability of obtaining such agreement by chance is extremely small and therefore the resulting significance is actually higher than that of the *3 or more agreement* method. We can conclude from these results that the taxonomy is reliable for the analysis/rating of an existing behavioral representation technique. This type of analysis can provide insight into areas of needed improvement for a given technique.

5.2 Demonstrating Utility of the Taxonomic Model

The other criterion for evaluation of the taxonomy of behavioral representation techniques was *utility* — to show that it can provide useful functionality in some measurable way. To do this, our taxonomy was applied to the problem of UAN development and extension as discussed in Section 4.3. If a new version of the UAN based on the taxonomy could be shown to be an improvement over the old version, then the conclusion could be drawn that the taxonomy of behavioral representation techniques does, indeed, provide useful functionality.

In order to perform the first part of the experiment, we wrote a UAN tutorial (Appendix A) for the original version of the UAN, providing basic information of how to use the UAN in representing an interface design. This original version of the UAN was the one used by most of the real-world development sites, mentioned earlier. Using this tutorial, the original UAN was introduced (or in some cases re-introduced) to approximately 30 industrial and academic interface development sites. These sites were asked to report to us critical incidents, as described in Chapter 3, in using the UAN. We mapped reported incidents into the taxonomy of behavioral representation techniques to identify areas of needed improvement. We used these mappings, along with the ratings of the UAN based on the taxonomy, and other observations and literature study (as discussed in Chapter 4) to create an improved, extended version of the UAN. This new version was documented with a new version of the UAN tutorial (Appendix B).

In order to evaluate utility of the taxonomy, we selected six students from a graduate human-computer interaction course in which all six had equal exposure to the original UAN and other human-computer interaction development concepts. These students were given a

single simple interface task, the **select (object)** task described in the new tutorial in Appendix B, to describe using the original UAN. The students were also asked to supply their grade point average. Based on the quality of their UAN descriptions and their grade point averages, the six students were divided into two groups of three, each, roughly comparable in both UAN skills and grade point average. One group was given the old UAN tutorial (Appendix A) and the other group was given the new tutorial (Appendix B).

We then gave both groups the same interface prototype, a graphical interface prototype for a disability assessment system developed in Hypercard (Appendix D), and asked them to write a description of the interface using *only* their respective versions of the UAN. Each group was cautioned to include in their UAN descriptions just the features and symbology of the UAN contained in their tutorial. The groups met separately and were given four hours to complete the UAN descriptions for the disability assessment interface. The members of each group were then asked to rate their respective versions of the UAN using the taxonomy of behavioral representation techniques. They accomplished this by rating their versions at each cell in the taxonomy on a scale of 1 to 3 (with 1 being the lowest rating and 3 being the highest rating).

We then compared the old and new versions using two criteria:

- each subject's perception of the version of the UAN used as recorded by taxonomy rating results and

- quality of UAN task descriptions for the disability assessment interface as determined by a panel of human-computer interaction experts.

Our behavioral representation technique taxonomy includes a total of 180 cells (5 levels of Scope by 12 levels of Content by 3 levels of

Quality), each of which contained a rating on the three-point scale by each subject. For each cell two means were computed, one across the three subjects who used the old version of the UAN, and one across the three subjects using the new version of the UAN. Old and new versions of the UAN were then compared across all 180 cells using a paired t-test. As can be seen in Table 5-3, we are extremely confident that the new version of the UAN is rated higher by its users than the old version.

Table 5-3. Results of t-test of user ratings comparison

Number of Observations	Mean Difference in Ratings	t-value	p-value
180	0.7044	15.20	< 0.00001

The mean rating for the old version of the UAN across all 180 cells was 1.5930, while the mean rating for the new version was 2.2974. Although these numbers may appear similar, we should bear in mind that our scale allowed ratings only between 1 and 3, so that the observed difference of 0.7044 represents approximately 35% of the entire scale. Looked at from another angle, we could state that the new version produced a rating 44% higher than the old version.

In the second part of the experiment to evaluate improvement in the UAN, a panel of four human-computer interaction experts used the same three point scale to rate the design representations created by the groups from the first part of the experiment. These experts were selected based on their expertise in human-computer interaction, as well as their level of experience with the UAN, again determined by the results of the UAN experience survey described in the reliability experiment. Each of these experts compared the two sets of UAN design representations to the interface prototype and rated each set at 36 of the 180 cells of the taxonomy. (Since actual design representations were being rated, the only relevant level on the

Scope axis was the *design* activity. Each UAN design description was therefore rated for each combination of the 12 levels of Content by 3 levels of Quality, resulting in 36 ratings.) Presentation order of original and new versions of UAN was balanced across subjects. The resulting ratings were averaged across the four judges for each cell and each version. These means were then compared using a paired t-test, the results of which are summarized in Table 5-4.

Table 5-4. Results of t-test of expert ratings comparison

Number of Observations	Mean Difference in Ratings	t value	p value
36	0.361	3.28	0.0012

Across all 36 cells rated, the mean rating of the UAN design representations created using the original version was 1.3889, while the mean for the UAN design representations based on the new version was 1.75, representing an improvement of 26%.

Since the new version of the UAN was developed using the taxonomy of behavioral representation techniques, we can conclude that the taxonomy does provide useful function in a statistically measurable way. However, we should note that this was not a double-blind study. The subjects in both cases were aware that they were either using or viewing a new version or an old version of the UAN. Because the new version is a superset of the old, a double-blind study was not possible. Therefore, these results may have been influenced by psychological demand, i.e. since it is a new version, it must be better than the old version.

6. SUMMARY AND CONCLUSIONS

As stated in Chapter 1, the contribution of this research to the field of human-computer interaction is at three different levels:

- Improvement in the UAN
- Development and evaluation of a taxonomic model of behavioral representation techniques
- Exploration and demonstration of the epistemological cycle applied in human-computer interaction.

6.1 Goal 1: Improvement in the UAN

The first of these goals, our original goal of improving the User Action Notation (UAN), was accomplished as demonstrated by the experiments described in Chapter 5. Users gave the new UAN 44% higher ratings than the previous version of the UAN. Also, the quality of task descriptions made using the new UAN was judged to be 26% better than those created using the original UAN.

6.2 Goal 2: Development and Evaluation of a Taxonomic Model of Behavioral Representation Techniques

Our second goal was to develop and evaluate a taxonomic model for discussing, comparing, and evaluating behavioral techniques for representing user interface designs. We needed this taxonomy to facilitate our improvement of the UAN. We developed such a taxonomy through a process of observation, theory formulation, and evaluation. The taxonomy contains three dimensions, each with discrete attributes: *Scope* (activities within the interface development process that can use the technique), *Content*

(interaction components that can be represented), and *Quality* (characteristics such as learnability, accuracy, and reliability).

It was not possible to evaluate the taxonomy on the basis of completeness, but it was evaluated with regard to reliability and utility. Through two empirical evaluations, we have shown that our taxonomy of behavioral representation techniques is *reliable* for incident mapping and technique analysis. We used an extension of Cohen's kappa as a measure of agreement to show that subjects agreed, beyond the level expected by chance, on mappings from critical incidents observed in the use of representation techniques to cells in the taxonomy. On average, we achieved more than 75% of the possible agreement beyond chance. We thus conclude that this taxonomy should be useful to behavioral representation technique designers/analysts who are iteratively refining an existing behavioral representation technique, even if results are being compared across different individuals/analysts. The fact that the taxonomy tested reliably confirms that there are discernible attributes of behavioral representation techniques. Our use of a mix of subjects from novices to experts implies that these attributes are recognizable even to novice interface designers/analysts. This reinforces the notion that interface designer/analysts should be able to recognize these attributes, select the ones that are most important in their environment and choose a technique accordingly using the taxonomy and associated ratings of behavioral representation techniques.

The *utility* of the taxonomy was also demonstrated in terms of its ability to provide the basis for an improved version of the UAN.

The taxonomy may provide the ability to predict problems with behavioral representation techniques in areas where none were reported and to foresee changes that could be useful. Just as with interfaces themselves, anything that can reduce the amount of

empirical testing of a behavioral representation technique is indeed useful.

6.3 Goal 3: Exploration and Demonstration of the Epistemological cycle Applied in Human-Computer Interaction

We accomplished our third goal, exploration and demonstration of the epistemological cycle applied in human-computer interaction, throughout the process of creating and evaluating the taxonomic model of behavioral representation techniques. Our model was developed, *theorized*, from data collected through *observation*. Specifically, we observed the use of the User Action Notation (UAN) and analyzed other existing behavioral representation techniques. From the data collected, we were able to construct, or *theorize*, a taxonomic model for classifying the attributes of behavioral representation techniques. The final step, *evaluation*, was accomplished by first identifying the criteria to be used, i.e. *utility*—as used by the developers of the device models—and *reliability*—as defined in the literature regarding behavioral representation techniques—and then developing strategies to test each of these criteria.

Our work demonstrated a method for creating a more scientific foundation in the area of human-computer interaction. This work should serve as a model of the application of the epistemological cycle of *observation*, *theorization*, and *evaluation* to problems within human-computer interaction.

7. FUTURE WORK

This research covered a wide variety of topics from the epistemological cycle in general to very specific modifications and extensions to the UAN. Four general areas of future work are suggested:

- *Model Validity*—the ability to base predictions about a representation technique on the model ratings for that technique,

- *Model Extension*—the further development of the model,

- *Method Extension*—the use of the model development method in other related domains,

- *UAN Extension*—the continued development and refinement of the UAN.

Model Validity means that if a taxonomy is truly a model of behavioral representation techniques then it ought to be capable of making predictions for things such as the number of critical incidents within a given cell. Certainly, at this point, judgments can be made about which technique should work best for a given application by comparing the ratings for a given cell or subspace. However, predicting the number of critical incidents in a given cell is a much more difficult problem. One major problem would be that the number of incidents might be more dependent on the nature of the interactive system being developed than on the representation technique or the taxonomy.

Model Extension will be necessary since, as stated earlier, we can only suggest that the model should be complete with respect to existing behavioral representation techniques. As new techniques

are developed, there will certainly be extensions necessary to the model. Further, the model could be extended to provide a connection to existing related models, e.g. the device models. This type of interconnection between models could eventually provide more comprehensive coverage, and hopefully a greater understanding, of the entire interaction development process.

Method Extension refers to the idea that since we have shown how such a model can be developed and evaluated, perhaps similar models in related domains, e.g. interaction development tools, analytical models, programming languages, could be useful.

UAN Extension probably needs little explanation. Any representation technique that is not continually updated and refined will most certainly soon be out of date. The UAN is no exception. New interaction styles will demand continued evolution of the technique. Further, many existing conventions need not remain unchanged. As we found with the temporal notation in the previous version of the UAN, even a good thing can be made better.

8. REFERENCES

Bennett, J. L. (1986). Tools for Building Advanced User Interfaces. *IBM Systems Journal, 25*(3/4),

Bleser, T. W. (1991). *An Input Device Model of Interactive Systems Design*. Dissertation, George Washington University,

Butler, K., Bennett, J., Polson P., and Karat, J. (1989). Report on the Workshop on Analytical Models - Predicting the Complexity of Human-Computer Interaction. *20*(4),

Card, S. K. & Moran, T. P. (1980). The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM, 23*, 396-410.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Hillsdale, NJ: Erlbaum.

Cardelli, L. & Pike, R. (1985). Squeak: a Language for Communicating with Mice. *Computer Graphics, 19*(3), 199-204.

Carroll, J. M., Kellogg, W. A., & Rosson, M. B. (1991). The Task-Artifact Cycle. In J. M. Carroll (Ed.), *Designing Interaction: Psychology at the Human-Computer Interface* (pp. 74-102). New York: Cambridge University Press.

Carroll, J. M. & Olson, J. R. (1988). Mental Models in Human-Computer Interaction. In M. Helander (Ed.), *Handbook of Human-Computer Interaction* (pp. 45-65). Amsterdam: North-Holland.

Census, B. o. t. (1990). *Statistical Abstract of the United States, 110th Edition* . U.S. Government Printing Office.

Chase, J. D. & Casali, S. P. (1991). *A Comparison of Three Cursor Control Devices on a Cursor Control Benchmark Task* (TR 91-8). Virginia Tech.

Chase, J. D., Casali, S. P., & Hartson, H. R. (1992). The Predictability of Cursor Control Device Performance Based on a Primitive Set of

User Object-Oriented cursor Actions. *Proceedings of Human Factors Society 36th Annual Meeting*, Santa Monica: Human Factors Society, 306-310.

Chase, J. D. & Hix, D. (1994). The Analysis of an Existing Interface Using the User Action Notation: A Case Study. *Proceedings of Mid-Atlantic Regional Conference of the Human Factors Society*, Reston VA: Human Factors Society,

Chase, J. D., Peretti, M., Hartson, H. R., & Hix, D. (1993). Task-Oriented User Documentation Using the User Action Notation, Submitted to *HCI International '93, Orlando, FL.*

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement, 20*, 37-46.

Flecchia, M. & Bergeron, R. D. (1987). Specifying Complex Dialogs in ALGAE. *Proceedings of CHI+GI Conference on Human Factors in Computing Systems*, New York: ACM, 229-234.

Fleiss, J. L. (1971). Measuring Nominal Scale Agreement Among Many Raters. *Psychological Bulletin, 76*(76), 378-382.

Foley, J., Gibbs, C., Kim, W., & Kovacevic, S. (1988). A Knowledge-Based User Interface Management System. *Proceedings of CHI Conference on Human Factors in Computing Systems*, New York: ACM, 67-72.

Gould, J. D. & Lewis, C. (1985). Designing for Usability: Key Principles and What Designers Think. *Communications of the ACM, 28*(3), 300-311.

Green, M. (1985). The University of Alberta User Interface Management System. *Computer Graphics, 19*(3), 205-213.

Green, T. R. G. (1989). Task Action Grammar, presented at the British Computer Society HCI Specialists Group Day Meeting on Task Analysis, May, London. No proceedings.

Hartson, H. R., and Hix, D. (1990). *Building Bridges and Interfaces:*

Developing User-Centered Representation Techniques For Communicating User Interface Designs. A proposal submitted to the National Science Foundation from the Department of Computer Science, Virginia Tech:

Hartson, H. R., Brandenburg, J. L., & Hix, D. (1992). Different Languages for Different Development Activities: Behavioral Representation Techniques for User Interface Design. In B. A. Myers (Ed.), *Languages for Developing User Interfaces* (pp. 303-326). Boston: Jones & Bartlett.

Hartson, H. R. & Gray, P. D. (1992). Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction, 7,* 1-45.

Hartson, H. R. & Hix, D. (1989). Toward Empirically Derived Methodologies and Tools for Human-Computer Interface Development. *International Journal of Man-Machine Studies, 31,* 477-494.

Hartson, H. R., Siochi, A. C., & Hix, D. (1990). The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs. *ACM Transactions on Information Systems, 8*(3), 181-203.

Hill, R. (1987). Event-Response Systems — A Technique for Specifying Multi-Threaded Dialogues. *Proceedings of CHI+GI Conference on Human Factors in Computing Systems,* New York: ACM, 241-248.

Hix, D. & Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability Through Product and Process.* New York: John Wiley & Sons, Inc.

Jacob, R. J. K. (1982). Using Formal Specification in the Design of Human-Computer Interfaces. *Proceedings of Human Factors in Computer Systems,* Gaithersburg, MD: ACM Press, 315-321.

Jacob, R. J. K. (1985). An Executable Specification Technique for Describing Human-Computer Interaction. In H. R. Hartson (Ed.), *Advances in Human-Computer Interaction* (pp. 211-242). Norwood, NJ: Ablex.

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

Johnson, P., and Johnson, H. (1988). Practical and Theoretical Aspects of Human Computer Interaction. *JIT, 3*(3),

Johnson, P., and Johnson, H. (1991). Knowledge Analysis of Tasks: Task Analysis and Specification for Human-Computer Systems. In A. Downton (Ed.), *Engineering the Human-Computer Interface* London: Mcgraw Hill.

Kaplan, A. (1964). *Conduct of Inquiry : Methodology For Behavioral Science.* Chandler Pub. Co.

Kieras, D. & Polson, P. G. (1985). An Approach to the Formal Analysis of User Complexity. *International Journal of Man-Machine Studies, 22*, 365-394.

Mackinlay, J., Card, S. K., & Robertson, G. G. (1990). A Semantic Analysis of the Design Space of Input Devices. *Human Computer Interaction, 5*(2),

Mirel, B., Fineberg, S., & Allmendinger, L. (1991). Designing Manuals for Active Learning Styles. *38*(1), 85-87.

Moran, T. P. (1980). A Framework for Studying Human-Computer Interaction. In e. a. Guedj (Ed.), *Methodology of Interaction* (pp. 293-301). North-Holland Publishing Co.

Moran, T. P. (1981). The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems. *International Journal of Man-Machine Studies, 15*, 3-51.

Nadin, M. (1988). Interface Design and Evaluation - Semiotic Implications. In H. R. Hartson and Hix, D. (Ed.), *Advances in Human-Computer Interaction* New Jersey: Ablex Publishing.

Norman, D. A. (1986). Cognitive Engineering. Chapter 3 In D. A. Norman & S. W. Draper (Ed.), *User Centered System Design* (pp. 31-61). Hillsdale, NJ: Erlbaum.

Olsen, J. R. (1988). Cognitive Analysis of People's Use of Software. In J. M. Carroll (Ed.), *Interfacing Thought* (pp. 260-293). London: MIT Press.

Olson, J. R. & Olson, G. M. (1990). The Growth of Cognitive Modeling in Human-Computer Interaction Since GOMS. *Human-Computer Interaction, 5*, 221-265.

Payne, S. J. & Green, T. R. G. (1986). Task-Action Grammars: A Model of the Mental Representation of Task Languages. *Human-Computer Interaction, 2*, 93-133.

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

Reisner, P. (1983). *Analytic Tools for Human Factors of Software* (RJ 3808 (43605)). IBM Research Laboratory, San Jose, CA.

Rosson, M. B., and Carroll, J.M. (1992). *Extending the Task-Artifact Framework: Scenario-Based Design of Smalltalk Applications* (RC 17852 (#78516)). IBM Research Division.

Rosson, M. B., Maass, S., & Kellogg, W. A. (1987). Designing for Designers: An Analysis of Design Practice in the Real World. *Proceedings of CHI+GI Conference on Human Factor in Computing Systems*, New York: ACM, 137-142.

Sharratt, B. (1990). Memory-Cognition-Action Tables: A Pragmatic Approach to Analytical Modelling. *Proceedings of Interact '90*, Amsterdam: Elsevier Science Publishers, 271-275.

Shneiderman, B. (1992). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (2nd Edition). Reading, MA: Addison-Wesley.

Sibert, J. L., Hurley, W. D., & Bleser, T. W. (1988). Design and Implementation of an Object-Oriented User Interface Management System. In H. R. Hartson (Ed.), *Advances in Human-Computer Interaction* (pp. 175-213). Norwood, NJ: Ablex.

Simon, T. (1988). Analysing the Scope of Cognitive Models in Human-Computer Interaction: A Trade-Off Approach. *Proceedings of People and computers IV: The Fourth Conference of the British Computer Society Human-Computer Interaction Specialist Group*, University of Manchester:

Wasserman, A. I. & Shewmake, D. T. (1985). The Role of Prototypes in the User Software Engineering Methodology. In H. R. Hartson (Ed.), *Advances in Human-Computer Interaction* (pp. 191-210). Norwood, NJ: Ablex.

Wilson, M. D., Barnard, P. J., Green, T. R. G., & Maclean, A. (1988). Knowledge-Based Task Analysis for Human-Computer Systems. In G. C. v. d. Veer, T. R. G. Green, J.-M. Hoc, & D. M. Murray (Ed.), *Working with Computers: Theory Versus Outcome* (pp. 47-87). London: Academic Press.

Young, R. M., Barnard, P., Simon, T., and Whittington, J. (1989). How Would Your Favorite User Model Cope With These Scenarios? *ACM SIGCHI, 20*(4), 51.

Yunten, T. & Hartson, H. R. (1985). A SUPERvisory Methodology And Notation (SUPERMAN) for Human-Computer System Development. In H. R. Hartson (Ed.), *Advances in Human-Computer Interaction* (pp. 243-281). Norwood, NJ: Ablex.

APPENDIX A: OLD UAN TUTORIAL

THE UAN: A USER-ORIENTED NOTATION FOR DIRECT MANIPULATION INTERFACE DESIGNS

A TUTORIAL

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

Contact Persons:

Dr. Deborah Hix or Joseph D. Chase
hix@vtopus.cs.vt.edu / chase@vtopus.cs.vt.edu
703/231-6199

ABSTRACT

Almost all existing user interface representation techniques are constructional, focused on interface implementation, and therefore do not adequately support a user-centered focus. *But it is in the behavioral domain of the user that interface designers and evaluators do their work.* We are seeking to focus on the interface user during design by providing a notation that specifies the *behavioral aspects* of an interactive system—the tasks and the actions a user performs to accomplish those tasks. In particular, this paper is a practical introduction to the use of the User Action Notation (UAN), a task- and user-oriented notation for behavioral representation of asynchronous, direct manipulation interface designs. Interfaces are specified in the UAN as a quasi-hierarchy of asynchronous tasks. At the lower levels, user actions are associated with feedback and system state changes. The notation makes use of visually onomatopoetic symbols, and is simple enough to read without much instruction. The UAN has been used by growing numbers of interface developers and researchers over the past few years.

1. INTRODUCTION

High usability of an interface stems from a good design. Good designs depend on the ability of each person in a developer role—for example, designer, implementer, evaluator, customer, bidder—to understand and evaluate (and thereby improve) interface designs in the development process. Understanding and evaluating designs depends, in part, on the methods used to *represent* those designs. Design and representation are very closely related; design is a creative, mental, problem-solving process and representation is the physical process of capturing or recording the design.

One behavioral technique that has long been used both formally and intuitively is scenarios (or story-boarding) of interface designs. While this technique is effective for revealing a very early picture of interface appearance, it cannot represent the complete design. Scenarios can show much about screen layout, but show little about the user's behavior while interacting with the computer.

The User Action Notation, or UAN, is a task-oriented notation that describes the behavior of the user and the interface during their cooperative performance of a task. The primary abstraction of the UAN is a user task, i.e. some action that the user takes. A user interface is represented as a quasi-hierarchical structure of tasks that are asynchronous, i.e., the sequencing within each task is independent of that in the others. User actions, corresponding interface feedback, and state information are presented at the lowest level. Levels of abstraction, where lower level tasks are combined under a single general task name, are used to hide these details and represent the entire interface. At all levels, user actions and user tasks are combined with temporal relations such as sequencing, interleaving, and concurrency to describe allowable temporal user behavior. The UAN is used to supplement scenarios, indicating precisely how the user interacts with screen objects shown in a scenario.

2. A SIMPLE EXAMPLE

This tutorial will present the UAN via a series of examples. It is important to note that the UAN is primarily a design representation technique. However, for purposes of illustration, some of the examples in this paper will present notation representing existing interface designs.

Imagine a hypothetical description in a user manual for the task of selecting a file icon. This task might be described in prose as:

1. Move the cursor to the file icon.
2. Click, (depress and immediately release), the mouse button

The UAN description for this task is as follows:

	User Action	Explanatory Notes
1.	~[file_icon]	move the cursor to the context of the file_icon
2.	Mv^	press and release the mouse button

The ~ denotes moving the cursor, in this case into the context of the file icon. The brackets [] represent the context of whatever is contained within them. The context of an object is defined to be whatever location that is needed to manipulate the object, i.e. the interior of a button, the edge of a box, or the handle of a graphical object. The second line represents depressing (v) and releasing (^)—i.e. clicking, the mouse button (M). Even this simple example illustrates the brevity and readability of the UAN. The UAN is presented in a column format with each column representing some specific aspect of the task description. The User Action column shown above is normally present in all examples of UAN. The Explanatory Notes column, however, is an optional column that is used to further describe aspects of the design to the implementors.

Let us describe another task, moving a file icon, that can be described in prose as:

1. Move the cursor to the file icon. Depress and hold down the mouse button. Depressing the mouse button selects the file, indicated by the highlighting of its icon.
2. With the button held down, move the cursor. An outline of the icon follows the cursor as you move it around.
3. Release the mouse button. The display of the icon is now moved to where you released the button.

The UAN description is shown below:

Line	User Action	Explanatory Notes
1.	~[file_icon] Mv	move the cursor to the context of the file icon and press the mouse button
2.	~[x, y]* ~[x', y']	move the cursor to the point represented by x',y'
3.	M^	release the mouse button

Reading this task description, we again note moving the cursor into the context of the icon and pressing the mouse button. In the second line, ~[x, y] indicates movement of the cursor to an arbitrary point x,y on the screen. The * means to perform, zero or more times, the task to which it is attached. Thus, ~[x,y]* ~[x',y'] means to move the cursor in a succession of zero or more arbitrary points about the screen, ending at the point x',y'. It is important to note that this movement represented by ~ is path independent. Thus a move to a point x',y' could be represented by ~[x',y']. The notation ~[x,y]* is usually only included when there is a need to associate feedback or interface state information with the movement. The reasons for naming the screen locations x,y and x',y' are more thoroughly explained in section 3.2. Finally, in the third line the mouse button is released.

Another example of a common interface task is that of entering a string of characters via some input device, say a keyboard. The prose description of that task might be:

Type the string "Hello World" on the keyboard.

The UAN representation of the same task is much more concise. Letting K represent the keyboard, the UAN notation for this task is:

User Action	Explanatory Notes
K"Hello World"	enter the string "Hello World" via the keyboard.

As you can see, the User Action Notation provides a much clearer and more concise description in all three of the above examples.

3 MORE ON THE UAN

3.1 Interface Feedback

Section 2 showed how to describe the *user actions* necessary for a task. Comparing the UAN description to the prose in the simple examples of chapter 2, we find that the prose also contains feedback and interface state information. *Interface feedback* information—interface responses to user actions—allows a more complete description of user and interface behavior, as seen in Figure 1.

TASK: Move_A_File_Icon	
USER ACTIONS	INTERFACE FEEDBACK
~[file_icon] Mv	file_icon!
~[x,y]* ~[x',y']	outline of file_icon follows cursor
M ^	display file_icon at x',y'

Figure 1. UAN description of the task "Move_A_File_Icon" with interface feedback in response to user actions

A UAN task description is read left to right, top to bottom, and indicates in the first line that when the user moves the cursor to the file icon and depresses the mouse button, the icon is highlighted (file_icon!). In the second line, as the user moves the cursor around the screen, an outline of the file icon follows the cursor, and upon release of the mouse button, the file icon is displayed at the new position. *Note the line-by-line association of feedback with the corresponding user action*; this level of precision is lost in the prose description, where actions and feedback are intermingled. For example, consider this description of selecting an icon:

1. Move the cursor to the icon.
2. Click the mouse button and the icon will be highlighted.

The corresponding UAN task description is shown is Figure 2.

TASK: Select_An_Icon	
USER ACTIONS	INTERFACE FEEDBACK
~[icon] Mv^	icon!

Figure 2. UAN description of the task "Select_An_Icon"

In the Macintosh™ interface[1], however, highlighting occurs when the mouse button is depressed (rather than when it is clicked—depressed and released). Figure 3 shows how the UAN can be used to represent, more precisely than in Figure 2, this correspondence between feedback and separate user actions in the sequence.

TASK: Select_An_Icon	
USER ACTIONS	INTERFACE FEEDBACK
~[icon] Mv	icon!
M ^	

Figure 3. UAN description of the task "Select_An_Icon" more precisely showing relationship of feedback to user actions

The feedback in Figure 3 is still not complete, however. In this case, selection (and, therefore, highlighting) of icons is mutually exclusive. This means that this task is technically the task to "select one icon and deselect all others." A highlighting action applied to the icon (icon!) is conditional; that is, it is applied only if the icon is not already highlighted; i.e., highlighting depends upon the condition icon-!, which means "the icon is not highlighted". The feedback description in Figure 4 has been extended to include these two notions, where ∀ means "for all" and a colon is used between the condition and corresponding feedback. Now the feedback of line 1 states that if the icon is not highlighted, highlight it, and for all other icons that are highlighted, unhighlight them.

™ Macintosh is a registered trademark of Apple Computer.

[1]The UAN is not limited to the Macintosh nor is it oriented toward any one specific graphical direct manipulation interface style. However, we have taken advantage of the popularity of the Macintosh desk top to illustrate use of the UAN.

TASK: Select_An_Icon	
USER ACTIONS	INTERFACE FEEDBACK
~[icon] Mv	icon-!: icon!, ∀icon'!: icon'-!
M ^	

Figure 4. UAN description of the task "Select_An_Icon" more precisely showing complete feedback

If the designer feels that added information about dehighlighting other icons clutters the feedback description for an icon, abstraction can be used to hide those details. For example, the definition of highlighting (!) can contain the unhighlighting (-!) behavior for all other icons in the same mutually exclusive set.

While the feedback column of the example to move a file icon in Figure 1 is easy to read, it can be more precise. In particular, the feedback for highlighting the file_icon is similar to that of the icon in Figure 4. Also, the symbology X > ~ is used to denote object X following the cursor. The exact behavior of the outline as it follows the cursor in the second line and displaying the file icon in the third line can be encapsulated as feedback functions, defined precisely in a single place. The task description then appears as shown in Figure 5.

TASK: Move_A_File_Icon	
USER ACTIONS	INTERFACE FEEDBACK
~[file_icon] Mv	file_icon-!: file_icon!, ∀file_icon'!: file_icon'-!
~[x,y]* ~[x',y']	outline(file_icon) > ~
M ^	@x',y' display(file_icon)

Figure 5. UAN description of the task "Move_A_File_Icon" with more precise feedback description

Making the feedback description more complete improves precision and consistency of the resulting interface descriptions.

3.2 Naming of Objects and Screen Locations

User task descriptions require that locations and objects on the

screen be named, so that designers and implementers can refer to them. The nature of the names is a little bit like logical or algebraic variables.

Generic Names

A *generic name* refers to possibly many screen locations or objects that share a common characteristic, but have no distinction among them. For example, Figure 1 refers to an arbitrary point x,y. The expression in which it appears, ~[x,y]*, says literally "move to the context of point x,y any number of times." The intent, of course, is not to move to the same point each time. Because of the multiple moves associated with the point x,y, this name refers to possibly many different locations on the screen. Further, there is nothing to distinguish among the points to which x,y refers. Thus, x,y is a generic name. When symbols such as * or ∀, symbols that denote multiple occurrences are used, the associated variables usually are generic.

Specific Names

A *specific name* refers to a single particular location or object, a location or object that has only one occurrence for each instance of the task (i.e., each time a user performs a task). In contrast to x,y above, the name x',y' is a specific name because it refers to the particular point at which cursor movement stops in the second line of Figure 1.

Binding of Names

There is often a need to refer to the same location or object in two or more places within a task description. For example, in Figure 5, we wish to say, in the feedback column of the third line, that the point where the file icon is now displayed is the same point where the user stopped moving the cursor in line 2. So we use the same point name—x',y'. When two or more occurrences of the exact same name are used in a task description, they are assumed to refer to the same location or object and are said to be *bound* to each other. Exceptions include cases where it is obvious the occurrences are not related. For example, there may be occurrences of ~[x,y]* in two different places in

a task and it would be clear that the points for one independent path of the cursor would not have to be the same as those of another path. On the other hand, if a user action of ~[x,y]* has a feedback of x,y!, then the two occurrences of x,y would be bound together, meaning that each point in the path of the cursor would be highlighted as the cursor passed by.

Another example is seen in the first line of Figure 5. Here file_icon is a specific name whose use in the user action column is bound to its use in the feedback column. In other words, the file_icon that is highlighted is the same file_icon to which the user moved and depressed the mouse button.

Generic names can also be bound. In Figure 5, the second line of feedback for the first user action, file_icon is a generic name because of its use with the ∀ symbol. The phrase ∀file_icon'! means "for all file_icons that are highlighted." There are possibly many file_icons that share this characteristic, with no distinction among them. All occurrences of these (highlighted) file_icons are bound to the name as it is used after the colon in this feedback description, ∀file_icon'!:file_icon-!, meaning that all such (highlighted) file_icons are now to be unhighlighted.

3.3 Interface State and Computational Information

In addition to feedback, it may be necessary to include some *interface state and computational information* associated with user actions. For example, if the interface design includes the concept of selectable objects, details about how this concept is applied to objects must be communicated to the implementer. Figure 6 shows just the first line of the task description from Figure 5 for moving a file icon. This shows that depressing the mouse button when the cursor is over a file icon causes the object represented by the file icon (i.e., the file) to be selected. This approach to semantics allows a connection to be made between an object being selected and its icon being highlighted.

TASK: Move_A_File_Icon		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
~[file_icon] Mv	file_icon-!: file_icon!, ∀file_icon'!: file_icon'-!	selected = file

Figure 6. UAN description of the task "Move_A_File_Icon" with interface state information

If the location of the icon is significant to the computational (semantic or non-interface) component of the application, the computational component must be informed, as shown in the lower right hand cell of Figure 7.

TASK: Move_A_File_Icon			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	CONNECTION COMPUTATION
~[file_icon] Mv	file_icon-!: file_icon!, ∀file_icon'!: file_icon'-!	selected = file	
~[x,y]* ~[x',y']	outline(file_icon) > ~		
M ^	@x',y' display(file_icon)		location (file_icon) = x',y'

Figure 7. UAN description of the task "Move_A_File_Icon" with connection to computational semantics

3.4 Conditions of Viability

Consider the description of the user task of selecting a file icon given in Figure 8.

TASK: Select_A_File_Icon		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
~[file_icon] Mv	file_icon-!: file_icon!, ∀file_icon'!: file_icon'-!	selected = file
M ^		

Figure 8. UAN description of the task "Select_A_File_Icon"

If we wish to show that this task applies only to a file icon that is not

already selected, we make use of a *condition of viability*, which is similar to the condition applied earlier to the feedback, except here it is a condition applied to a user action or to an entire task. In Figure 9 the condition of viability is file_icon-!. The scope of the condition of viability is indicated with parentheses. A condition of viability acts as a precondition that must be true in order for user actions within its scope to be performed as part of this task. A condition of viability with a false value does not necessarily mean that a user cannot perform the corresponding action(s); it just means that the action(s), if performed, will not be part of this particular task. The same actions, however, might be part of another task in the overall set of asynchronous tasks that comprise an interface. Note that the use of this condition as a condition of viability for the user action removes the need for its use with the feedback. Note also that the object name, file_icon, in the condition is bound to the name in the user action and feedback.

TASK: Select_A_File_Icon		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
file_icon-!: (~[file_icon] Mv	file_icon!, ∀file_icon'!: file_icon'-!	selected = file
M ^)		

Figure 9. UAN description of the task "Select_A_File_Icon" with condition of viability (i.e., file icon is not already highlighted)

In Figure 10 the condition of viability (file_icon-!) from Figure 9 is written as a built-in binding (~[file_icon-!]), which means "move the cursor to an unhighlighted file icon." This is more concise and easier to read. In this form, conditions quite naturally provide specific instructions for user behavior, i.e., move the cursor to an unhighlighted file icon.

TASK: Select_A_File_Icon		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
~[file_icon-!] Mv	file_icon!, ∀file_icon'!: file_icon'-!	selected = file
M ^		

Figure 10. UAN description of the task "Select_A_File_Icon" with alternative (built-in binding) form for condition of viability

3.5 Another Example

The task description in Figure 11 ties together many of the previous concepts; it represents one method for the task of deleting a file from the Macintosh^{TM} desktop, namely by dragging the files icon to the trash icon.

TASK: Delete_File			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	CONNECTION COMPUTATION
~[file_icon] Mv	file_icon-!: file_icon!, ∀file_icon'!: file_icon'-!	selected = file	
~[x,y]*	outline(file_icon) > ~		
~[trash_icon]	outline(file_icon) > ~, trash_icon!		
M ^	erase(file_icon), trash_icon!!	selected = null erase outline(file_icon),	mark file for deletion

Figure 11. UAN description of the task "Delete_File"

3.6 Combining Tasks in UAN

So far we have discussed the UAN as it is used to represent simple, low level physical user actions directly associated with devices; what we call the articulation level of interface design. It is possible to build, on these physical actions, levels of abstraction to represent the complete task structure for an entire application. We shall use the terms task and action interchangeably here to underscore the fact that the higher level tasks and the lower level physical user actions

have many of the same properties when combined into a task structure. Following are some definitions that show how tasks can be combined in the UAN:

- The physical actions on devices described so far are tasks. Examples include all actions such as ~[X], Mv^, and so on.
- If A is a task, so are (A), A* where * means zero or more times, and {A} where { }
  means zero times or one time.(i.e. A is optional)
- If A and B are tasks, so are A B, A OR B, A&B, A⇔B, and A || B, where these are all
  temporal combinations that will be described shortly.

Thus, a task description written in the UAN is a set of user actions interspersed with logical and temporal operators. Higher level tasks built up as combinations of articulation level tasks using the above rules can be named and the name used as a reference to the task. A task is invoked by using its name as an action within another task, in the same manner that a procedure call in a computer program is used. Just as with computer procedures, this leads to higher levels of abstraction, necessary for understanding by readers and writers of the notation, and a quasi-hierarchical "calling" structure of tasks. Use of a task name as a user action corresponds at run-time to the invocation of a user-performed procedure.

Task descriptions written in the UAN form an *articulation* between two major activities within the interface development life cycle: task analysis and design. Because these task descriptions are at once representation of the task analysis hierarchy and a representation of the user interface design, the UAN supports *task analysis as a natural driver of the design process.*

As one example of the way symbols are used together in higher level task descriptions, consider the symbol |, used to indicate a disjunction of choices. For example, A | B | C denotes a three-way choice among user actions or tasks; a description commonly used when alternative methods exist for a given task. A common high level construct in the UAN is seen in this example:

(A | B | C)*

This expression is called a repeating disjunction and means that tasks A, B, and C are initially concurrently available. The user may chose

to perform A, B, or C. One of the three tasks is chosen and performed to completion, at which time the three tasks are concurrently available again. The cycle continues arbitrarily, each time just one of the three tasks is initiated and performed to completion. Note that this notation is a compact high level description of the use of a three choice menu.

Let us consider an example of disjunction to represent alternate methods of performing the same user task. The task of deleting multiple files from the Macintosh™ desktop is a good simple example of how a high level task can be decomposed into lower level tasks until the articulation level is reached. For example, the task of deleting multiple files can be decomposed into the two tasks:

1. Select multiple files
2. Delete selected files

However, we now see that we need notation to represent the temporal relationship among tasks at the same level, i.e. whether they are sequenced, exclusive, interleaved, etc., so we will discuss this notation before continuing this example.

3.7 Temporal Relations

User actions in many graphical interfaces are asynchronous, having rather more complex (less constrained) temporal behavior than those of simpler interfaces that are largely constrained to pre-defined sequences. A brief introduction to the concepts of temporal relationships among tasks is given in this section.

The most basic temporal relationships we have identified are

- sequenced with,
- are order independent,
- interruptible by,
- interleavable with,
- can be concurrent with,

listed in decreasing order of the temporal constraints they impose.

Sequencing is the most constrained temporal relation; the first action must be performed completely, then the next exactly in order, and so on, until all actions in a sequence are completed. while this constraint may make the interface some what easier to design and implement, it is often arbitrary and even opposed to the cognitive and task needs of the user. For example, initiation of a second task in the middle of a first task may be required in order to get information necessary to the completion of the first task. It may be very desirable that the second task can be interleaved with the first, so it will not destroy the working context of the first task.

Order independence is similar to a sequence in that all actions must be performed and each one completed before another is begun. But the constraint on specific ordering among actions is removed. An example of order independence at a very low task level is seen in the task of entering a "**Command-X**" in a Macintosh™ application, a combination of the and X keys. Since the key must be depressed before the X key, but the order of their release does not matter, the task is defined in UAN as:

Task: Command-X
v Xv (^ & X^)

While order independence relaxes the sequentiality constraint, *interleaving* removes the constraint of a task or action having to be performed to completion before beginning another action, i.e., allowing an action to be interrupted.

Tasks are mutually interleavable if they can *interrupt* each other. Consider the case of a help facility available during some other complex user action such as editing a document. If the help information, when invoked, appears in a window separate from the document being edited, the editing and help actions are interleavable. The user may alternate the focus of attention, and therefore task performance, from one window to the other.

While interleavability allows user actions to be alternated among tasks, *concurrency* allows user actions for two or more tasks to occur simultaneously. Concurrency is a temporal relation that has not been greatly exploited in user interfaces. Nevertheless, it has been shown [1] that there are cases in which it is possible and, indeed, preferable, to carry out more than one task at the same time—for example, via

input techniques that rely on the simultaneous use of both hands or for redundant input channels (e.g., keyboard and voice input).

Time intervals are also important in task descriptions. For example, the prose description of a double click with the mouse button might tell the user to click the mouse button and immediately click it again. This task can be represented precisely in the UAN. Note that waiting a certain amount of time, i.e., not doing any action for that interval, is itself a user action. The UAN task description for double clicking is:

Mv^ (t<n) Mv^

where the value of n can be controlled by the user via a control panel setting and an appropriate default value can be empirically determined by developers.

Now that we have seen how to combine tasks into higher level tasks using UAN, we will return to our earlier example of deleting multiple files on the Macintosh™ desktop. Looking at this example from a top down perspective, we realize that we need a task called **Delete_Multiple_Files** and that this task will be made up of two lower level tasks, **Select_Multiple_Files** and **Delete_Selected_Files**, which must be performed in sequence. Using what we now know about temporal relations, we can write the UAN description for the **Delete_Multiple_Files** task as follows:

Task: **Delete_Multiple_Files**

User Action	Explanatory Notes
select_multiple_files	invokes the task select_multiple_files
delete_selected_files	invokes the task remove_multiple_files

This description shows that in order to delete multiple files the user must first select multiple files and then delete the selected files. Each of these second level tasks can be decomposed into more detail using the UAN, (**NOTE:** It is still the UAN at all levels) as shown below for the task of selecting multiple files. This task illustrates a lower level of abstraction in this example since this task may be performed one of two ways. Therefore, this task is written as the

disjunction of, or a user selection between, two lower level tasks:

Task: Select_Multiple_Files

User Action	Explanatory Notes
Shift_select_files	invoke the shift_select_files task
OR Draw_box_select_files	invoke the draw_box_select_files task

Again, these two tasks can be further decomposed using the UAN. However, we have reached a level of decomposition where at least some of the tasks in the next lower level of detail will be articulation level UAN. Here it will be useful to examine the concept of reuse as it applies to task descriptions in the UAN.

3.8 Abstraction and Macros

The use of task names as abstractions—for modularity, consistency, and reusability—is illustrated in the following example, using the task of deleting multiple files from the Macintosh^{TM} desktop discussed above. To begin, Figure 12 shows the task description without the use of abstraction.

TASK: Delete_Multiple_Files			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	CONNECTION COMPUTATION
(Sv (~[file_icon] Mv M ^)+ S^)+	 file_icon-!: file_icon!, file_icon!: file_icon-!,	 selected = selected » file selected = selected—file	
~[file_icon!] Mv ~[x,y]*	 outline(icons!) > ~		
~[trash_icon]	outline(icons!) > ~, trash_icon!		
M ^	erase(icons!), trash_icon!!	selected = null	mark selected files for deletion

Figure 12. UAN description of the task "Delete_Multiple_Files"

Note that S denotes the shift key used here as a switch device. Figure 12 shows the user task sequence for using the shift key and mouse combination to select one or more files to be deleted and then dragging the file(s) to the trash icon to delete the file(s).

This task of deleting multiple files can be decomposed into two tasks:

1. Select files (the top block in Figure 12)
2. Delete selected files (the other three blocks in Figure 12)

Both of these tasks are performed often and can be shared as subtasks of other tasks as well. Therefore we can use the concept of macros to facilitate reusability, to help control interface consistency, and for brevity. Figure 13 shows how the overall task description of Figure 12 is stated in terms of names for these lower level tasks, which are then defined elsewhere. These tasks can be considered macros in that they are general, reusable task descriptions that can be used to delete any set of files and do not have to be rewritten

every time they are to be used. A more complete example of macros in the UAN is found in Section 5.

TASK: Delete_Multiple_Files			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	CONNECTION COMPUTATION
select_multiple_files			
delete_selected_files			

Figure 13. UAN description of the task "Delete_Multiple_Files" at a higher level of abstraction

The task of selecting multiple files can be done in (at least) two ways: using the shift key, as described in the first block of Figure 12, or by dragging out a selection rectangle with the mouse, as described in Figure 15 below. Figure 14 is a higher level description of the task **Select_Multiple_Files**, stated as a disjunction of the names of these two versions of the task.

TASK: Select_Multiple_Files			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	CONNECTION COMPUTATION
shift_multiple_select I			
drag_box_multiple_select			

Figure 14. UAN description of the task "Select_Multiple_Files" showing alternative methods

TASK: Drag_Box_Multiple_Select			
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE	CONNECTION COMPUTATION
~[x,y] Mv		x,y is fixed corner of rectangle	
~[x',y']*	dotted rectangle >> ~		
~[x",y"] M ^	items intersected by rectangle are !	selected = all intersected items	

Figure 15. UAN description of the task "Drag_Box_Multiple_Select"

4 FURTHER DISCUSSION OF THE UAN

4.1 Summary of UAN Symbols

The definition of the UAN has deliberately been kept open in the sense that interface developers can add and/or modify symbols or columns as needed for their particular design situation. Many of the symbols and idioms we have found useful are summarized in Table 1.

Table 1. Summary of some useful UAN symbols

Symbol	Meaning	Reference Sect.
~	move the cursor	2
[icon]	the context of object icon, the "handle" by which the icon is manipulated	2
~[icon]	move cursor into context of the icon	2
~[x,y]	move the cursor to (arbitrary) point x,y outside any object	4
~[x,y in icon]	move the cursor to (arbitrary) point within icon	4
~[X in Y]	move to object X within object Y (e.g., [OK_icon in dialogue_box])	4
[icon]~	move cursor out of context of the icon	4
v	depress	2
^	release	2
Xv	depress button, key, or switch called X	2
X ^	release button, key, or switch X	2
Xv^	idiom for clicking button, key, or switch X	2
X"abc"	enter literal string, abc, via device X	4
X(xyz)	enter value for variable xyz via device X	4
( )	grouping mechanism	4
*	iterative closure, task is performed zero or more times	4
+	task is performed one or more times	4
{ }	enclosed task is optional (performed zero or one time)	4
A B	sequence; perform A, then B (same if A and B are on adjacent, lines)	3
OR	disjunction, choice of tasks (show alternative ways to perform a task)	3
&	order independence; connected tasks must all be performed, but relative order is immaterial	3
⇔	interleavability; performance of connected tasks can be interleaved in time	3
||	concurrency; connected tasks can be performed simultaneously	3
;	task interrupt symbol; used to indicate that user may interrupt the current task at this point (the effect of this interrupt is specified as well, otherwise it is undefined, i.e., as though the user never performed the previous actions)	4
∀	for all	4
:	separator between condition and action or feedback	4

Feedback	Meaning	
!	highlight object	2
-!	dehighlight object	4
!!	same as !, but use an alternative highlight	4
!-!	blink highlight	4
$(!\text{-}!)^n$	blink highlight n times	4
@x,y	at point x,y	4
@icon	at the icon	4
display(icon)	display the icon	4
erase(icon)	erase the icon	4
icon > ~	the icon follows (is dragged by) cursor	4
icon >> ~	the icon is rubber-banded as its follows cursor	4
outline(icon)	outline of the icon	4

5. An In-Depth Example

As mentioned earlier, we believe the best way to learn UAN is by examples. Therefore, this section is dedicated to an in-depth example using the UAN. This example is based on a hypothetical automated calendar management system.

From systems, requirements, functional, and task analyses, lets assume we have determined that this calendar management system needs to provide four basic tasks: adding, viewing, modifying, and deleting an appointment on the calendar as well as the ability to establish an alarm associated with a given appointment. Lets also assume that we begin with a top-down approach. One possible design of the highest level of abstraction is shown in Figure 16 as a repeating disjunction of the five basic functions. At this level, the user has the choice of adding, viewing, modifying, or deleting an appointment or establishing an alarm, and this choice may be done an arbitrary number of times. Further, the system provides the ability to view the calendar by month, by week, by day, by individual time slot, or by a direct search for a specific date and/or time or appointment content, shown as a user choice of corresponding tasks in Figure 17. This repeating choice represents the process of locating a particular day of an appointment to be viewed. Then the user views the particular time slot. It is important to notice that at these higher levels of abstraction there is little or no feedback associated with user actions.

TASK: MANAGE_CALENDAR		
USER ACTION	FEEDBACK	INTERFACE STATE
(view_appointment | add_appointment | mod_appointment | del_appointment | establish_alarm)*		

Figure 16. Highest level of abstraction for calendar management system

TASK: VIEW_APPOINTMENT		
USER ACTION	FEEDBACK	INTERFACE STATE

(search
  | view_month
  | view_week
  | view_day)*
  view_time_slot		

Figure 17. UAN description of the view_appointment task for the calendar management system

Figure 18 shows the **View_Month** task of Figure 17 broken down into more detail. The notation here represents the user's choice, in this case to either select a particular month, move forward one month, or move back one month. As above, this choice may be made zero or more times. It is also important to note the use of a parameter, "any_month", in the first line of this example. This parameter allows the use of a macro **Select(object)** task through which the user will select any given object provided as a parameter by an invoking task. Figure 19 shows the articulation level UAN defining the **Select(object)** task. Again, here, it is important to note that at this low level of detail, feedback and state information are present.

TASK: VIEW_MONTH		
USER ACTION	FEEDBACK	INTERFACE STATE

(select(any_month)
  | move_forward_1mo
  | move_back_1mo)*		

Figure 18. UAN description of the view_month task for the calendar management system

TASK: SELECT(OBJECT)		
USER ACTION	FEEDBACK	INTERFACE STATE
~[OBJECT ICON-!]Mv	object_icon!, ∀object_icon'!: object_icon'-!	selected=object
M^		

Figure 19. UAN description of the select task for the calendar management system

Figures 20, 21, and 22 show the decomposition of the **Add_Appointment** task from Figure 16. The **Add_Appointment** task is relatively simple at the level of abstraction of Figure 20. The user merely views the time slot in which they would like to add an appointment and then edits the appointment. It is also important to note that, unlike most of the previous examples, this task is completely sequential. There is no user choice or iteration involved. Figure 21, showing the **View_Time_Slot** task, the final task of **View_Appointment** provides an example of a condition of viability, namely "view_level = day". If the calendar is not in the "day" view level (i.e. view_level = day), then the user cannot perform this task. That is to say, individual time slots can only be viewed at the day level and not at the week or month levels. This task description goes on to show that when the view_level is "day", the user may either scroll up or scroll down zero or more times and then select "any_time_slot". It is important to notice again, the use of the **select(object)** macro and parameter passing.

TASK: ADD_APPOINTMENT		
USER ACTION	FEEDBACK	INTERFACE STATE
view_appointment edit_appointment		

Figure 20. UAN description of the add appointment task for the calendar management system

TASK: VIEW_TIME_SLOT		
USER ACTION	FEEDBACK	INTERFACE STATE
view_level = day: ((scroll_up | scroll_dn)* select(any_time_slot))		

Figure 21. UAN description of the view time slot task for the calendar management system

TASK: EDIT_APPOINTMENT		
USER ACTION	FEEDBACK	INTERFACE STATE
(edit_person		
& edit_description		
& edit_location)		

Figure 22. UAN description of the edit appointment task for the calendar management system

Figure 22, showing the "edit_appointment" task, is an example of the use of the order independence notation "&". This means that all three of the described tasks must be performed to completion but they may be performed in any order.

Finally, Figure 23 shows the **Establish_Alarm** task invoked in Figure 16. This task invokes the **Set_Alarm** task, which is shown at the articulation level UAN in Figures 24. Figure 23 provides another example of a condition of viability. In this case, the view_level is checked to make sure that the interface is at the time_slot level. If it is not, then the "view_time_slot" task must be performed, otherwise it is not performed. The closing parenthesis after "view_time_slot" shows the end of the scope of the condition of viability. The user then performs the **Set_Alarm** and **Set_Alarm_Parameter** tasks sequentially.

TASK: ESTABLISH_ALARM		
USER ACTION	FEEDBACK	INTERFACE STATE
(view_level π time_slot: view_time_slot) set_alarm set_alarm_parameters		

Figure 23. UAN description of the establish alarm task for the calendar management system

TASK: SET_ALARM		
USER ACTION	FEEDBACK	INTERFACE STATE
view_level = time_slot: (~[alarm_icon]		
Mv	alarm_icon-!: alarm_icon!, ∀cmd_icon'!: cmd_icon'-!	selected=alarm command
~[x,y]*	outline(copy(alarm_icon))>~	
~[appointment_icon]	outline(copy(alarm_icon))>~, appointment_icon!	
M^)	display(copy(alarm_icon)), @x',y' in appointment_icon	

Figure 24. UAN description of the set alarm task for the calendar management system

Figure 24 again shows a condition of viability, "view_level = time_slot". However, checking for a closing parenthesis and not finding until the end of the last action shows us that this condition effects the entire task. Also it is important to notice that at this articulation level of the UAN the feedback and state information can be quite detailed and descriptive. For example, the feedback column in line two says that if the alarm_icon is not highlighted, highlight it. Further, if any command icons other than the alarm icon are highlighted then unhighlight them.

6. Extensions of the UAN

One of the strengths of the UAN is its adaptability to a wide variety of interface types, interactions styles, and design concerns. As more diverse uses are found for computers and the needs of special user groups, such as those with physical disabilities, are addressed, it becomes apparent that any notation intended to describe the behavioral domain must be extensible to address new technologies. Two such extensions to the UAN have already been documented. Notation has been added to the UAN[4] to describe physical dimensions of the interface, i.e., size of target icons, distance to target icons, direction to target icons, and so on. These extensions are listed in Table 2.

Table 2. Physical Extensions to the UAN

Distance Notation	~xx	move xx centimeters
	~xx[target]	move xx centimeters to target
Direction Notation	~ > [target]	move right to target
	~ < [target]	move left to target
	~ ^ [target]	move up to target
	~ v [target]	move down to target
	~ /^ [target]	move diagonally up (l to r) to target
	~ \^ [target]	move diagonally up (r to l) to target
	~ /v [target]	move diagonally down (r to l) to target
	~ \v [target]	move diagonally down (l to r) to target
Target Notation	target0.27	a 0.27 cm target
	target0.54	a 0.54 cm target
	target1.07	a 1.07 cm target
	target2.14	a 2.14 cm target

This extension of the UAN into physical dimensions was done in order to facilitate a study of the effects of physical dimensions on user performance and the predictability of user performance based on individual dimensions. An example of a task description using this physical notation is shown in Figure 25.

TASK: drag a 0.27 cm target 8 cm to the right recording movement time		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
~[target0.27]Mv	target0.27!	start timer target0.27 = selected
~>8	outline(target0.27)>~	
M^	@x',y' display(target)	stop timer

Figure 25. UAN Task description using physical notation

The first line of Figure 25 represents moving the cursor from its current location to an icon named target that is 0.27 cm in size, followed by depressing the mouse button. The result of this action is that the target is highlighted and timing is begun. Line two shows the action of moving the cursor 8 cm to the right with an outline of the target following the cursor. Line three shows the release of the mouse button, the redisplay of the target icon, and the end of timing.

Another example of an extension to the UAN is memory-cognition-action-tables[26]. In this case, rather than merely adding new symbols to the UAN, new columns were added to record user memory requirements and cognition requirements. User memory requirements are broken down into the following tasks:

Memory task	Meaning
recall LTM	recall from long term memory
retain WM	store in working memory
recall WM	recall from working memory
forget WM	erase from working memory

These tasks are then used to create a memory column. The cognition column is annotated with pseudo-code. The user action column, which is the connector to the basic UAN, may contain either UAN task names referring to previously defined macros or explicit UAN notation. Figure 26, taken from Sharratt[26], shows an example of the memory, cognition, and action columns for finding an application on the Macintosh™ desktop.

@@

TASK: Find an Application on the Macintosh™ Desktop		
MEMORY	COGNITION	ACTION
recall LTM(AN;HDI)		
retain WM(AN;HDI)		
recall WM(HDI)		
	Look at (desktop) for (HDI)	
		~[(result of look)]
		Mv^v^
forget WM(HDI)		
recall WM(AN)		
	Look at (window) for (AN)	
	Decide:	
	IF (AN present) THEN (end task) END	
recall LTM (scroll bar &		
elevator)		
	Look at (window) for (scroll bar & elevator)	
	Decide:	
	IF (elevator present and not at bottom of scroll bar) THEN	
retain WM(elevator		
position)		task:Scroll Window
(go to Recall WM(AN))	ELSE	task:SearchFolders
	END	
End Task		

Key: AN - Application Name HDI - Hard Disk Icon
WM - Working Memory LTM - Long Term Memory

Figure 26. Memory, Cognition, and Action Columns of UAN[26]

In this way, the UAN can provide not only a description of tasks the user performs, but also a description of the cognitive processes and memory loading involved in performing those tasks. This information can be very useful in designing user documentation and online help systems.

7 Summary

The techniques introduced here have the capability to represent the behavioral design of direct manipulation user interfaces. The User Action Notation—UAN—is a user-task-oriented representation of user actions. It associates feedback and state changes with the corresponding user actions. This technique is a powerful notation for describing direct manipulation interfaces with precision, clarity, and brevity. This technique does not provide the only solution to the problem of representing direct manipulation, asynchronous interfaces. Rather, it is a framework which interface developers can use to formulate their own solutions. We encourage you to use, adapt, and extend the UAN in any way that best fits your needs.

8 Acknowledgement

The idea of the UAN was originated by Dr. Antonio C. Siochi. This tutorial, and further research and development of the UAN, is currently funded by the National Science Foundation, Dr. John Hestenes, Project Director.

Bibliography

1. Buxton, W. "There's More to Interaction than Meets the Eye: Some Issues in Manual Input." *User Centered System Design.* Norman and Draper ed. Lawrence Erlbaum Associates, New Jersey, 1986.

2. Card, S. K., and Moran, T. P. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Comm. of the ACM. 23* (1980), 396-410.

3. Card, S. K., Moran, T. P., and Newell, A. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

4. Chase, J. D., and Casali, S. P., A Comparison of Three Cursor Control Devices On a Cursor Control Benchmark Task, Technical Report HCI-91-101, January 1991, Industrial and Systems Engineering, VPISU, Blacksburg VA 24061-0118.

5. Draper, S. Personal communication, 1989

6. Fitts, P. M. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology. 47* (1954), 381-391.

7. Flecchia, M., and Bergeron, R. D. Specifying Complex Dialogs in ALGAE. In Proceedings of *CHI+GI Conference on Human Factors in Computing Systems* (Toronto, Apr. 5-9). ACM, New York, 1987, 229-234.

8. Foley, J., Gibbs, C., Kim, W., and Kovacevic, S. A Knowledge-Based User Interface Management System. In Proceedings of *CHI Conference on Human Factors in Computing Systems* (Washington, D. C., May 15-19)., New York, 1988, 67-72.

9. Gould, J. D., and Lewis, C. Designing for Usability: Key Principles and What Designers Think. *Commun. ACM. 28*, 3 (1985), 300-311.

10. Green, M. The University of Alberta User Interface Management System. *Comput. Graph. 19*, 3 (1985), 205-213.

11. Green, M. A Survey of Three Dialog Models. *ACM Trans. Graph. 5*, 3 (July 1986), 244-275.

12. Hartson, H. R., and Gray, P. Temporal Aspects of Tasks in the User Action Notation. , 1992, to appear in Human Computer Interaction, Lawrence Erlbaum Pub.

13. Hartson, H. R., and Hix, D. Toward Empirically Derived Methodologies and Tools for Human-Computer Interface Development. *Int. J. Man-Machine Studies. 31* (1989), 477-494.

14. Hartson, H.R., Siochi, A.C., and Hix, D., The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs, ACM Transactions on Information Systems, 1991.

15. Jacob, R. J. K. "An Executable Specification Technique for Describing Human-Computer Interaction." *Advances in Human-Computer Interaction.* Hartson ed. Ablex, New Jersey, 1985.

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

17. Kieras, D., and Polson, P. G. An Approach to the Formal Analysis of User Complexity. *Int. J. Man-Machine Studies. 22* (1985), 365-394.

18. Moran, T. P. The Command Language Grammar: A Representation for the User Interface of Interactive Computer Systems. *Int. J. Man-Machine Studies. 15* (1981), 3-51.

19. Myers, B. Creating Dynamic Interaction Techniques by Demonstration. In Proceedings of *CHI+GI Conference on Human Factors in Computing Systems* (Toronto, Apr. 5-9). ACM, New York, 1987, 271-278.

20. Norman, D. A. "Cognitive Engineering." *User Centered System Design.* Norman and Draper ed. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

21. Olsen, D. R. MIKE: The Menu Interaction Kontrol Environment. *ACM Trans. on Graph. 5*, 4 (1986), 318-344.

22. Olsen, D. R., Jr., and Dempsey, E. P. Syngraph: A Graphical User Interface Generator. *Comput. Graph. 17*, 3 (1983), 43-50.

23. Payne, S. J., and Green, T. R. G. "Task-Action Grammars: A Model of the Mental Representation of Task Languages." *Human-Computer Interaction.* Lawrence Erlbaum Associates, Inc., Hillsdale, NJ, 1986.

24. Reisner, P. Formal Grammar and Human Factors Design of an Interactive Graphics System. *IEEE Trans. Soft. Eng. SE-7* (1981), 229-240.

25. Richards, J. T., Boies, S. J., and Gould, J. D. Rapid Prototyping and System Development: Examination of an Interface Toolkit for Voice and Telephony Applications. In Proceedings of *CHI Conference on Human Factors in Computing Systems* (Boston, April 13-17). ACM, New York, 1986, 216-220.

26. Sharratt, B., Memory-Cognition-Action Tables: A Pragmatic Approach to

Analytical Modelling. *Human-Computer Interaction - INTERACT '90,* D. Diaper et al. (Editors), Eisevier Science Publishers B.V. (North-Holland), 1990.

27. Shneiderman, B. Multi-Party Grammars and Related Features for Designing Interactive Systems. *IEEE Trans. Syst. Man Cybern. 12*, 2 (Mar.-Apr. 1982), 148-154.

28. Siochi, A. C., and Hartson, H. R. Task-oriented Representation of Asynchronous User Interfaces. In Proceedings of *CHI'89 Conference on Human Factors in Computing Systems* (Austin, Texas, April 30 - May 4). ACM, New York, 1989, 183-188.

29. Siochi, A. C., Hartson, H. R., and Hix, D. *Notational Techniques for Accommodating User Intention Shifts.* TR 90-18, Department of Computer Science, Virginia Polytechnic Institute and State University, 1990.

30. Wasserman, A. I., and Shewmake, D. T. "The Role of Prototypes in the User Software Engineering Methodology." *Advances in Human-Computer Interaction.* Hartson ed. Ablex, Norwood, New Jersey, 1985.

31. Yunten, T., and Hartson, H. R. "A SUPERvisory Methodology And Notation (SUPERMAN) for Human-Computer System Development." *Advances in Human-Computer Interaction.* Hartson ed. Ablex, New Jersey, 1985.

APPENDIX B: NEW UAN TUTORIAL

A Composite Method for Interaction Development Using the User Action Notation

Tutorial

J. D. Chase
chasej@vtcc1.cc.vt.edu
Jeffrey L. Brandenburg
brand@vtcc1.cc.vt.edu
H. Rex Hartson
hartson@cs.vt.edu
Deborah Hix
hix@cs.vt.edu

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061 USA

1. The User Action Notation

The User Action Notation (UAN) is a user- and task-oriented notation that describes the behavior of a user and an interface during their cooperative performance of a task. The primary abstraction of the UAN is a *user task* — a user action or group of temporally related user actions performed to achieve a work goal. A user interface is represented as a quasi-hierarchical structure of tasks that are *asynchronous* — the sequencing within each task is independent of that in the others. User actions, corresponding interface feedback, and state information are presented at the lowest level. Levels of abstraction, where lower level tasks are combined under a single general task name, are used to hide these details and represent the entire interface. At all levels, user actions and user tasks are ordered and combined using temporal relations such as sequencing,

interleaving, and concurrency. Since textual notations are not always convenient for specifying all components of an interface, the UAN includes screen pictures, or scenarios, and can be supplemented with state-transition diagrams to indicate precisely how the user interacts with the interface.

The UAN is primarily a notation for behavioral representation of an interaction design. However, through empirical work with industrial users of the UAN, we have found that it has a variety of uses across the entire interaction development process. As we continue to collect information on how the UAN is being used, a composite, seamless method of organization, representation, and communication has emerged. In this paper we describe and illustrate this method for interaction development using the UAN.

2. The Interaction Development Process

We consider the *interaction development process*— the process for developing the content, appearance, and behavior of a user interface— to incorporate a wide variety of activities and roles, beginning at the recognition of a user need for a system to accomplish a specific task and ending at the time that an acceptable system is delivered to the user. Johnson and Johnson describe activities of *analysis*, *specification*, *development*, and *evaluation* (Johnson, 1988); Hartson and Hix describe *task/functional/user analyses*, *requirements specification*, *conceptual/formal/detailed design*, *rapid prototyping*, *software production*, *evaluation*, and *deployment* (Hartson, et al., 1989; Hix, et al., 1993). We combine these two views with our observations of system development sites currently using the UAN to yield five general activities within the interaction development process:

- Analysis (including task/functional/user analyses)
- Design (including conceptual/formal/detailed and redesign)
- Implementation
- Evaluation (including prototyping and acceptance testing)
- Documentation.

This list includes *documentation* as a distinct activity. We have observed that in some cases the documentation specialist is a key

member of the interface design and development team. Further, documentation is certainly a part of a system with which a user interacts. Therefore, we include documentation development as a component of interaction development.

The following sections illustrate some basic methods of applying the UAN in these activities. We center our discussion of the method and notation around a simple example. More detailed information about temporal aspects of the UAN is available in (Hartson, et al., 1992); an exposition of our view of the development process appears in (Hartson, Brandenburg, & Hix, 1992; Hix, et al., 1993).

3. Analysis

Development of interactive systems begins with analysis, and *task analysis* is one of the most important components of this process. Task analysis is generally defined as the activity of decomposing a user's real-world tasks into ordered sets of subtasks (Olsen, 1988). The UAN, since it is a task-based representation technique and is particularly adept at representing the temporal and hierarchical relationships within and among tasks, provides a mechanism for capturing the results of task analysis.

As an example, consider a simple calendar management system. We assume systems and requirements analysis have determined that the user wants to perform five basic tasks: adding, viewing, modifying, and deleting an appointment on the calendar, and setting an alarm associated with a given appointment. Assume further that we begin with a top-down approach. One possible design at the highest level of abstraction is a repeating disjunction of the five basic tasks (Figure 1).

Task: manage calendar		
User Action	**Feedback**	**Interface State**
OR(add appointment, view appointment, modify appointment, delete appointment, set alarm)*		

Figure 1. Highest level of abstraction for calendar management system

The keyword OR indicates that the user may choose any one of the five tasks. The asterisk (*) after the disjunction indicates that the user may perform this choice zero or more times. (We borrow this notation from the Kleene star closure operator of formal language theory; the UAN also provides a plus (+) operator to indicate that an action must be performed one or more times.) Thus, the description of Figure 1 specifies that the user can perform a sequence of tasks of any length (including zero), with each task selected independently from those specified in the disjunction. Table 1 provides a summary of all of the ways that tasks can be temporally combined in the UAN through either overlapping or combination.

Task Overlapping	And	Inclusive Or	Exclusive Or
None	A single task: A	A single task: A	A single task: A
Sequenced	Two or more tasks written as AND(A, B)	Two or more tasks written as INCOR(A, B)	Two or more tasks written as OR(A, B)
Order Independent	Two or more tasks written as IND/AND(A, B)	Two or more tasks written as IND/INCOR(A, B)	Two or more tasks written as IND/OR(A, B)
Inter-leaved	Two or more tasks written as INTER/AND(A, B)	Two or more tasks written as INTER/INCOR(A, B)	Two or more tasks written as INTER/OR(A, B)
Con-current	Two or more tasks written as CONC/AND(A, B)	Two or more tasks written as CONC/INCOR(A, B)	Two or more tasks written as CONC/OR(A, B)

Task Combination

Table 1. Temporal Notation Combinations in the UAN

Table 1 shows the UAN symbols to represent all of the ways that tasks can be overlapped:

- None— which is the degenerate case of only one task,
- Sequenced— meaning that the tasks will not be overlapped but must be performed in particular order,

- Order Independent— meaning that the tasks will not be overlapped but may be performed in any order,

- Interleaved— meaning that the user may alternate between tasks,

- Concurrent— meaning that the user may perform all tasks simultaneously,

and combined— meaning

- And— meaning that the user must perform all tasks,

- Inclusive Or— meaning that the user must perform at least one of the tasks but may perform any or all of them,

- Exclusive Or— meaning that the user must perform one and only one task.

Thus, for example, a task description written as INTER/AND(A,B) means that the user may alternate between tasks A and B but must eventually complete both tasks.

Returning to our example, the task decomposition of Figure 1 can also be represented as a tree-like hierarchical structure (Figure 2). Such familiar pictorial representations provide a larger global view of the relationships among tasks.

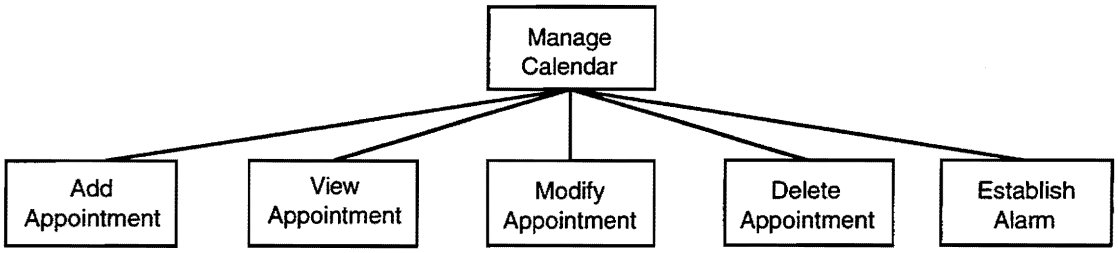


Figure 2. Highest level of hierarchy for calendar management system

To illustrate the next level of abstraction in our example, we expand the **view appointment** task. Assume that the proposed calendar management system must allow the user to view the calendar by month, week, or day, or to search for an individual appointment or time, *before* viewing an individual time slot. (This is analogous to a paper calendar, in which a user can look up a particular month, week, or day, then pick a time within a day to enter or examine an appointment.) The UAN description of **view appointment** (Figure 3) shows how the UAN represents this sequential behavior. The first user action in the description is a repeated disjunction, similar to the **manage calendar** task (Figure 1). The *second* action, **view time slot**, is listed in a separate row *below* the first action; this indicates that the first action must be completed before the second action can take place. (Note, though, that the first action can be repeated *zero* or more times, so it can be "completed" even if the user selects none of the possible choices. In this example, the user might already be at the desired time slot, and so would not need to perform any additional navigation.)

Task: view appointment		
User Action	**Feedback**	**Interface State**
OR(search, view month, view week, view day)*		
view time slot		

Figure 3. UAN description of **view appointment** task for calendar management system

We can expand the UAN hierarchy illustration of Figure 2 to include our elaboration of **view appointment**, yielding the hierarchy of Figure 4. Note that the structure of Figure 4 only represents the hierarchical relationships between the user tasks and not the temporal relationships. The task description of Figure 3, while it contains information about the hierarchical relationships, emphasizes the temporal relationships among the tasks which make up the **view appointment** task. Thus, the task descriptions and the hierarchical structures are complimentary, not redundant.

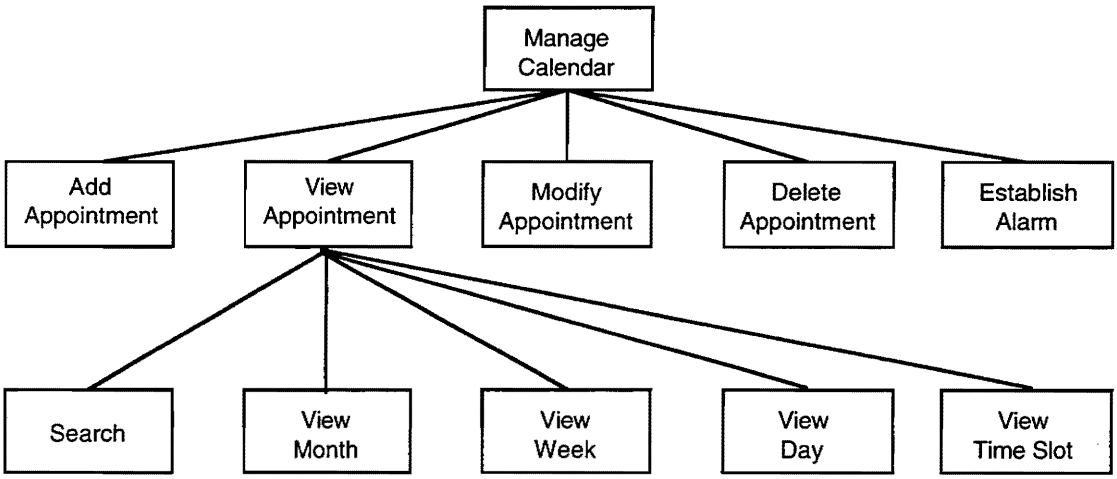


Figure 4. UAN hierarchy of **view appointment** task for calendar management system

For the sake of brevity, we are only expanding one path in the overall hierarchy. We should note that the other paths in the hierarchy would look very similar.

UAN descriptions at these high levels of abstraction specify little or no feedback or state information in response to user actions. In fact, we now have a three-level decomposition of the user's main task (that is, using a calendar) into meaningful and temporally ordered subtasks, yet we have said nothing about the design or the implementation of these tasks. This information begins to appear at the next level of decomposition.

4. Design

At some level in decomposition of the user's tasks, the interaction developer reaches a point where decisions must be made regarding an implementation platform and introduction of interface- or computer-related tasks, rather than tasks residing purely in the problem domain. For example, so far, we have not introduced any tasks that are not part of the user's concept of using a calendar. At some point, we need to introduce tasks that are part of the computer domain — tasks such as activating buttons, selecting from menus, and so forth. In addition, scenarios or screen pictures can make it easier to understand the context of the tasks we are describing. Thus, interaction design joins task analysis very early in the development process.

At this point, task analysis can proceed in a new direction. Instead of continuing from the top down, the developer can build intermediate tasks from the bottom up by composing *primitive* user actions — tasks at the lowest level of abstraction. Our experience with users of the UAN suggest a combination of the two directions is most common. Usually, an interaction style — direct manipulation, command line, voice, etc. — is chosen early in the development process. The interaction style chosen will usually have a set of predefined *artifacts*, i.e. low-level interface components. For example, designers choosing a direct manipulation interaction style immediately know that they have buttons and menus at their disposal. It is through these artifacts that the user and the system communicate during task performance. In turn, the artifacts introduce new tasks — those used to manipulate the artifacts. These tasks describe the actual physical interaction between the user and the system.

We use the term *articulation level* to refer to UAN descriptions that specify physical interactions between the user and the machine. There is a level immediately above this at which tasks are described in terms of artifacts; we call this the *artifact level.* Figure 7, the **select (object)** task a little farther along in our example, is an example of an *artifact level* task which contains *articulation level* descriptions.

Once the readily available artifacts are defined— for some interaction styles these may come from a library of task descriptions for the interaction style while for others it may merely be a matter of drawing on the experience of the designer— by selection of interaction devices and styles, we have observed that designers continue top down until they discover the need to define a new primitive or redefine an existing one. Developers alternate between top down and bottom up in this way until they produce a task hierarchy representing the complete structure of the interface. We illustrate this point by extending our example from the previous section to a lower level.

Figure 5 shows a possible screen design for the calendar management system with a direct manipulation interaction style. This scenario has a label (Scenario 1) associated with it so we can refer to it in the UAN task description.

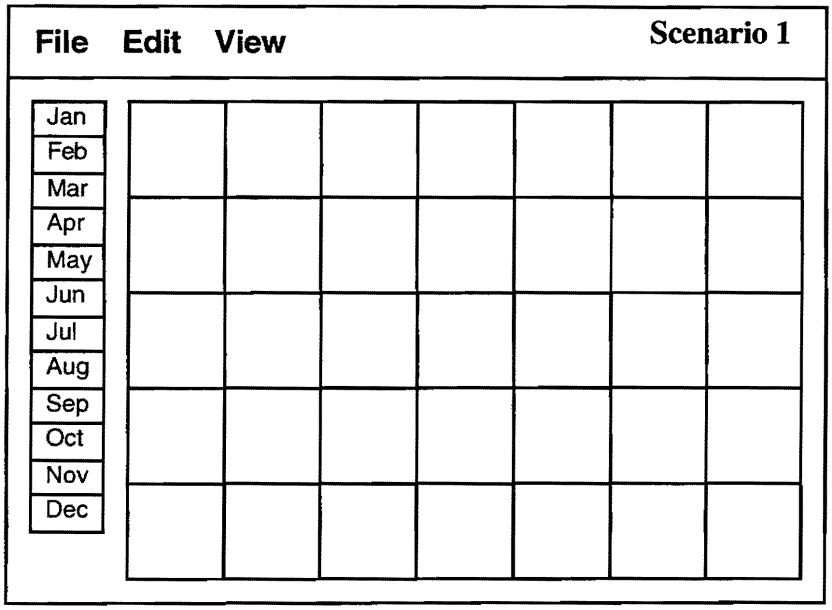


Figure 5. Scenario 1 of calendar management system design.

Figure 6 shows the **view month** task of Figure 3 broken down into more detail. This task description adds an "Arena" label. An *arena* reference in a UAN task description refers to the context in which the task takes place, usually represented by a scenario or screen picture. For instance, file selection in a direct manipulation interface frequently takes place through a dialogue box; this box is the arena for the file selection task. In our example, the arena label specifies that the **view month** task takes place within the screen represented by Scenario 1.

In addition, the **select** task takes a parameter, **any month**. This allows **select** to describe a generic task in which a user selects any given object. Instead of specifying the task in full each time it is used, we simply refer to it by name. We refer to such parameterized tasks as *macro* tasks, since their semantics resemble macro expansion; each occurrence of the formal parameter in the parameterized task is replaced by a copy of the actual parameter supplied by the invoking task.

Task: view month		
Arena: Scenario 1		
User Action	**Feedback**	**Interface State**
OR(select(any month), move forward 1mo, move back 1mo)*		

Figure 6. UAN description of **view month** task for calendar management system

Task: select(object)		
User Action	**Feedback**	**Interface State**
~[object icon'-!]Mv	object icon'! ∀object icon!: object icon-!	selected=object
M^		

Figure 7. UAN description of **select** task for calendar management system

Figure 7 describes the articulation-level **select(object)** task. This task description illustrates how the UAN precisely represents behavior at the articulation level, and also introduces several new symbols.

In the User Action colum, the tilde or ~ symbol represents cursor movement. The brackets or [] symbols around the phrase **object icon'-!** represent the *context* of the icon — the screen area associated with the icon. The prime or ' symbol following **object icon** indicates a *specific* **object icon**, rather than any arbitrary one of the **object icons** that might be on the screen. The exclamation or ! symbol in the UAN represents highlighting; therefore, **-!** represents unhighlighting. The symbol **M** represents the mouse, **v** represents pressing the mouse button and ^ represents releasing it. Thus, the phrase **~[object icon'-!]Mv** means "move the cursor to the context of the unhighlighted object icon and depress the mouse button." The next line, **M^**, means "release the mouse button." Note that temporal sequence is represented by proceeding from left to right, as well as top to bottom — in this example, the user must move the cursor to the icon *before* pressing the mouse button, and both these actions must happen before the mouse button is released.

In the *feedback* column, the phrase **object icon'!** means "highlight the object icon" and ∀**object icon!: object icon-!** means "for all other highlighted object icons, unhighlight them." Moving the **M^** action onto a separate, following line allows us to specify temporal ordering with great precision: feedback is associated with the **Mv** action, *not* with the **M^** action. While this may seem like a minor point, it can actually be very important; recent research has shown that whether the user must depress and release the mouse on a target or just depress the mouse on a target can have a great deal of influence on user performance (Chase & Casali, 1991; Chase, et al., 1992). If details at this level can affect user performance, the designer must be able to accurately and concisely specify such details.

The *interface state* column represents the detail that once the mouse button has been depressed (**Mv**), the currently selected object in the interface is the one that was passed to this task. The interface state column is used to represent any changes which user actions cause in the interface — for instance, changing the "current" mode, window, or selected object, in interface designs that incorporate such concepts. Note also that the parameter is represented by the name "object" while the user action and feedback columns refer to "object icon". This reflects the binding between real objects and interface objects. Real objects are generally represented in the interface by interface objects or icons. Therefore, to manipulate a real object such as a file, the user must act on the *representation* of that object, such as the file's icon. Note that in the interface state column, the thing that is actually selected by this task is the object, not the object icon.

Table 2 provides a summary of useful UAN symbols. The temporal notations described in Table 1 are not repeated in Table 2.

Symbol	Meaning
~	move the cursor
[icon]	the context of object icon, the "handle" by which the icon is manipulated
~[icon]	move cursor into context of the icon
~[object.z]	move the cursor to the context of handle z of the object
~[x,y]	move the cursor to (arbitrary) point x,y outside any object
~[x,y in icon]	move the cursor to (arbitrary) point within icon
~[X in Y]	move to object X within object Y (e.g., [OK_icon in dialogue_box])
[icon]~	move cursor out of context of the icon
object.a:	asks if condition a is true for the object
~[icon]?:	asks if the cursor is within the context of the icon
v	depress
^	release
Xv	depress button, key, or switch called X
X^	release button, key, or switch X
Xv^	idiom for clicking button, key, or switch X
X"abc"	enter literal string, abc, via device X
X(xyz)	enter value for variable xyz via device X
( )	grouping mechanism
*	iterative closure, task is performed zero or more times
+	task is performed one or more times
{ }	enclosed task is optional (performed zero or one time)
;	task interrupt symbol; used to indicate that user may interrupt the current task at this point (the effect of this interrupt is specified as well, otherwise it is undefined, i.e., as though the user never performed the previous actions)
∀	for all
:	separator between condition and action or feedback
<A>	means that task A is atomic or uninterruptible
do until x: A B end do	means repeat the task sequence A, B until condition x is true

while x: do
A
B
end while
	means while condition x is true, do the task sequence A, B

Table 2. UAN symbols and their meanings

Feedback	Meaning
!	highlight object
-!	dehighlight object
!!	same as !, but use an alternative highlight
!-!	blink highlight
$(!-!)^n$	blink highlight n times
@x,y	at point x,y
@icon	at the icon
display(icon)	display the icon
erase(icon)	erase the icon
icon > ~	the icon follows (is dragged by) cursor
icon >> ~	the icon is rubber-banded as its follows cursor
outline(icon)	outline of the icon

Table 2 (continued). UAN symbols and their meanings

Use of the **select()** macro task suggests that objects to be selected may have certain similarities. For example, ∀**object icon!: object icon-!** implies the ability to act or iterate over "all object icons." However, **object icon'!** does not define what it means to highlight an icon. Where there are common attributes among objects, artifact and group or class definitions can provide another form of abstraction and improve locality of definition. For example, consider this partial definition for file icons:

Declaration of artifact class: Icons

Class Includes:
file icons, command icons, . . .

. . .

Attributes:
highlighting mutually exclusive with respect to other objects which are members of this class (i.e. icons) for mouse down without shift

Highlighting:
! means inverse video

. . .

Now the description of Figure 7 can be written as shown in Figure 8. Since the description of icon behavior specifies that highlighting is mutually exclusive for all members of the class of icons, we need not repeat the detail that all other icons are unhighlighted when one icon is highlighted. In this example, we save only one line from one articulation level task; but there may be tens if not hundreds of articulation level tasks for any given task structure. Centralized definition of artifact behavior makes it much easier to keep an interface design consistent, and can considerably reduce the overall size of the interface description. The sample declaration above is rather sparse; interface developers may find it convenient to include more or less information, depending on the particular design task or individual preference.

Task: select(object)		
User Action	**Feedback**	**Interface State**
~[object icon'-!]Mv	object icon'!	selected=object
M^		

Figure 8. Modified UAN description of **select** task for calendar management system

Note that this task has no arena label. The **select(object)** task may be invoked from a number of other tasks, and therefore from a

number of different arenas. For this reason, the arena of a macro task is defined by the invoking task.

The rest of the calendar management example can be expanded in the same way as this one path through the structure. In fact, we have used this example as a exercise in training designers to use the UAN. We encourage UAN users to change or extend the notation to suit their own purposes. For example, while we have written out this example using a terse symbolic notation, some or all of these symbols can be replaced with words and phrases for novice and occasional UAN users. Novice users can also annotate the UAN description with a fourth column containing a prose description of the information on that line. An explanatory *notes* column for the **select(object)** task in Figure 8 might read:

> Move the mouse to the context of the object icon and depress the mouse button, causing the object icon to highlight and causing the associated object to become the currently selected object.

The unwieldiness of this sort of exposition is one reason that the UAN was developed. However, it can be useful for getting novice UAN users familiar with the notation.

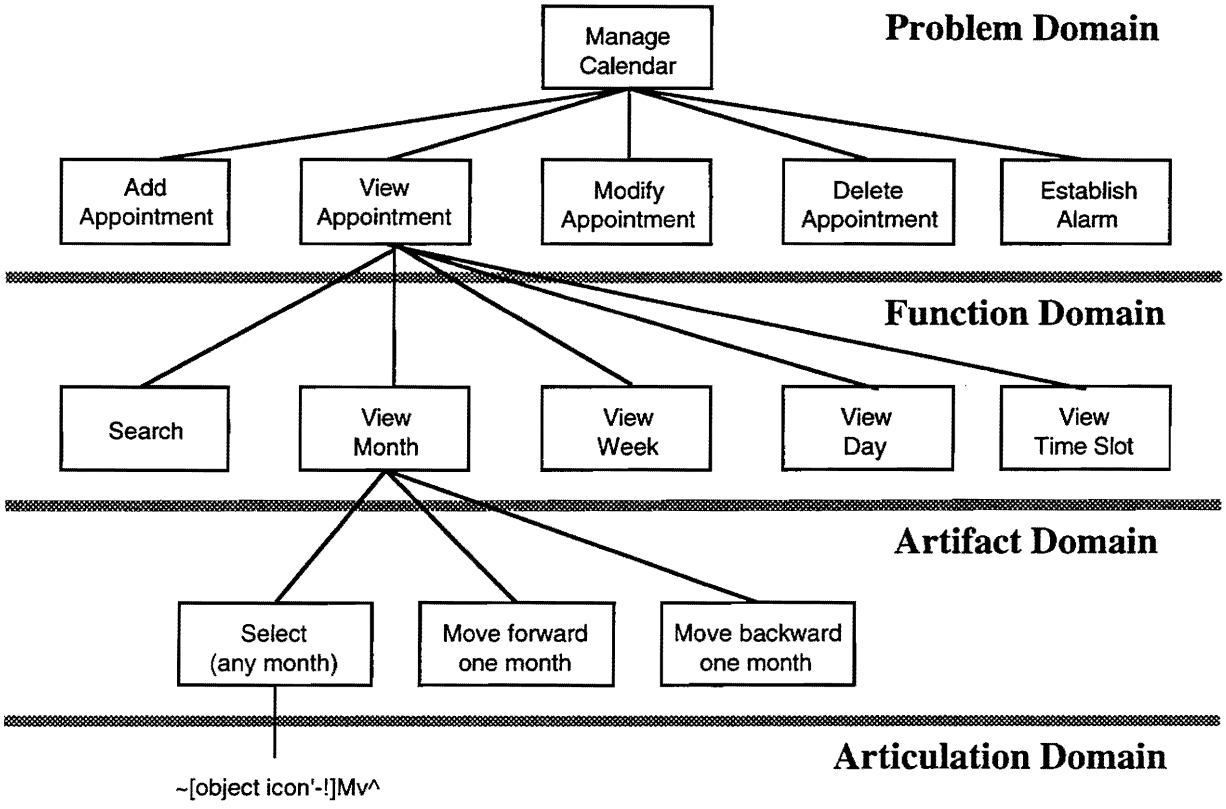


Figure 9. UAN hierarchy of **view month** task for calendar management system, separated by domains

Figure 9 presents the task hierarchy for the part of the calendar management system we have elaborated in this example. As we have descended in this tree structure, we have gone from real-world user tasks relating to real-world entities such as calendars and appointments to a level describing the user's physical interaction with objects in the interface. This seamless transition among domains and levels of abstraction blurs the distinction between task analysis and design.

The lines dividing Figure 9 indicate one view of where the borders between these processes fall. At the beginning of the design process, there is one highest-level task — in our case, to manage a calendar. Next, the design for the major subtasks that can be defined in the *problem domain* begins to take shape, without details of which system functions are used. It may be possible to decompose some of

these tasks further without leaving the problem domain. At some level, though, the subtasks can no longer be decomposed without some idea of the basic set of functions to be offered to the user and how these functions will be invoked.

At this point decisions can be made about the platform on which to implement the target system. In our example, this platform was a direct manipulation system using a mouse. This level includes decisions about artifacts to support the desired interaction in the developing task structure. Once these decisions are made, a variety of *artifact domain* tasks are automatically available to the designer. For example, selection, multiple selection, selecting from a menu, and so forth are all well-defined in most direct manipulation platforms such as the Macintosh, Microsoft Windows, or Motif. These tasks are defined in terms of physical user actions, and these articulation level descriptions constitute the *articulation domain.*

After the high level and low level domains are populated, all that remains is to connect the two. We call the area in which this connection takes place the *function domain.* It is here that the user will invoke and compose system functions to perform work in the problem domain. The user manipulates artifacts to invoke these functions by sequences and combinations of artifact domain tasks.

We have found these four domains sufficient for discussing the task structure of the physical interaction between the user and the system. For other areas of interest, such as cognitive studies, it may be useful to add to or modify these domains. We have developed this view of the UAN's role in task analysis and design empirically through observations of UAN industrial and academic users. While our evaluation continues, responses from our users to date indicate that they consider the development method we describe very powerful and effective.

5. Documentation

The UAN provides a user's view of an interactive system while also providing concise and unambiguous details of the interaction design. This description may be available to a documentation writer well before there is a working system or prototype, allowing an early

start on the documentation process. Like the UAN, most user inquiries are task-oriented (Mirel, Fineberg, & Allmendinger, 1991), making documentation written from the UAN potentially more beneficial to the user than the typical system- or feature-oriented documentation. The hierarchical task structure of the UAN can be translated directly into a documentation outline by performing an in-order traversal of the tree structure (visiting first the parent node and then each of its' children). The task descriptions within the UAN can be translated directly, perhaps eventually automatically, into prose to create a draft user document, saving time and possibly reducing errors in the documentation process. Of course, this draft document only provides content and structure; it still requires a professional writer to edit the document.

6. Future Work

Implementation

One of the original motivations for development of the UAN was the need for a vehicle to communicate interface designs to programmers who implement a system. While it has proven very useful in this capacity, the level of detail and precision available in UAN descriptions suggests that automated processing of such descriptions might facilitate communication, and perhaps even allow at least some automatic implementation.

The UAN is a user- and task-based representation of the interaction design — what we have chosen to call a *behavioral* approach. Implementers, on the other hand, work from a system- and software-based point of view — a *constructional* approach. In the past, the task of transforming UAN descriptions into something approaching an object-oriented description of a system has fallen to implementers. We are now developing more formal approaches to this translation, with the goal of automating large parts of the process.

One technique for easing translation involves building up object descriptions by searching a UAN description for references to an artifact and collecting each behavior mentioned in conjunction with the artifact. In Section 4, we described artifact class declarations,

and showed how they can represent in one location details that would otherwise be restated everywhere an artifact is used. An automated search that collects all behaviors associated with an artifact or artifact class can facilitate the generation of these class declarations. This collection can be done after the UAN description is finished, or while it is being developed. As mentioned previously, these object descriptions can reduce the size of the finished interface specification; since they collect object behavior in one place, they can also help verify consistency. This allows inconsistencies to be corrected in the early stages of design, when corrections are less expensive (since they do not involve reimplementation).

It is important to note that this transformation process must go both ways — UAN to object-oriented description, and object-oriented description to UAN. The implementation process can reveal changes that must be made in the interface design, and it must be possible to state these changes in behavioral terms so interface designers can approve them. It is also important to have accurate UAN descriptions for activities that follow the implementation process.

Evaluation

The UAN provides two avenues of benefit in the process of evaluating the interaction component of a software system. First, a UAN specification is a step by step description of how an interactive system should behave in response to specific user actions under specific conditions. For this reason, it is an excellent script for system behavioral testing. This process helps ensure that the system behaves as it was designed to behave in all circumstances.

Second, since it is a user- and task-based representation, the UAN provides an excellent basis for usability testing. We have shown previously that the UAN, with its detailed description of interaction between the user and the system, can provide a basis for performance prediction (Chase, et al., 1992). This can help predict expected user performance before usability testing begins. Further, UAN descriptions are already decomposed into meaningful, discrete task descriptions, providing a logical decomposition for testing.

Conclusion

The UAN was originally developed as a way to record and communicate interface designs. Our observations of industrial and academic users of the UAN, in conjunction with our own ongoing use of the notation, have demonstrated that it is useful over a much broader scope. It supports *task-based analysis*, through its own task-oriented structure; *design*, through its ability to represent all levels of abstraction from high-level tasks to low-level physical behaviors; *implementation*, through its detailed representation of individual user and system actions at the lowest level; *evaluation*, through its complete and step-by-step representation of user actions and system responses; and *documentation* of interactive systems, through its user orientation and its representation of overall interface structure.

Acknowledgements

The calendar management example is derived from a much more extensive presentation in (Hix, et al., 1993). Portions of this research were funded by a grant from the National Science Foundation. We also wish to acknowledge Dr. Antonio Siochi, the originator of the UAN.

References

Chase, J. D. & Casali, S. P. (1991). *A Comparison of Three Cursor Control Devices on a Cursor Control Benchmark Task* (TR 91-8). Virginia Tech.

Chase, J. D., Casali, S. P., & Hartson, H. R. (1992). The Predictability of Cursor Control Device Performance Based on a Primitive Set of User Object-Oriented cursor Actions. *Proceedings of Human Factors Society 36th Annual Meeting*, Santa Monica: Human Factors Society, 306-310.

Hartson, H. R., Brandenburg, J. L., & Hix, D. (1992). Different Languages for Different Development Activities: Behavioral Representation Techniques for User Interface Design. In B. A. Myers (Ed.), *Languages for Developing User Interfaces* (pp. 303-326). Boston: Jones & Bartlett.

Hartson, H. R. & Gray, P. D. (1992). Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction*, *7*, 1-45.

Hartson, H. R. & Hix, D. (1989). Toward Empirically Derived Methodologies and Tools for Human-Computer Interface Development. *International Journal of Man-Machine Studies, 31*, 477-494.

Hix, D. & Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability Through Product and Process*. New York: John Wiley & Sons, Inc.

Johnson, P., and Johnson, H. (1988). Practical and Theoretical Aspects of Human Computer Interaction. *JIT, 3*(3),

Mirel, B., Fineberg, S., & Allmendinger, L. (1991). Designing Manuals for Active Learning Styles. *38*(1), 85-87.

Olsen, J. R. (1988). Cognitive Analysis of People's Use of Software. In J. M. Carroll (Ed.), *Interfacing Thought* (pp. 260-293). London: MIT Press.

APPENDIX C: RELIABILITY EXPERIMENT MATERIALS

Subject Informed Consent Form

An Experiment to Evaluate an Empirically Derived Model of Behavioral Representation Techniques

Principle Investigator: Joseph D. Chase
Ph.D. Candidate
Dept. of Computer Science

I. The Purpose of This Research

You are invited to participate in a study about the structure of design representation techniques. This study involves experimentation for the purpose of evaluating a proposed model of behavioral representation techniques. This study involves 3 subjects in addition to yourself.

II. Procedures

You will be provided with a copy of the proposed model and a list of critical incidents which have been reported by users of behavioral representation techniques. Your first task is to place each of these incidents within the model. Do so by recording the number of the incident in the cell in which it best fits. Some incidents may appear to fit in a variety of places in the model. You should pick the cell in the model that best describes the incident. Some cells may have more than one incident and some cells may have none. This task should require no more than 1 hour.

Your second task is to go through each cell/intersection in the model and rate the UAN as it applies to that cell on a scale of 1 to 3. For example, since one of the intersections in the model represents the accuracy of feedback notation in the design activity, you will be asked to rate the UAN as to how accurately it can represent system feedback in the design activity with 1 meaning virtually not at all, 2 meaning not very accurately, and 3 meaning very accurately. You

should be aware that you may have a tendency to avoid the extremes and rate every category as a 2. Please be aware of this tendency and try to be as accurate as possible. This task should also take no more than 1 hour to complete.

You may take these materials with you and complete these tasks at your leisure. Please refer any questions that you may have while completing these tasks only to the principle investigator. Do not discuss these tasks with anyone other than the principle investigator either during or after your participation as this experiment may be repeated using other subjects in the future.

III. Benefits of This Project

Your participation in this project will provide valuable insight into the reliability of the proposed model for both critical incident evaluation and analysis of behavioral representation techniques.

You may receive a summary of this research by requesting one from the principle investigator.

IV. Anonymity and Confidentiality

The results of this study will be kept strictly confidential. At no time will the researchers release the results of the study to anyone other than individuals working on the project without your written consent. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research.

V. Compensation

There will be no compensation for participating in this study. Your willingness to volounteer without compensation is greatly appreciated.

VI. Freedom to Withdraw

You are free to withdraw from this study at any time without penalty of any kind.

VII. Approval of Research

This research project has been approved, as required, by the Institutional Review Board for projects involving human subjects at Virginia Polytechnic Institute and State University and by the Department of Computer Science.

VIII. Subject's Responsibilities

I know of no reason I cannot participate in this study.

_____
Signature

IX. Subject's Permission

I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project.

If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project.

Should I have any questions about this research or its conduct, I will contact:

Joseph D. Chase Investigator	231-6470
H. Rex Hartson Advisor	231-4857
Chair, IRB Research Division	231-6077

Model Evaluation Incidents List

Please write the number of each incident in the cell in the model in which it best fits. Remember, incidents may be negative reports, e.g. a failure of the representation in some way, or they may be positive reports, e.g. a positive effect of the representation. Each of these incidents should take no more than a minute or two to place. If you find yourself spending more time on them than that, you may be looking to deeply into the wording. Some of the incidents may appear to fit in more than one cell. Please place them in the one that seems to fit the best. Be aware that phrases such "given interactive system" do not necessarily mean that the system has already been developed.

1. An individual attempting to record the cursor manipulation required to operate a given system found no notation available to represent the fact that this particular system required manipulation of two distinct cursors.

2. An individual attempting to record the cursor manipulation required to operate a given system found no notation available to represent the condition that the cursor must be within a particular icon before the user may continue.

3. An individual attempting to record the cursor manipulation required to operate a given system found no notation available to represent the idea that the user must move iconx until it overlaps with icony.

4. An individual attempting to record information about a system requested a way to record information about the size, shape, color, etc. of objects, and icons on the screen.

5. An individual attempting to record information about a system found no notation to represent the idea that some objects have multiple handles.

6. An individual attempting to record information about user tasks within a given system found the notation to represent the idea that one or more of those tasks are atomic, i.e. uninterruptible, difficult to use and understand.

7. An individual attempting to record information about user tasks realized that what he was doing was recording what the user should do, rather than what the user could do. He became concerned because there was nothing available to him to represent what the user could do, i.e. things like moving outside of a menu without selecting anyting.

8. An individual reported that the use of x and x' when representing user cursor manipulation was inconsistent with the use of x and x' in other disciplines. We were using x to mean a specific value and x' to mean a general class of values where as mathematics and statistics use x' to mean a specific value.

9. An individual trying to record information about the users real-world task found that he needed a way of seeing an overall view of all of the tasks and which tasks invoked others, etc.

10. An individual trying to record information about a given system found no notation to represent the idea that the user should make the cursor leave the context of a particular icon.

11. An individual trying to check a system against its recorded design with regard to user actions, found that it would be nice if there were some way to analyze the recorded information with respect to completeness, i.e. is this information a complete description of the given interface.

12. An individual working to record information about a users real-world tasks with a particular notation found that she was having difficulty remembering what all of the symbols represented and decided to use words and phrases instead of symbols to represent user cursor manipulation.

13. A programmer attempting to understand information which had been recorded about a given system found that he needed an object oriented description of the system rather than the user- and task- oriented description that he had been given.

14. An individual attempting to record information about a users real-world tasks found that there was no perscribed process by which to get and record that information.

15. An individual attempting to record information about the tasks in an interactive system wanted to find a way to record the context of each task, i.e. the screen or screen region in which that task takes place.

16. An individual attempting to describe a users real-world tasks found that there was no notation available to describe the dependencies and inter-relationships between two highly concurrent tasks.

17. An individual attempting to record information about a given interactive system found that there was no notation available to describe the initial state of the interface.

18. An individual attempting to record information about a given interactive system found that there was no notation available to represent the knowledge that a user would require in order to operate the given system.

19. An individual attempting to record information about a given interactive system found that there was no notation available to represent the users memory load while interacting with the system.

20. A programmer attempting to understand the information given him about the system responses within a given interactive system complained bitterly that the notation was cryptic and thus of little use.

21. A programmer attempting to understand the information given him about the user input to a given interactive system complained that if the information were complete there ought to be a way of automatically deriving or compiling the interface from the description.

22. An individual using recorded information to check an interface to make sure that the system responds as the recorded information says that it should complained that if the information is complete then there ought to be a way of having the computer check the interface against the description to see if the interface is correct.

23. An individual recording information about a users real-world tasks found that a symbol meaning INCLUSIVE OR or choose one or more of the actions in this group was needed and not available.

24. An individual working on the users manual for the user activity in a given interactive system invented a process for automatically deriving a draft user document from a UAN description of a system.

25. A programmer trying to understand the information given him about a given interactive system complained that there ought to be a way to automatically check that a given objects behavior is consistent across all of the tasks in a system.

26. An individual working on the users manual for a given system realized that different tasks within the system were designed for specific groups of users and found the notation available for defining the differences in these groups of users very difficult to learn and use.

27. An individual working on the users manual for a given interactive system discovered that there was a natural translation between the screen pictures needed in the design of the system and those needed in the users manual.

28. An individual recording information of the cursor manipulation within a given system found that he was working with a new 3 dimensional input device which was not describable using any of the currently available notations.

29. An individual checking user performance on a new system found that there was no notation for recording this user performance information.

30. An individual checking user performance on a new system found that a user could be tested on only a few tasks within the system and then those results used to automatically project user performance on the whole system.

31. An individual working on the users manual for a given interactive system found that the notation recording information about the ordering and combination of tasks within the system was cryptic and difficult to read and understand.

32. A programmer working on a new system found that the information regarding input devices which had been recorded for this system was too ambiguous for him to determine which devices or even how many devices to allow for.

33. A programmer working on a new system found it difficult to determine from the notational description which tasks were invoked from other tasks.

34. A researcher suggested that if the notational description of the users memory load activity were sufficient, an automated analysis could report whether a system being reviewed was over taxing its users.

35. An individual working on the users manual for a given system finds that the notation for and the general concept of things that a user can do within a task that are not actually a necessary part of that task very confusing and difficult to learn and pass on.

36. A programmer working a new system found that the notation that he was given describing the user inputs to the system contained many symbols which had conflicting meanings with those that he used in programming and software design.

37. An individual working on recording state information about a new system determined that if the state information were complete an automated translation to state-transition diagrams could be achieved in order to communicate with individuals who prefer that notation.

38. A programmer working on a new system discovered that if the scenario package for a new system were complete, it could automatically be translated into a stubbed prototype.

39. An individual working on the user manuals for a given system complained that if the the user class descriptions were sufficiently detailed in terms of prior user knowledge, then they could be used to automatically translate a single user document into a tutorial for novices, a users guide for intermediate users, and a reference for experts.

40. An individual checking a systems responses to make sure that the system behaves as designed complained that the design notation was not sufficiently precise so as to always determine what the appropriate response should be.

APPENDIX D: UTILITY EXPERIMENT MATERIALS

Subject Informed Consent Form

An Experiment to Evaluate the Improvement in the User Action Notation

Principle Investigator: Joseph D. Chase
Ph.D. Candidate
Dept. of Computer Science

I. The Purpose of This Research

You are invited to participate in a study about the structure of design representation techniques. This study involves experimentation for the purpose of evaluating a proposed model of behavioral representation techniques. This study involves 5 subjects in addition to yourself.

II. Procedures

You will be asked to write a User Action Notation(UAN) description for a simple user task. You will then be grouped with two other subjects into a design team. You will be provided with a copy of the proposed model and documentation for the User Action Notation(UAN). After a fifteen minute question and answer session with the experimenter, you will then be given a prototype of an interactive system on the Macintosh. Your group will then write a UAN description of this prototype using the documentation provided. This task should require no more than 4 hours.

Your second task is to go through each cell/intersection in the model and rate the UAN as it applies to that cell on a scale of 1 to 3. For example, since one of the intersections in the model represents the accuracy of feedback notation in the design activity, you will be asked to rate the UAN as to how accurately it can represent system feedback in the design activity with 1 meaning virtually not at all, 2 meaning not very accurately, and 3 meaning very accurately. Your should be aware that you may have a tendency to avoid the

extremes and rate every category as a 2. Please be aware of this tendency and try to be as accurate as possible. This task should take no more than 1/2 hour to complete.

Please refer any questions that you may have while completing these tasks only to the principle investigator.

III. Benefits of This Project

Your participation in this project will provide valuable insight into the utility of the proposed model of behavioral representation techniques.

You may receive a summary of this research by requesting one from the principle investigator.

IV. Anonymity and Confidentiality

The results of this study will be kept strictly confidential. At no time will the researchers release the results of the study to anyone other than individuals working on the project without your written consent. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research.

V. Compensation

There will be compensation of $25 for participating in this study. Your willingness to volunteer is greatly appreciated.

VI. Freedom to Withdraw

You are free to withdraw from this study at any time without penalty of any kind. You will be paid $5/hour for every hour you have participated prior to withdrawing. This total shall not exceed $25.

VII. Approval of Research

This research project has been approved, as required, by the

Institutional Review Board for projects involving human subjects at Virginia Polytechnic Institute and State University and by the Department of Computer Science.

VIII. Subject's Responsibilities

I know of no reason I cannot participate in this study.

_____
Signature

IX. Subject's Permission

I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project.

If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project.

Should I have any questions about this research or its conduct, I will contact:

Joseph D. Chase Investigator	231-6470
H. Rex Hartson Advisor	231-4857
Chair, IRB Research Division	231-6077

Subject Informed Consent Form

An Experiment to Evaluate the Improvement in the User Action Notation

Principle Investigator: Joseph D. Chase
Ph.D. Candidate
Dept. of Computer Science

I. The Purpose of This Research

You are invited to participate in a study about the structure of design representation techniques. This study involves experimentation for the purpose of evaluating a proposed model of behavioral representation techniques. This study involves 3 subjects in addition to yourself.

II. Procedures

You will be asked to compare a User Action Notation(UAN) description for a simple user interface with a prototype of that interface and rate the description using a simple survey.

Your task is to go through each cell/intersection in the survey and rate the UAN description of the prototype as it applies to that cell on a scale of 1 to 3. For example, since one of the intersections in the model represents the accuracy of feedback notation, you will be asked to rate the UAN description as to how accurately it represents system feedback in the prototype with 1 meaning virtually not at all, 2 meaning not very accurately, and 3 meaning very accurately. You should be aware that you may have a tendency to avoid the extremes and rate every category as a 2. Please be aware of this tendency and try to be as accurate as possible. This task should take no more than 1/2 hour to complete. You will then be asked to repeat this task with a second description of the same interface.

Please refer any questions that you may have while completing these tasks only to the principle investigator.

III. Benefits of This Project

Your participation in this project will provide valuable insight into the utility of the proposed model of behavioral representation techniques.

You may receive a summary of this research by requesting one from the principle investigator.

IV. Anonymity and Confidentiality

The results of this study will be kept strictly confidential. At no time will the researchers release the results of the study to anyone other than individuals working on the project without your written consent. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research.

V. Compensation

Your willingness to volunteer is greatly appreciated.

VI. Freedom to Withdraw

You are free to withdraw from this study at any time without penalty of any kind.

VII. Approval of Research

This research project has been approved, as required, by the Institutional Review Board for projects involving human subjects at Virginia Polytechnic Institute and State University and by the Department of Computer Science.

VIII. Subject's Responsibilities

I know of no reason I cannot participate in this study.

_____

Signature

IX. Subject's Permission

I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project.

If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project.

Should I have any questions about this research or its conduct, I will contact:

Joseph D. Chase Investigator	231-6470
H. Rex Hartson Advisor	231-4857
Chair, IRB Research Division	231-6077

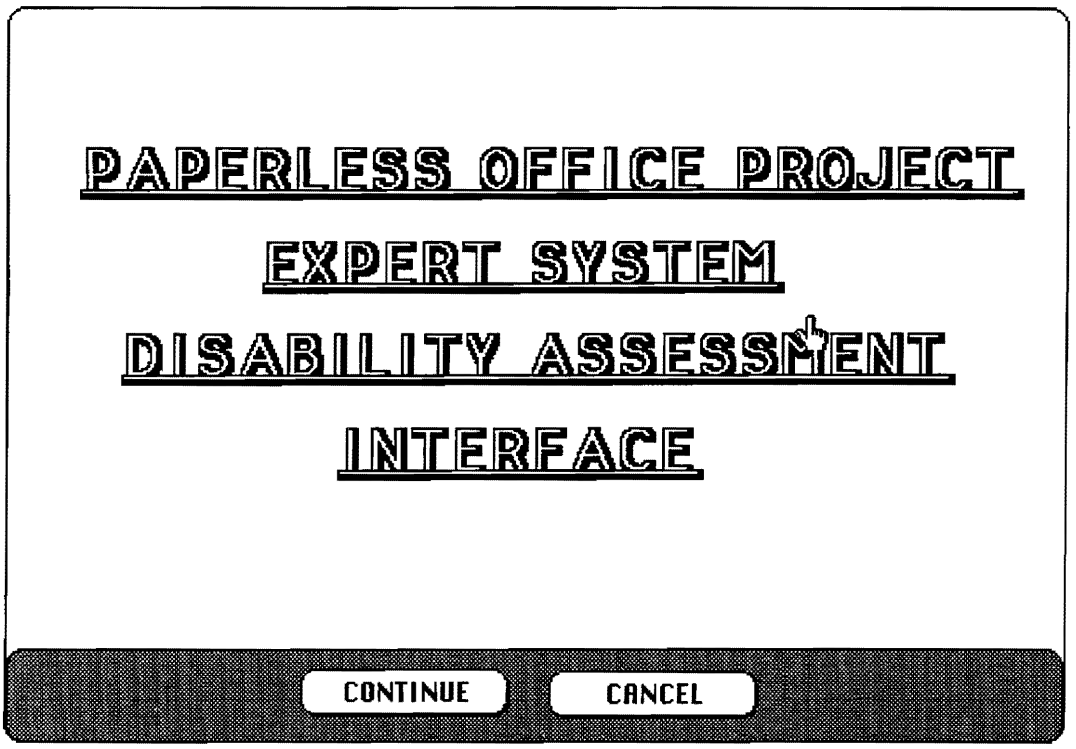
Subject evaluation test for Utiltiy experiment

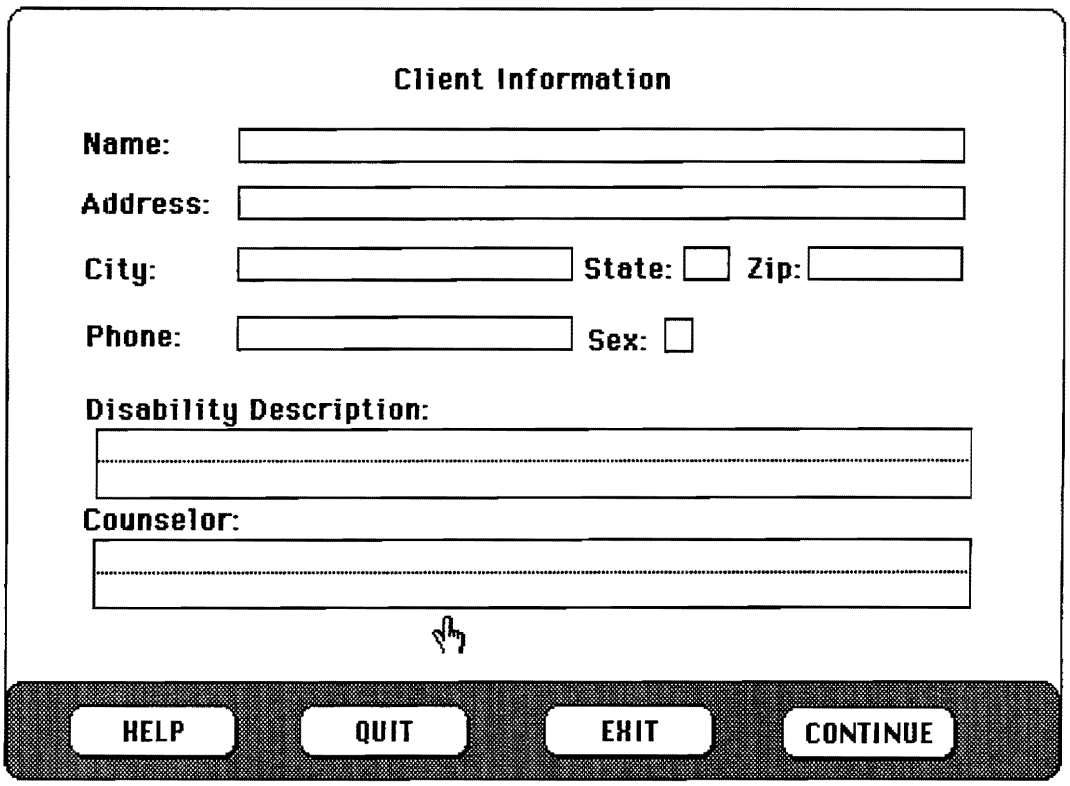
Please print your name: _____

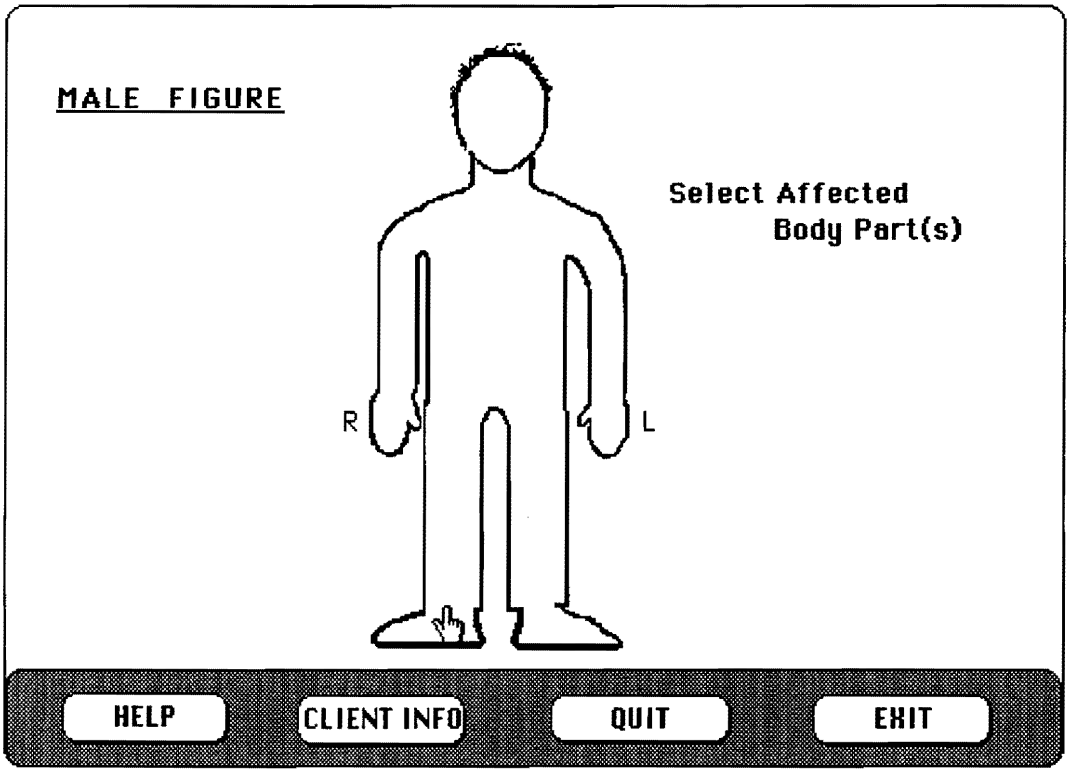
Write a brief UAN task description to select a generic object, i.e. use a parameter. This objects icon should hilight upon selection and all other icons should unhilight at the same time. You may or may not need all of the space provided.

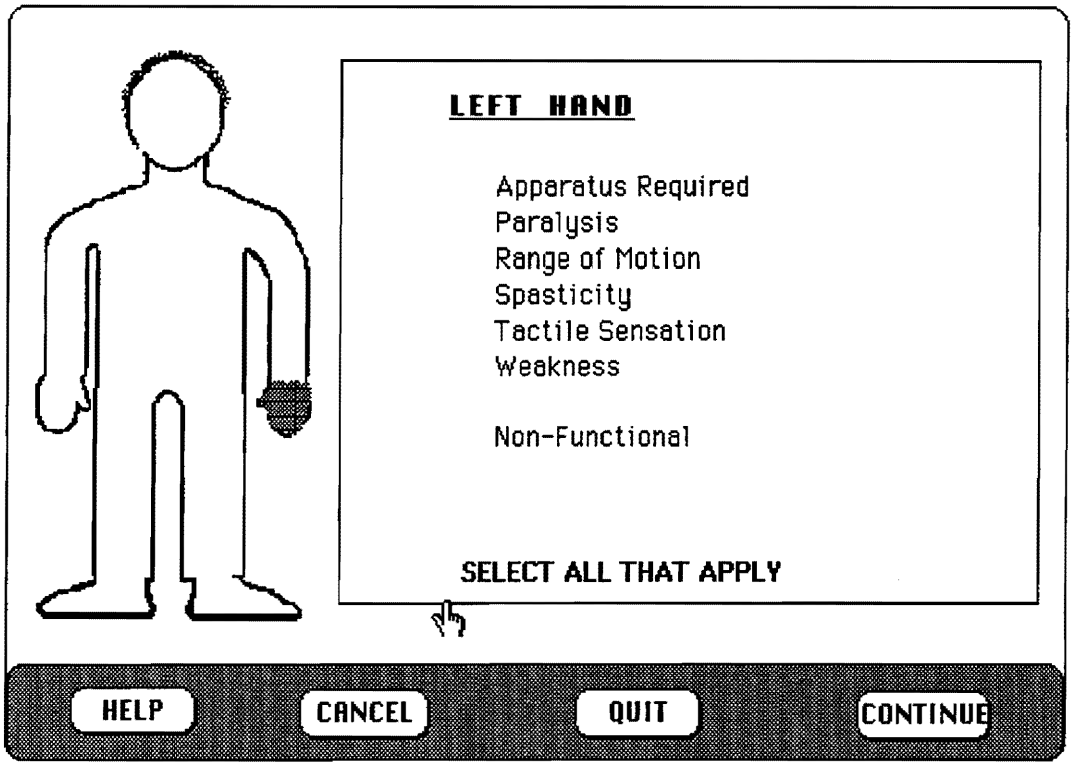
Please print your current QCA: ____ (remember, this information will not be viewed by anyone other than the principle investigator)

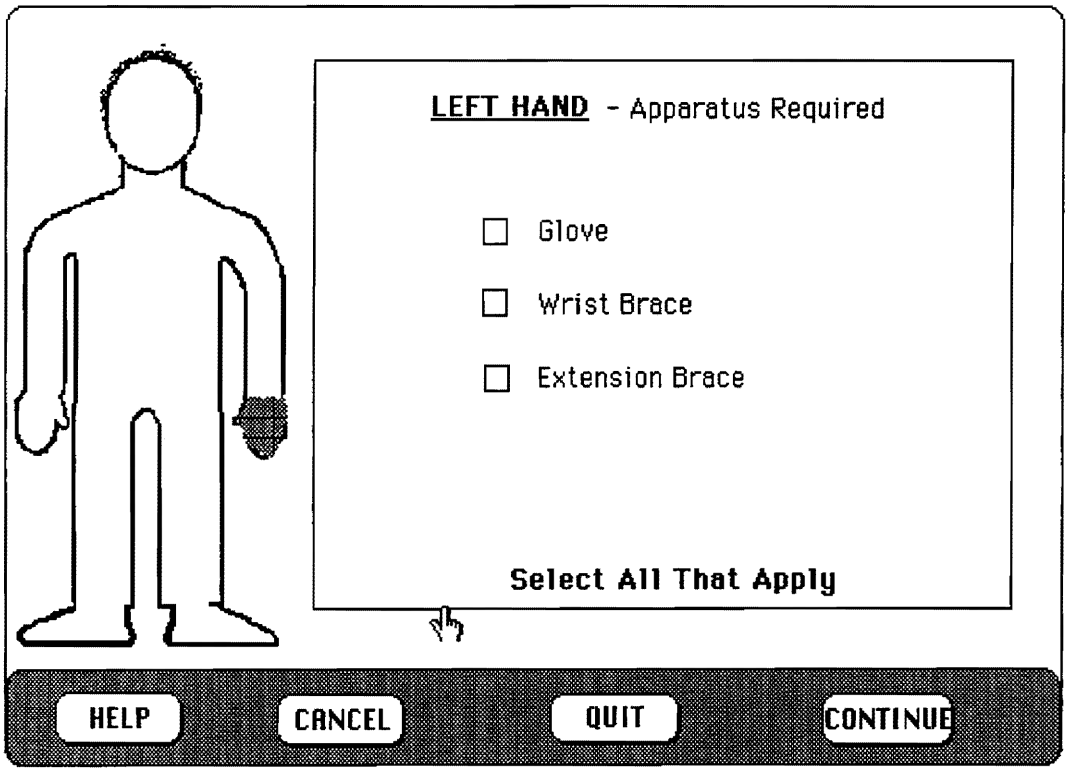
Interface Prototype for Utility Experiment

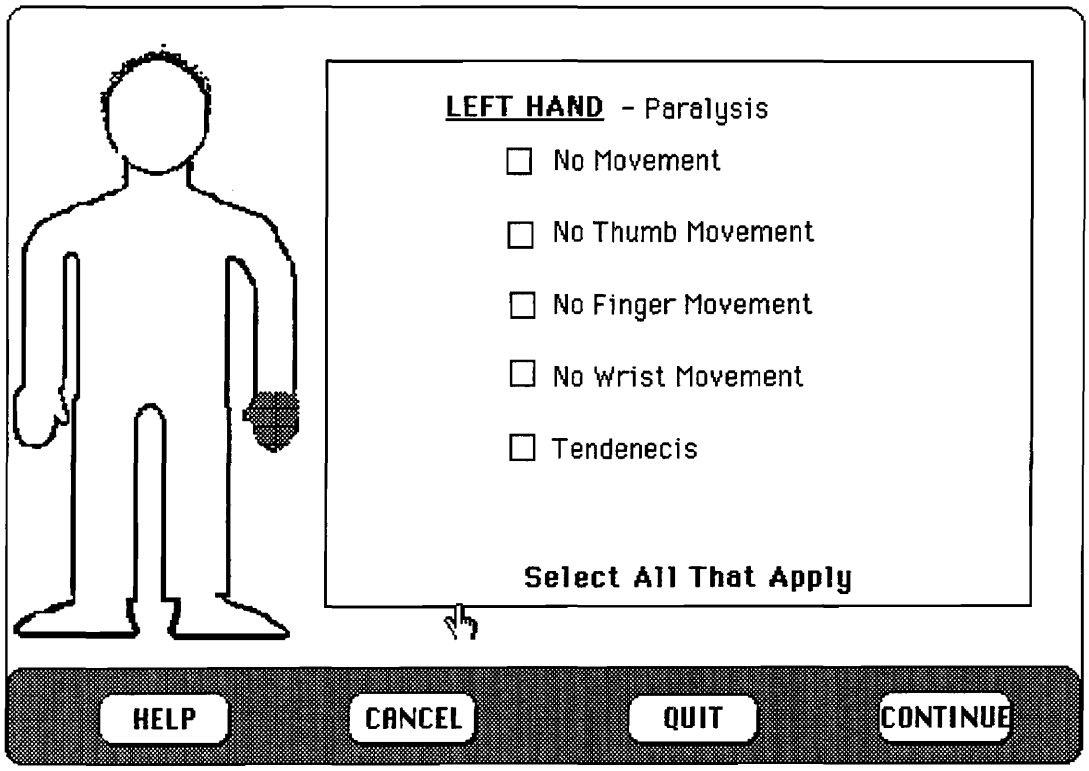


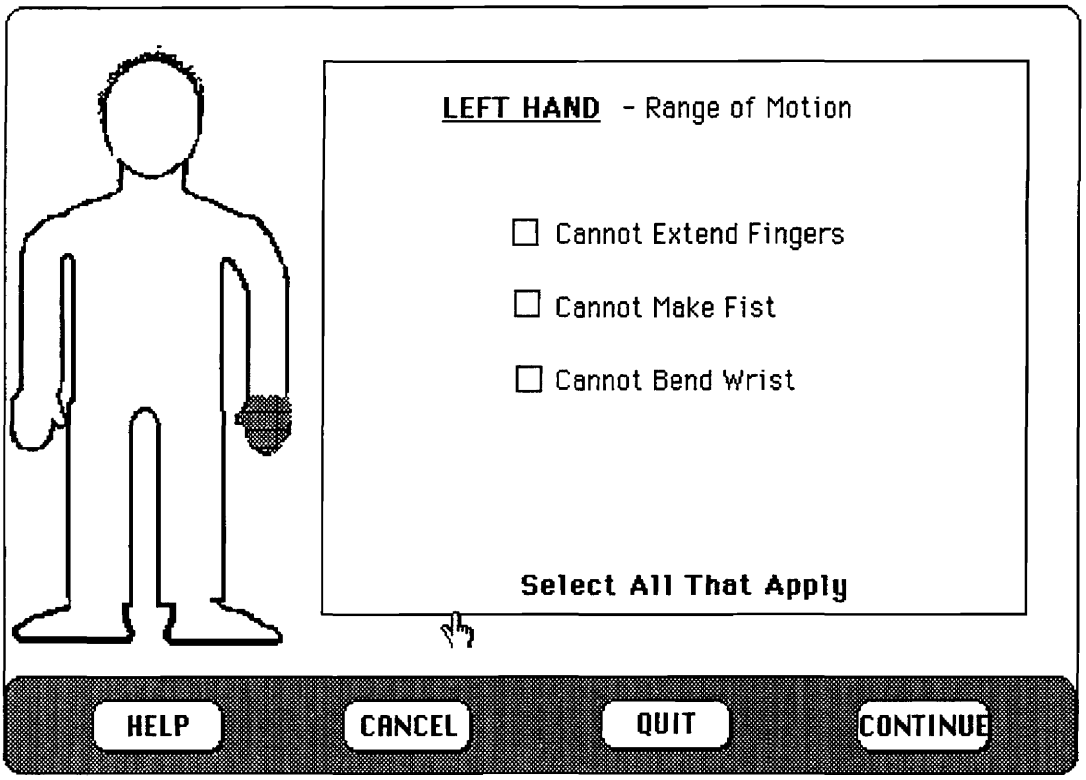

Client Information
Name:
Address:
City:
State:
Zip:
Phone:
Sex:
Disability Description:
Counselor:
HELP
QUIT
EXIT
CONTINUE

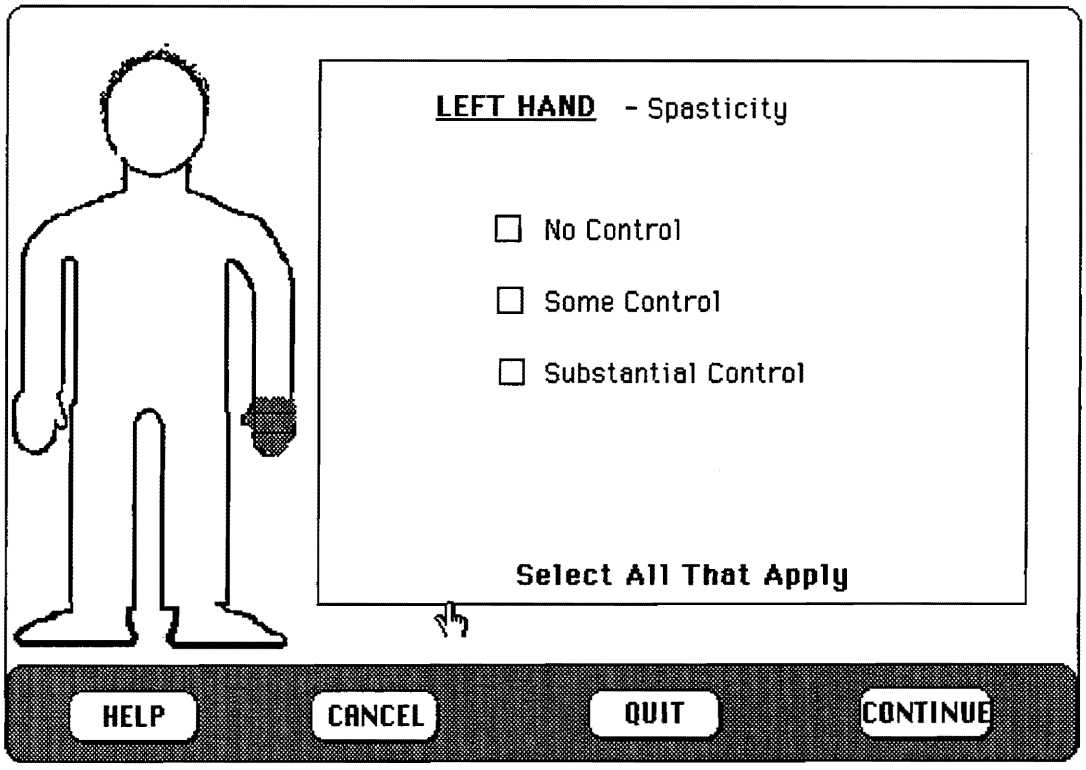

MALE FIGURE
Select Affected
Body Part(s)
R
L
HELP
CLIENT INFO
QUIT
EXIT

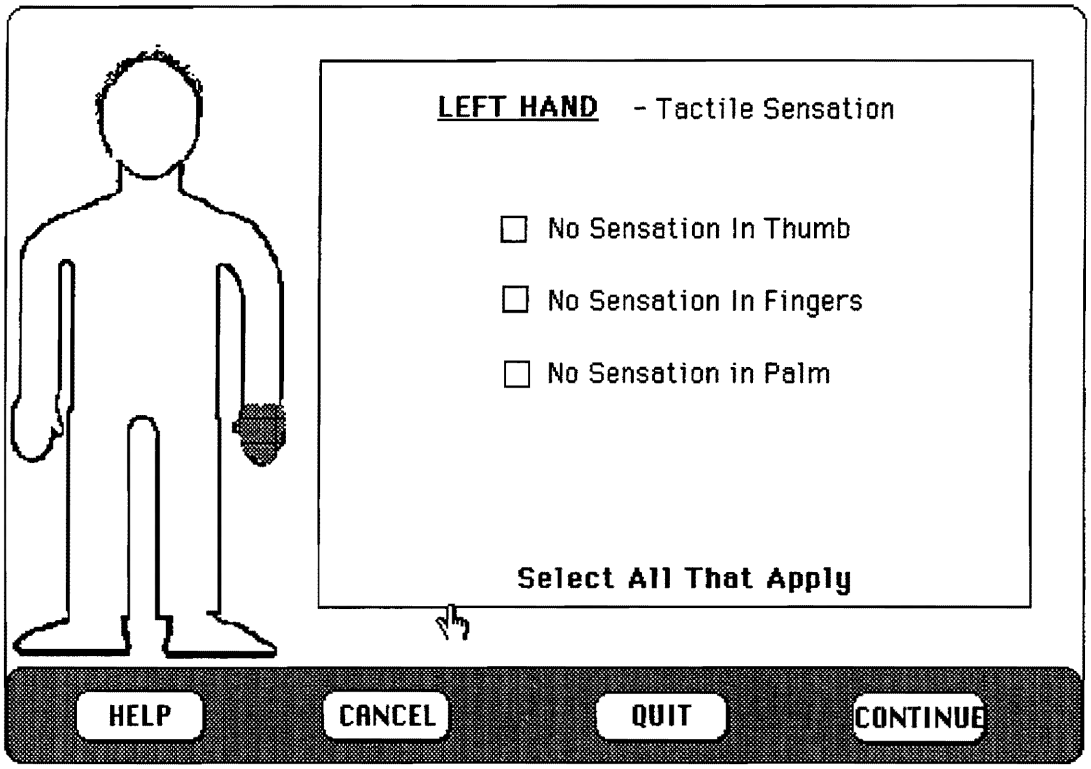

LEFT HAND
Apparatus Required
Paralysis
Range of Motion
Spasticity
Tactile Sensation
Weakness
Non-Functional
SELECT ALL THAT APPLY
HELP
CANCEL
QUIT
CONTINUE

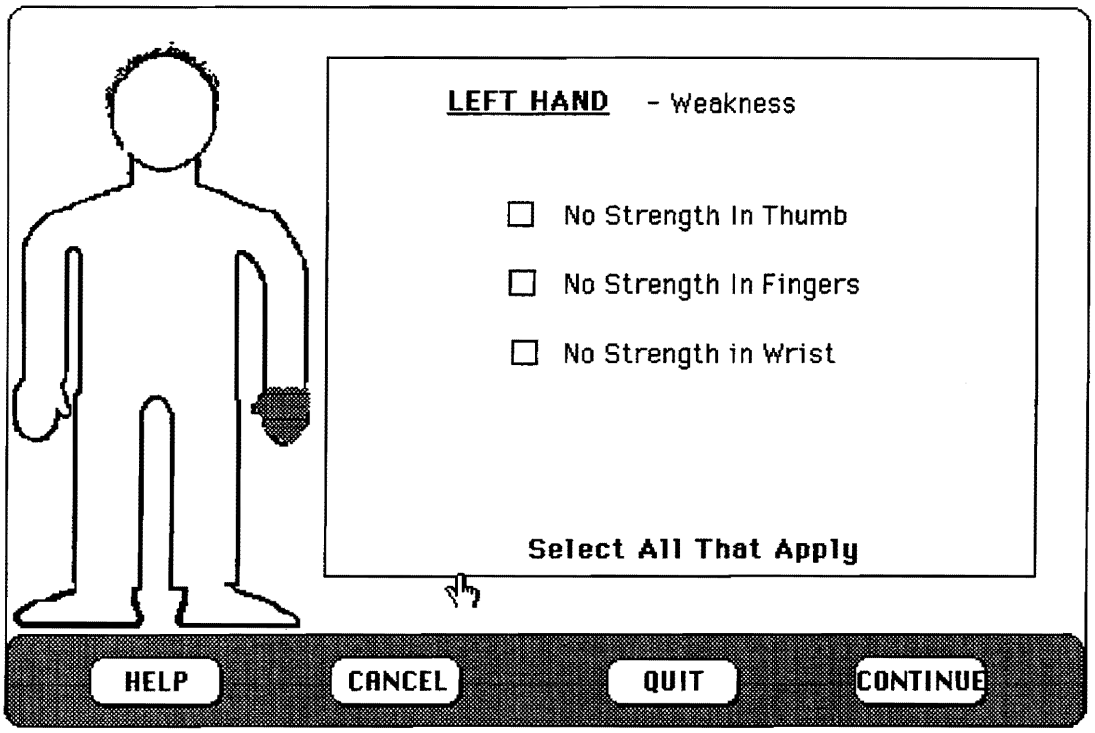

LEFT HAND - Apparatus Required
Glove
Wrist Brace
Extension Brace
Select All That Apply
HELP
CANCEL
QUIT
CONTINUE

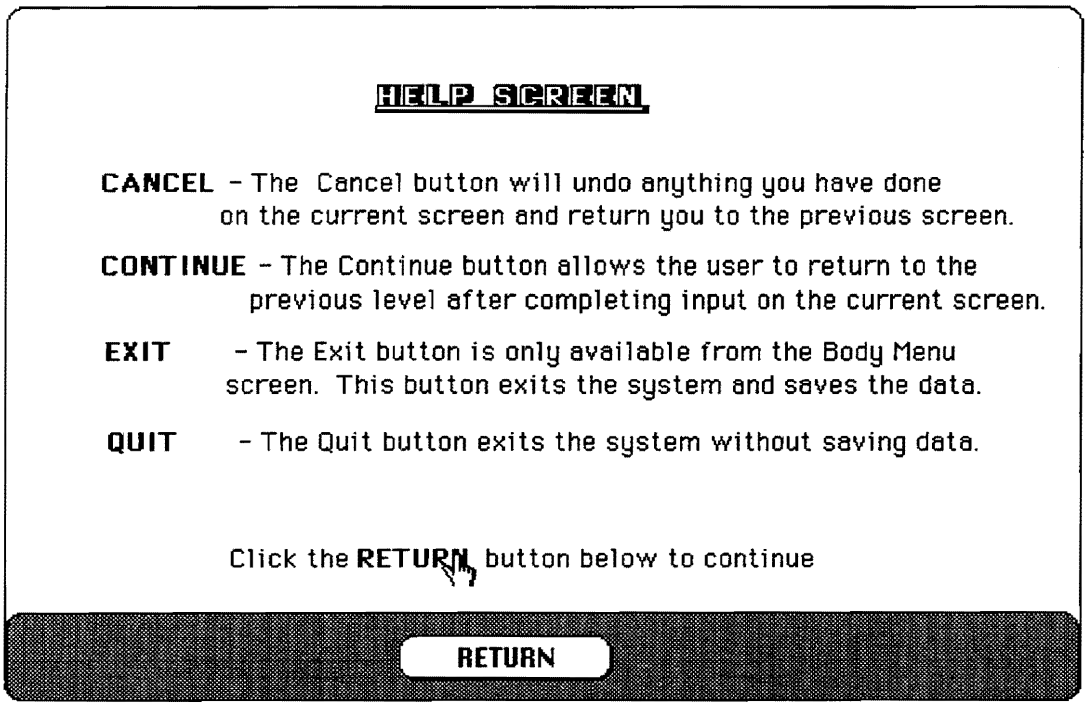

LEFT HAND - Paralysis
No Movement
No Thumb Movement
No Finger Movement
No Wrist Movement
Tendenecis
Select All That Apply
HELP
CANCEL
QUIT
CONTINUE

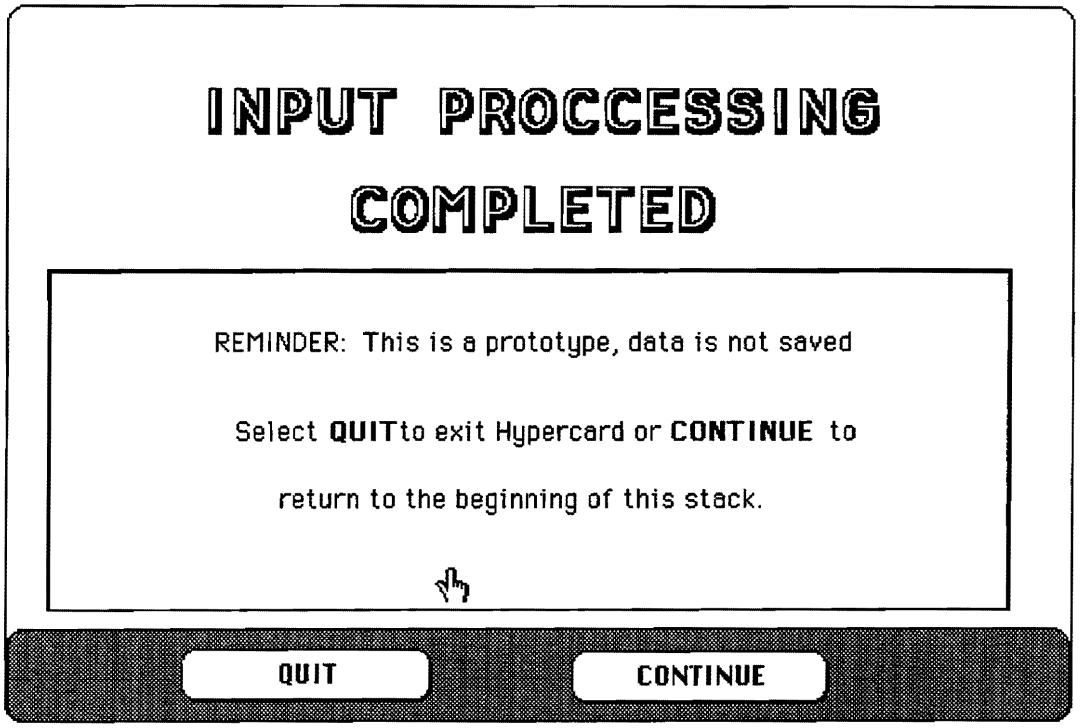

LEFT HAND - Range of Motion
Cannot Extend Fingers
Cannot Make Fist
Cannot Bend Wrist
Select All That Apply
HELP
CANCEL
QUIT
CONTINUE


LEFT HAND - Spasticity
No Control
Some Control
Substantial Control
Select All That Apply
HELP
CANCEL
QUIT
CONTINUE


LEFT HAND - Tactile Sensation
No Sensation In Thumb
No Sensation In Fingers
No Sensation in Palm
Select All That Apply
HELP
CANCEL
QUIT
CONTINUE


LEFT HAND - Weakness
No Strength In Thumb
No Strength In Fingers
No Strength in Wrist
Select All That Apply
HELP
CANCEL
QUIT
CONTINUE


HELP SCREEN
CANCEL - The Cancel button will undo anything you have done on the current screen and return you to the previous screen.
CONTINUE - The Continue button allows the user to return to the previous level after completing input on the current screen.
EXIT - The Exit button is only available from the Body Menu screen. This button exits the system and saves the data.
QUIT - The Quit button exits the system without saving data.
Click the RETURN button below to continue
RETURN


INPUT PROCCESSING
COMPLETED
REMINDER: This is a prototype, data is not saved
Select QUIT to exit Hypercard or CONTINUE to
return to the beginning of this stack.
QUIT
CONTINUE

APPENDIX E. SAMPLE BENCHMARK TASKS

1. Create a new file with nothing important in it. Move the cursor to this file and press the mouse button. Drag the file into the context of another folder in the desktop. **While dragging the file**, type a short new file name on the keyboard. After typing the new file name string, when the cursor is within the icon of the folder to which you are moving, release the mouse button.

Now repeat this same task, however, instead of dragging the file to the folder icon, first open the folder, then drag the file into the folder window while typing the new file name.

2. Create a new folder called A and a new folder called B. Within folder A create several new files again with nothing important in them. Now copy these new files into a new folder within folder B.

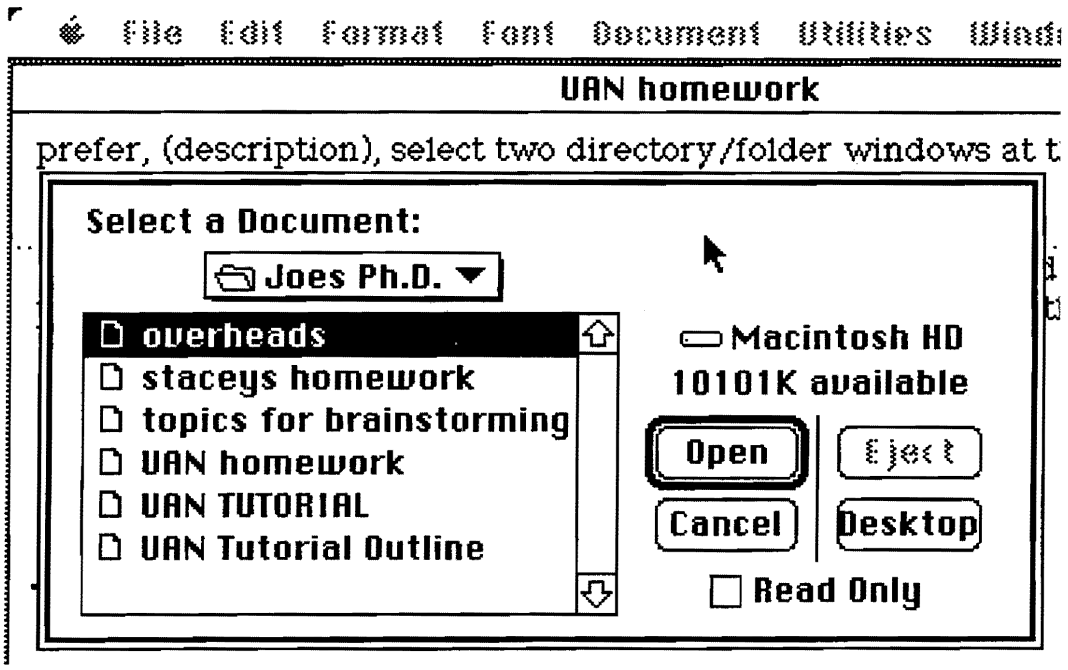
3. Print a directory. **Note** that a directory must be selected in order to perform this task, however, you may not assume that one is selected.

4. Using whatever method, or combination of methods that you prefer, select two directory/folder windows at the same time.

5. This question is about macros, levels of abstraction and passing parameters. Consider the task of opening a file, say, from within Microsoft Word. Let's say we want a task like this at a higher level:

TASK: Open-File-A
from (file) menu select (open)
use Open dialogue box
close Open dialogue box

where the Open dialogue box looks like this:



The other possibility is for the Open-File task to look like this:

TASK: Open-File-B
from (file) menu select (open)
use Open dialogue box

In this case, closing the Open dialogue box is part of the "use Open dialogue box" task. Which method of abstraction is better: Open-File-A or Open-File-B? Why?

Now suppose that we wanted to create a task description for closing a generic dialogue box with a scroll list, which can, in general be closed by:

- carriage return,
- double click on a list item,
- Apple command key combination,
- click on a labelled button.

For example, for the Open dialogue box above, the button that does this is labelled "Open" and the –O key combination works as well.

What parameters would you use for these and how? Why?

Vita

Born to Delmar and Janet Chase in Bowlingreen, KY, the author spent the first ten years of his life in several southern states. At the age of eleven, the author moved with his family to his father's hometown of Lewisburg, WV. Here he remained until graduating as valedictorian of Greenbrier East High School in 1982. In the Fall of 1982, the author entered the computer science program at Virginia Polytechnic Institute and State University.

After graduating Cum Laude from Virginia Tech with a B.S. in computer science, the author went to work for NCNB corporation in Charlotte, N.C. After moving from programmer to systems planning office over a period of two years, the author returned to Virginia Tech to continue his education, completing his M.S. in computer science in the Summer of 1990 and his Ph.D. in computer science in the Spring of 1994. A summary of the authors academic, professional, and research activities is given below. The author is currently teaching as a Visiting Assistant Professor at Radford University while searching for a permanent academic position.

EDUCATION:

Ph. D.	Department of Computer Science Virginia Tech (Blacksburg, Virginia) Dissertation: "A Study to Develop and Evaluate a Taxonomic Model of Behavioral Techniques for Representing User Interface Designs"	May, 1994
M. S.	Department of Computer Science Virginia Tech (Blacksburg, Virginia) Thesis: "A Controlled Experiment to Identify and Test A Representative Primitive Set of User Object-Oriented Cursor Actions"	August, 1990
B. S.	Department of Computer Science Virginia Tech (Blacksburg, Virginia)	June, 1986

ACADEMIC EXPERIENCE:

8/93 - present	Instructor, Radford University, Dept. of Computer Science.
6/93 - 8/93	Computer Science Instructor, VPISU, Dept. of Computer Science.
8/88 - 6/93	Teaching/Research Assistant, VPISU, Dept. of Computer Science.
6/90 - 8/90	Computer Science Instructor, VPISU, Dept. of Computer Science.

INDUSTRIAL EXPERIENCE:

8/93 - 8/93 Consultant, Naval Research Laboratory, Washington, D.C.
Investigating human factors and design representation issues surrounding modern interface techniques.

1/93 - 8/93 Consultant, Litton Poly-Scientific, Blacksburg, VA.
Investigating interface and performance issues for an object oriented security management system.

3/90 - 12/92 Systems/Programmer Analyst, Advanced Integrated Techn., Radford VA.
Responsible for the management of VAX/VMS system and associated networks. Also responsible for the design and development of a variety of government contract software.

1/89 - 6/90 Systems Manager, Human Computer Interaction Laboratory, VPISU.
Responsible for the maintenance and management of a variety of systems including a VAX 11/750, IBM PCs, and Apple Macintosh computers.

6/86 - 8/88 Programmer/Analyst, NCNB Corporation, Charlotte NC.
Responsible for the design and implementation of a variety of applications including a communications intensive credit bureau reporting system.

6/85 - 6/86 Programmer/Analyst, VPISU Department of Geological Sciences.
Responsible for the design and implementation of research applications.

6/84 - 6/85 Programmer/Analyst, VPISU Learning Resources Center.
Responsible for the design and implementation of office applications.

RECENT INVITED LECTURES

October	1993	"Human-Computer Interaction: Setting the Stage for the Next Century", Department of Computer Science, Concord College, Athens, WV.
July	1993	"The User Action Notation", NRL, Washington, D.C.
February	1993	"The Empirical Evolution of Behavioral Representation Techniques," Department of Information Systems, University of Maryland Baltimore County, Baltimore, MD.
October	1992	"Applying the User Action Notation in the Interaction Development Process", Department of Computer Science, Norfolk State University, Norfolk, VA.

RESEARCH PUBLICATIONS:

Sherry P. Casali and J. D. Chase, 1994, "The Effects of Physical Attributes of Computer Interface Design on Novice and Experienced Performance of users with Physical Disabilities", to appear in the International Journal of Industrial Ergonomics.

J.D. Chase, Robert S. Schulman, H. Rex Hartson, and Deborah Hix, "Development and Evaluation of a Model of Behavioral Representation Techniques", Proceedings of CHI '94 "Celebrating Interdependence", ACM Conference on Human Factors in Computing Systems, April 1994, Boston, MA.

J.D. Chase and Sherry P. Casali, "An Investigation of Factors Affecting the Size of Graphical User Interface Objects", Proceedings of The 2nd Annual Mid-Atlantic Human Factors Conference, February 1994, Reston, VA.

J.D. Chase and Deborah Hix, "The Analysis of an Existing User Interface Using the User Action Notation: A Case Study", Proceedings of The 2nd Annual Mid-Atlantic Human Factors Conference, Februrary 1994, Reston, VA.

Sherry P. Casali and J. D. Chase, 1993, "The Effects of Physical Attributes of Computer Interface Design on Novice and Experienced Performance of users with Physical Disabilities", Proceedings of The 37th Annual Meeting of The Human Factors and Ergonomics Society, Human Factors Society, 849-853.

J. D. Chase, H. Rex Hartson, and Deborah Hix, 1993, "A Model of Behavioral Techniques for Representing User Interface Designs", Proceedings of HCI International '93, Elsevier Science Publishers, 861-866.

J. D. Chase, Marie Peretti, H. Rex Hartson, and Deborah Hix, 1993, "Task-Oriented User Documentation Using the User Action Notation: A Case Study", Proceedings of HCI International '93, Elsevier Science Publishers, 421-426.

J. D. Chase and Sherry P. Casali, 1993, "The Effect of Direction on Object-Oriented Cursor Control Actions", Proceedings of HCI International '93, Elsevier Science Publishers, 231-236.

J. D. Chase, Sherry P. Casali, and H. Rex Hartson, 1992, "The Predictability of Cursor Control Device Performance Based on a Primitive Set of User Object-Oriented Cursor Actions", Proceedings of the Human Factors Society 36th Annual Meeting, Human Factors Society, 306-310.

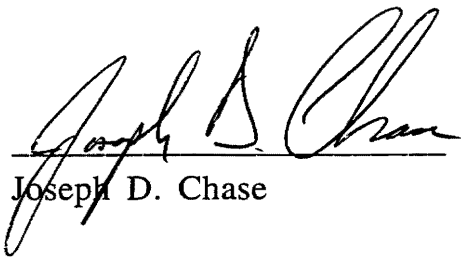
J. D. Chase, 1992, "The Empirical Exploration of Usability and Behavioral Representation Techniques", Poster session, Virginia Computer Users Conference, Blacksburg, VA.

J. D. Chase and Sherry P. Casali, 1991, "A Comparison of Three Cursor Control Devices on a Cursor Control Benchmark Task", TR 91-8, Department of Computer Science, VPISU.

J. D. Chase, 1990, "A Controlled Experiment to Identify and Test a Representative Primitive Set of User Object-Oriented Cursor Actions", Masters Thesis, Department of Computer Science, VPISU.

POINTS OF INTEREST:

* Eagle Scout, B.S.A.
* Upsilon Pi Epsilon, CS honor society
* Phi Kappa Phi, collegiate honor society
* High School Valedictorian
* Gamma Beta Phi, colleg. honor society
* Avid sportsman and naturalist



Joseph D. Chase