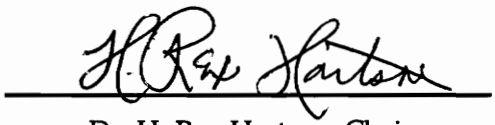
QUANTUM:
Quick User Action Notation Tool for User interface Management

by
Arcel Macaraeg Castillo

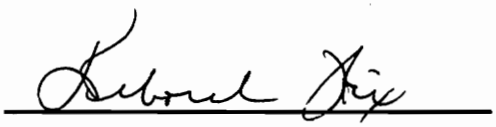
Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment for the requirements for the degree of

MASTER OF SCIENCE
in
Computer Science and Applications

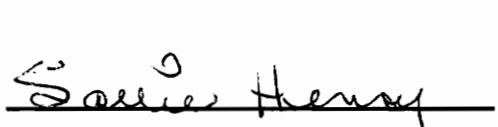
APPROVED:



Dr. H. Rex Hartson, Chair



Dr. Deborah Hix



Dr. Sallie Henry

September, 1993
Blacksburg, Virginia

Abstract

The UAN (User Action Notation) is a task- and user-oriented behavioral representation technique for specifying interface designs that is used to communicate interface designs, support task analysis, and facilitate usability evaluation. A UAN interface description is made up of several linked sheets of paper containing task descriptions, notes, and sketches. UAN users have reported that writing and reading these designs is difficult. It is our hypothesis that these problems in usability are not inherent in the notation itself, but rather that effective use of UAN is hampered by the manual manipulation necessitated by pencil and paper. QUANTUM (Quick User Action Notation Tool for User interface Management), a software tool to facilitate writing and reading of UAN for user interface design, was developed through an iterative process based on formative evaluation by expert UAN users. QUANTUM supports development of a graph-based representation of a task abstraction hierarchy for a user interface design, with each node representing a user task. Task description windows for each task node allow entry of UAN in a tabular, spreadsheet-like format. Task libraries enable designers to build archives of task descriptions, enabling their reuse in other interface design projects. Note addenda containing explanatory text, screen sketches, design questions, and/or audio/video can be placed anywhere within a design description. These features, not afforded by pencil and paper, give QUANTUM a functional advantage over such media. Expert evaluators used in qualitative formative evaluation of the QUANTUM prototype indicated a strong preference over pencil and paper for using QUANTUM to produce UAN task descriptions. As a result of this work, we claim that QUANTUM improves usability of the UAN.

Acknowledgments

I would like to thank my advisor and role model, Dr. H. Rex Hartson for his guidance and understanding throughout my graduate work. It is his devotion to his research and teaching, and the joy with which he executes each that inspired me to keep going despite discouragement. His patience, honesty, and friendship are invaluable. Likewise, Dr. Deborah Hix's precision in her work and her advisement kept my work focused, and I thank her for her flexibility. Dr. Sallie Henry has always provided a watchful eye over me since my undergraduate years, and I wouldn't have been here in graduate school without her. To my family, who waited patiently for me to finish my schooling so I could get a job like a real human being, my love and thanks. I hope I can count on you for support when I pursue my Ph.D. To Kevin Mayo, Joe Chase, Jeff Brandenburg, and all the other members of the HCI Research Group, thank you for the help and advice. To Craig Struble and Matt Jackson who laid down the foundation for the prototype, you know that it couldn't have happened without you. To my friends who saw me through my graduate years, Stacey Ashlund, Joe Lavinus, Ed Dorsey, Bob Mullins, Loretta Auvil, my friends in the GSA, and all the others who kept urging me on, I can finally say, "I made it!". Now they can all say, "It's about time!" To all my friends back home who wondered why I was still in school, well you're holding the reason in your hands.

Finally to Sue, your understanding and love got me through till the end. When I doubted my ability you were there to give me support (and sometimes a kick in the pants). I hope I can make you as happy as you make me.

Table of contents

Table of figures

1. Introduction

This thesis reports the design, prototype construction, and expert evaluation of QUANTUM (Quick User Action Notation Tool for User interface Management), a software tool to support reading and writing of UAN (User Action Notation) task descriptions of a user interface design. QUANTUM is intended for user interface designers, and facilitates reading and writing of detailed UAN descriptions of user interface designs, from high-level task abstraction descriptions to low-level task articulation descriptions.

1.1. Problem: Reading and writing UAN

The UAN (User Action Notation) is a user- and task-oriented user interface design representation technique that describes the behavior of the user and the computer during their cooperative performance of a task (Hartson, Siochi, & Hix, 1990) (Hix & Hartson, 1993). Usage of the notation is widespread and more than 50 real-world development sites have used the UAN. A user interface design is defined in terms of user tasks, with each task composed of a set of subtasks. This quasi-hierarchical task structure represents levels of task abstraction. At the highest abstraction level, user tasks represent broad goals and intentions. At intermediate abstraction levels, user tasks become more specific, and are thought of as 'macros' made up of still lower-level tasks. At the lowest abstraction level, user tasks represent primitive user actions such as mouse movement and keyboard usage. Symbology of the UAN is a visually onomatopoeic mixture of keywords and symbols that represent logical and temporal relationships between task actions and interface feedback. The UAN is used by an interface designer as a structured yet flexible method for precisely representing the behavior of an interface design.

Many students in human-computer interaction classes at Virginia Tech, as well as clients in industry, use the UAN, and find its usage difficult. This may result in reducing the effective use of the notation, which can lead to poor quality design representations. The non-linear link nature of an interface design presents a difficult navigation problem when represented in the sequential format of manually manipulated physical media such as pencil and paper. Description of a user interface design using UAN involves management of tens or hundreds of task descriptions. Each task description, recorded in a template on a sheet of paper, is linked to other subtask descriptions, as well as figures and other annotations. Manual manipulation of

these interface descriptions makes storage, access, and usage of such designs awkward. It is our hypothesis that these problems are not due to usability problems inherent in the UAN itself, but rather that usage of the notation by manual manipulation makes effective and efficient use of the UAN difficult. This is primarily due to the linear representation of an interface design using pencil and paper. Usage of word processors and other individual software tools for interface design development with UAN alleviates some of the problems of entering and formatting UAN, but does not address the problems of navigation of interface designs and integration of interface design components.

1.2. Solution: QUANTUM

The goal of this research is to alleviate these problems by the design of an integrated, dedicated software support environment. Such a design for a software tool for developing and managing UAN interface descriptions was developed and is called QUANTUM for Quick User Action Notation Tool for User interface Management. It is a primary hypothesis of this research that QUANTUM offers demonstrable advantages over conventional pencil and paper in terms of capability and efficiency. These advantages translate into increased usability of the UAN and increased utility of the UAN as a design tool.

QUANTUM supports the development of a graph-based representation of a user task abstraction hierarchy. Within this graph-based representation, nodes represent user task descriptions at various levels of abstraction, and edges represent hierarchical dependencies of those tasks. Each node is hyperlinked to a window containing a detailed UAN description for that task, which is presented in a tabular spreadsheet format. Direct manipulation of the task abstraction hierarchy enables easy navigation among tasks in an interface design.

Five types of annotation objects, referred to as notes, can be directly hyperlinked at any point in an interface design. These notes increase the level of detail a designer can specify in an interface design. In addition, these notes can be linked among themselves, creating annotations of multiple information dimension. Text notes allow addition of supplementary text for further explanation. Sketch notes enable representation of task arenas, state diagrams, or any other suitable pictorial design component. Issue notes allow highlighting of design issues by way of problem and solution discussion. Audio and video notes enable inclusion of multi-media information in an interface design - this may include voice or video comments or recording of user task performances.

Task libraries allow designers to build meaningful collections of task descriptions for later reuse. These libraries allow development of platform- or metaphor-specific archives, useful for later design projects. An on-line UAN reference sheet gives QUANTUM users quick access to UAN symbology, and provides usage examples. Extensive on-line help instructs the designer on QUANTUM usage, and gives instruction on the UAN. QUANTUM supports both bottom-up design and top-down design by way of simultaneous access to both the UAN description and the hierarchical status for any task. These features are not afforded by pencil and paper, giving QUANTUM an advantage over such media. The interactive nature of QUANTUM should enable efficient storage, access, management, and distribution of interface designs.

1.3. Approach

Design for QUANTUM was derived from extensive interviews and discussion with UAN users ranging from novice to expert. Initial formative evaluation of the QUANTUM design was performed several times by various groups of UAN and interface design experts, through storyboard walkthroughs and later through prototype interactions. A QUANTUM prototype was implemented using Tcl/Tk, a public-domain windowing language, on a DECstation 5000 Model 100 running ULTRIX 4.3 and X11R5. The results of several evaluations were used to redesign the QUANTUM user interface. After construction of a fully testable prototype was completed, a qualitative formative evaluation of the design's usability was performed by three expert UAN users. These users were chosen based on their expertise with both UAN and user interface design. The evaluation tasks consisted of writing and reading tasks using the prototype, resulting in formulation of a list of design recommendations to be implemented in further development of QUANTUM. Evaluators were also asked to subjectively evaluate use of the tool as compared to pencil and paper in developing UAN task descriptions. Results from these evaluations indicate a strong preference over pencil and paper for using QUANTUM to produce UAN task descriptions. The support environment of QUANTUM provides a higher degree of usability for the UAN, thus freeing users from many physical mechanics of working with the UAN.

1.4. Summary

The motivation of this research was to demonstrate the need for interactive support for the UAN. The goal was producing a design for such an interactive tool. The hypothesis of this research is that such a software tool makes the UAN easier to use, thereby increasing the utility

of the UAN as a design technique. Results from qualitative formative expert evaluation of a QUANTUM prototype indicate a strong preference for QUANTUM over pencil and paper. Future research and development of QUANTUM will include further formative subjective and objective evaluation. Redesign of the QUANTUM user interface will be based on these evaluations. Future work will also include development of proposed extensions to the tool.

2. Context

This chapter sets the context for importance of the development of a UAN tool. An overview of user interface development is presented, followed by a domain model for the interface design process, tools and techniques associated with each domain, and motivation for a behavioral representation technique such as the UAN.

2.1. User interface development

It is widely recognized that the user interface plays an increasingly prominent role in contemporary interactive applications. This has lead to the birth of an entire research community devoted to the study of human-computer interaction. For end users, *usability* of an application has become as, if not more, important than its capability. The amount of application development time and effort spent on the user interface has also increased. It is estimated that nearly 50 percent of the code and development time for modern interactive systems is devoted to the user interface (Myers & Rosson, 1992). This illustrates the importance of developing usable user interfaces in a cost-effective and efficient manner.

In many situations, the user interface is designed by application programmers, who view the design from a system's viewpoint, in terms of input/output, code, and data structures. This approach usually leads to the creation of user interfaces that are structured around the way the system works and not around the way the user works. For a usable design, an early focus must be taken on users and their tasks (Gould & Lewis, 1985) (Gould, Boies, & Lewis, 1991). User interfaces should be designed from the user's point of view and take into account the user's goals and intents. *User-centered design* is based on the principle that interface designs should be built from the user's viewpoint to ensure usability (Norman & Draper, 1986). Creation of a user interface can be thought of as involving two broad processes; development of the interaction component, and development of the interface software (Hix & Hartson, 1993). This dichotomy is shown in the following figure.

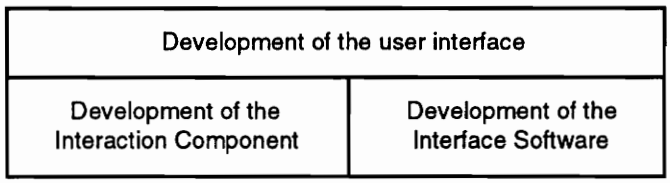


Figure 2.1 User Interface Development from (Hix & Hartson, 1993)

The interaction component is the user interface design - this is the model of user-computer interaction dialogue that defines the user interface. The development process of this component is carried out from the user viewpoint. The interface software is the user interface implementation - this is the actual application code that runs the user interface on the computer. The development process of this component is carried out from the system viewpoint. It is generally accepted that a separation between the interaction dialogue and the computational components results in more maintainable and modifiable systems (Hartson, 1989). These two processes thus exist in different domains of activity, outlining the distinction between the behavioral domain and constructional domain of user interface development.

2.1.1. Behavioral vs. Constructional

The *behavioral domain* is concerned with development an interaction design. The *constructional domain* is concerned with the software implementation of an interaction design. The following chart illustrates the difference between the process domains.

	Behavioral Domain	**Constructional Domain**
View	User View	System View
Describes	Describes user actions	Describes system actions
Design	Interaction design	Interface software design
Involves	Human factors,user interviews, task analysis, design representation, prototype testing and evaluation	Interface toolkits and standards, algorithms, low-level programming and debugging

Figure 2.2 Behavioral vs. Constructional Domains adapted from (Hix & Hartson, 1993)

Much human-computer interaction research is concerned with the constructional domain. Many interface builders and toolkits exist to support activities in this domain. Ranging from programming languages for windowing systems to design-by-example systems, these environments support the implementation of a user interface design into run-time system code (Myers, 1989). In contrast, the behavioral domain encompasses activities that lead up to and include development of a user interaction design. This takes into account factors such as users' background knowledge and capability (user modeling), analysis of the tasks being performed as well as their environment (task analysis), and creation of benchmark tasks and usability specifications (usability evaluation).

2.1.2. Models of user interface development

Design of a usable user interface follows a quite different process model than those traditionally used in the study of software engineering. The *waterfall model* (Boehm, 1988) (pictured below) has long been used as an example of top-down, structured development of a software application. The waterfall model is based on a sequential progression through stages of requirements analysis, through design, and on through testing and maintenance. However, this model fails to capture the notion of feedback from various stages into previous stages as a development component.

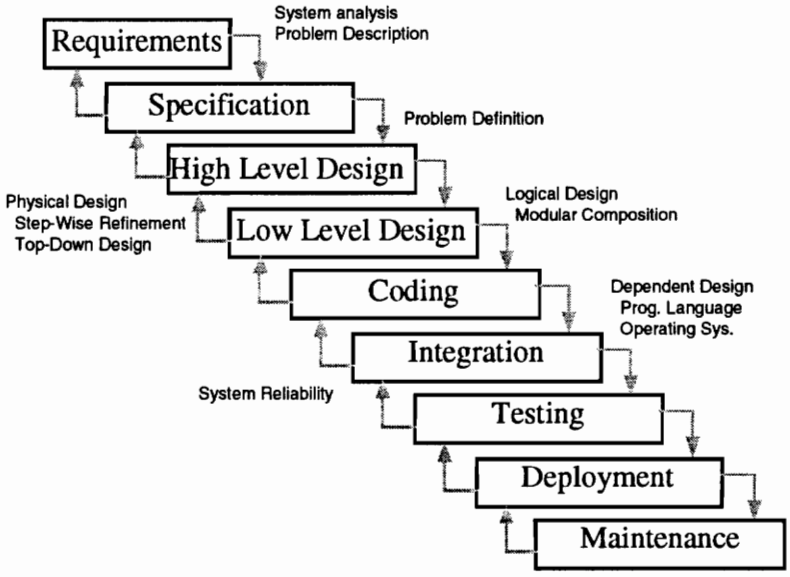


Figure 2.3 Waterfall Lifecycle Adapted from (Boehm, 1988)

The *spiral model* (Boehm, 1988) of software engineering is an adaptation of the waterfall model that includes iteration as part of the life cycle. While the model is essentially sequential, testing and evaluation are included at various stages in the development process, the feedback of which is incorporated into one or more development stages.

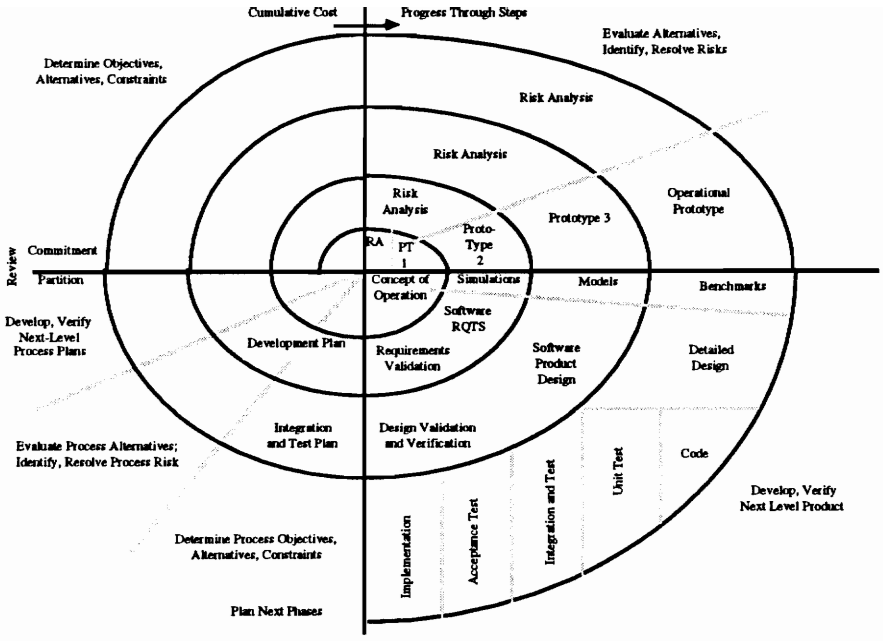


Figure 2.4 Spiral model of software development (adapted from (Boehm, 1988))

However, this model does not accurately model the development of an interface design, mainly due to the dynamics of the interaction situation. Due to the unpredictability of human behavior (Gould & Lewis, 1985), many aspects of the user-computer interaction are unknown until later evaluation. These aspects, based on the context of use, can only be determined through observation. This illustrates the need for a process model that is *iterative by nature*.

The **star life cycle** (Hartson & Hix, 1989) (Hix & Hartson, 1993) is a model for user interaction development that places evaluation as the center of all development activity. This model is pictured below.

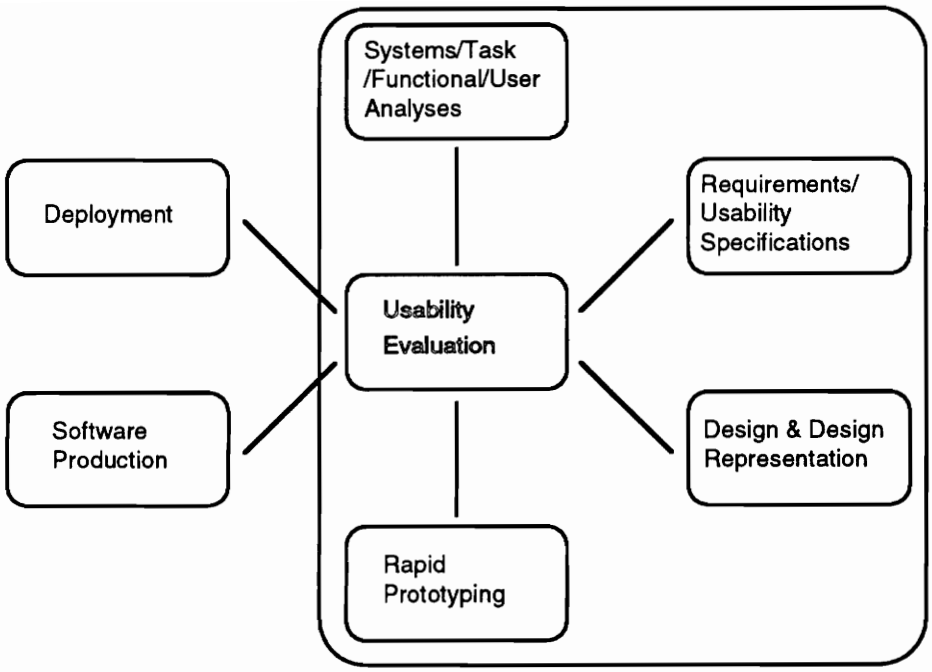


Figure 2.5 Star life cycle (from (Hix & Hartson, 1993))

This model mandates no explicit ordering of activity stages, rather each activity is interdependent with the others, using evaluation results of one activity driving another. This model more accurately describes the real-life processes of iterative redesign, evaluation, analysis, and implementation that occur during application development. This interdependence of activities must be supported by tools and techniques for evaluation and communication (Hix & Hartson, 1993).

2.1.3. Tools to ensure and evaluate usability

MUSiC (Metrics for Usability Standards in Computing) (Macleod & Bevan, 1993) is a set of tools to support usability measurement and evaluation. The MUSiC Performance Measurement Method (PMM) provides a validated method for usability evaluation through the application of performance-based usability metrics to video recordings of users. This analysis is carried out by DRUM (Diagnostic Recorder for Usability Measurement), a software tool that automates processing of video recordings of usability evaluation sessions. The MUSiC Context Guidelines Handbook enables evaluators to identify and describe characteristics of the 'context of use,' defined as the users, tasks, and environments for which a system is designed. Several other tool environments exist to support work performed in usability labs (Weiler, Cordes,

Hammontree, Hoiem, & Thompson, 1993). These tools are specifically designed to improve the data collection and analysis process for usability labs. All these tools focus on the evaluation of user interfaces through usability testing.

However there are few tools that support other processes that lead up to the formation of a user interface design, activities such as task analysis, user modeling, and interaction description (representation of the user-computer interaction). **ADEPT** (Advanced Design Environment for Prototyping with Task Models) (Johnson, Wilson, Markopoulos, & Pycock, 1993) is a novel design environment based on construction of a graphical representation of a user task structure. ADEPT incorporates task and user modeling components with a rapid-prototyping tool, offering the designer a user and task-centered interface design environment. This environment, while offering extensive support for the design processes of user interface development, offers no connections to the evaluation phases of user interface development.

IDEAL (Interface Development Environment and Analysis Lattice) (Ashlund, 1991)is a framework of logically linked tools that attempts to integrate support for all phases of user-interface development. The major focus of the IDEAL environment is support for empirically-based formative evaluation of user interfaces through support of such activities as benchmark task creation, and video recording of usability evaluations. IDEAL also supports other activities throughout the interface design process such as user classing and design representation. Each activity is supported by a specialized tool. Data from each tool are logically linked to data from other tools in the lattice, directly supporting the interdependent nature of the star life cycle. With all tools working in concert, the interface designer is provided with a powerful environment for managing activities in the behavioral domain of the user interface development process. The following figure outlines the relationship of IDEAL and other behavioral tools and techniques to those of the constructional domain.

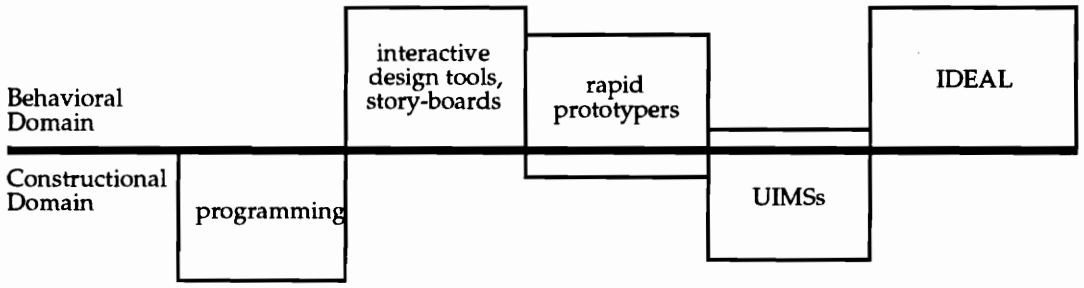


Figure 2.6 IDEAL relative to other tools, adapted from (Ashlund, 1991)

The IDEAL environment is a topic of ongoing research, and tool prototypes are currently being designed and developed. One of the key differences between the IDEAL environment and other tools/environments is its foundation on a technique for the structured representation of the interaction between the user and computer. This representation technique, the UAN, is capable of high detail, yet flexible enough to make its usage practical.

The focus of this thesis research is the development of QUANTUM, a tool to support interface design representation using UAN. This tool will be described in detail in a later chapter. QUANTUM is designed to operate as both a standalone tool, and as a component tool within the IDEAL (Ashlund, 1991) framework.

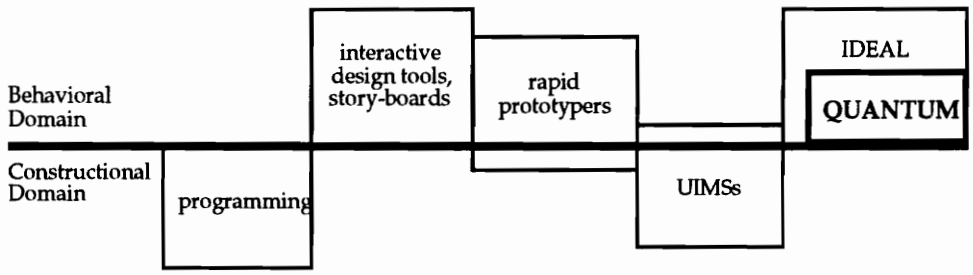


Figure 2.7 QUANTUM as related to IDEAL, adapted from (Ashlund, 1991)

The next section explores the motivation for the creation of a behavioral representation technique such as the UAN.

2.2. Motivation for a behavioral representation technique

Key to interface design is having a means of accurate and precise design representation technique. For representing interface designs, there is a need for a vehicle that is expressive, precise, and extendible.

2.2.1. Existing representation techniques

One way to represent the behavior of a user interface design is with formal grammars. The **SPI** (Specifying and Prototyping Interaction) (Alexander, 1987) methodology treats the dialogue of human-computer interaction as a set of discrete events acting on a state, the structure of which is specified in a subset of CSP (Hoare, 1985). The SPI method uses a functional specification language called **me too** to view an interactive system description as a function from its input stream to its output stream. Formal grammars such as action grammars (Reisner, 1981)

have also been shown to have application in the analysis of interface designs, and in the detailed representation of user interfaces, as in CLG (Moran, 1981). However, while adequate for small-scale systems, specifications written in formal grammars for realistic systems become too complex for effective usage in the development of a user interface design.

Another way to represent an interface design is through the use of storyboarding. Similar to the use of storyboards in filmmaking, this involves the use of a combination of written descriptions and screen sketches to describe interaction scenarios. These storyboards fail to offer enough information about temporal and logical relationships between tasks, and do not capture enough of the necessary detail to construct the interface, possibly leaving a great deal of decision-making up to the interface implementer.

The behavioral nature of task analysis methods makes them ideal candidates for design representation, but they lack both the power and the flexibility needed for full description of a user-computer interaction. These schemes, including TAG (Payne, 1986), the work of Kieras and Polson (Kieras, 1985), the keystroke-model (Foley, 1980) are only intended for analysis of existing designs, and have little applicability for use in generation of interface designs. While the GOMS (Card, Moran, & Newell, 1983) model has been successfully used in describing user interfaces, the subjective nature of the analysis and the lack of temporal relationships among tasks make it incomplete for full description of an interface.

2.2.2. Problem: languages do not fully capture cooperative interaction

The nature of an interface design is the detailed description of the interaction behavior between the user and computer in their cooperative performance of tasks. This does not simply take into account the layout of objects on the screen, but also task performance issues such as conditions of viability, interleaving, abandonment, and logical connections between tasks. Any vehicle that attempts to capture this detail must be powerful enough to address these issues in a manner that is easy to read, easy to use, and easy to extend. This calls for a notation that is behavioral by nature.

2.3. The User Action Notation (UAN)

The User Action Notation (UAN) (Hartson, et al., 1990) was developed at Virginia Polytechnic Institute and State University, growing out of a need by a research group working on

the interface design for Dialogue Management System. It is a task- and user-oriented behavioral representational technique for describing asynchronous, direct-manipulation interface designs in terms of user tasks at different levels of abstraction. By specifying both the user actions and interface feedback associated with a user's task, UAN captures the **cooperative behavior** of a user and computer in their performance of that task. This allows designers to be very specific in describing the interaction behavior.

This notation is seeing widespread use in academia and industry as both a design aid and a testing aid. It can aid in communicating a design to an interface implementer, aid in performing task analysis by providing a notational technique, and aid in usability testing by providing a vehicle for performance analysis. Since its creation, UAN has been used in over 50 academic and commercial settings.

2.3.1. Examples

The following subsection presents a brief overview of the UAN. For further detail, the reader is referred to (Hartson, et al., 1990), (Hartson, 1991), (Hartson, Brandenburg, & Hix, 1992), and (Hix & Hartson, 1993).

Abstraction

The central concept of an interface design described in UAN is the user task. By way of abstraction, the UAN allows the designer to break down high-level tasks and define them in terms of lower-level tasks. This **hierarchical decomposition of tasks** lets the interface designer build the interface around the task structure of the end-user. This also increases the power and detail of the tasks being described while making it easier to write. This is similar to the way that programmers will break high-level procedures and functions down into lower-level units.

As an example of task structuring, consider the task of selecting and opening a file from within Microsoft Word. The user has many different task paths available, all leading to the same final state of an opened file. The user may first elect to choose the Open... command from the File menu via a mouse click-drag-release sequence, or may use the keyboard shortcut of Command-O, both bringing up the Open dialog box (shown below).

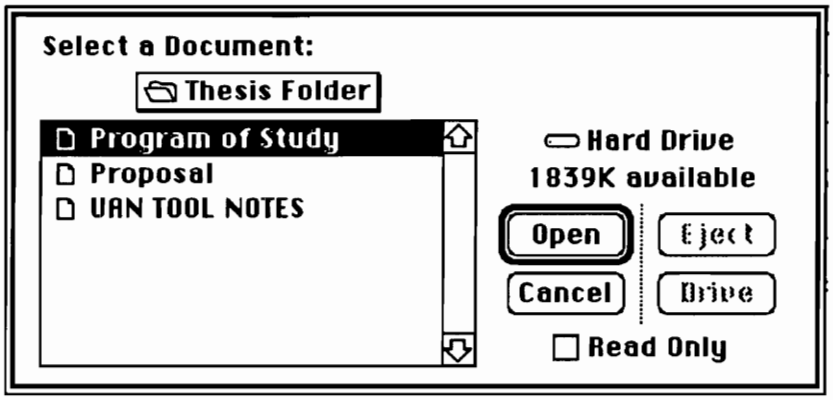


Figure 2.8 Sample Macintosh Open Dialog box

The user must then choose the file to be opened by selecting from a list of candidates in a list box. Manipulation of the current selection can be accomplished by either using the keyboard (pressing arrow keys) or the mouse (clicking directly on the target filename) or a combination of both (moving through folders or switching drives). Representing this behavior involves addressing temporal notions of interleaveability and dependency, notions not easily presented without the use of abstraction. One way to look at the structure of this task (at a very high level) might be:

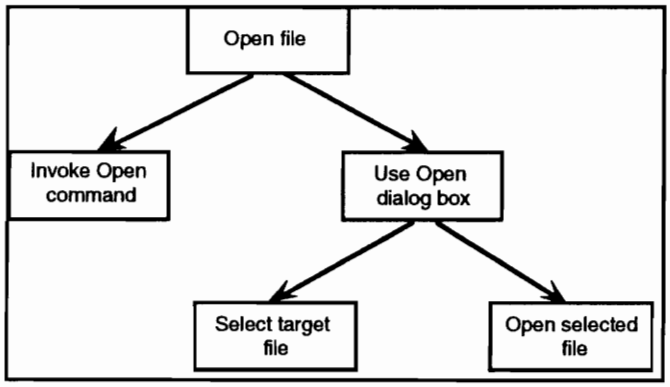


Figure 2.9 Abstraction Structure for Open Dialog box

This quasi-hierarchical task structure represents levels of task abstraction. From top to bottom of this structure, the level of abstraction decreases as high-level tasks are decomposed into lower-level tasks. At the highest abstraction level, user tasks represent broad goals and intentions At intermediate abstraction levels, user tasks become more specific, and are thought of as 'macros' made up of still lower-level tasks. At the lowest abstraction level are primitive user actions such as mouse movement and keyboard usage. The use of abstraction in this manner

results in a logically structured interface design description, oriented in terms of user tasks rather than system-oriented.

Notation

The symbology of the UAN is a visually onomatopoeic mixture of keywords and symbols that allow representation of logical and temporal relationships between task actions and interface feedback, and of any other interface design component. These symbols were chosen for easy readability and terseness of expression. For example, a mouse is represented by the letter M. Movement is represented by a tilde mark (~). The use of this shorthand greatly reduces the length of interaction descriptions.

An 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" using the keyboard.

In UAN, `K` represents the keyboard, and the string `"Hello World"` is treated as an action to be performed using the keyboard. Thus, the UAN notation for this task is:

```
User Action
K"Hello World"
```

The UAN representation of the this task is much more concise than its English counterpart.

As a more complete example, the following is the UAN description for another task, that of moving a file icon within the Macintosh Desktop, a direct-manipulation interface that uses icons to represent objects such as files. This 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 for this task is shown below:

Task: move a file icon			
ARENA	**USER ACTIONS**	**INTERFACE FEEDBACK**	**INTERFACE STATE**
desktop	~[file_icon] Mv	file_icon !	file is selected
	~[x,y]* ~[x',y']	outline(file_icon) > ~	
	M^	display(file_icon) @x',y'	file is placed at new location

Figure 2.10 UAN task description for moving a file icon

A UAN task description represented in a table, broken down into steps of user action, each represented on a line made up of four columns. The 'Arena' column specifies the context within which the task takes place, in this case the `desktop`, a screen sketch of the Macintosh desktop. The 'User Actions' column contains the notation that represents the physical user behavior. The first line of the User Actions specifies that the cursor is to be moved into the context of the file icon on the screen, at which time the mouse button is to be depressed. The 'Interface Feedback' column describes the feedback presented by the interface in response to the associated user action. The first line of this column indicates that in response to the mouse button press, the file icon will highlight. The 'Interface State' column indicates any state that the interface may enter upon performance of the described user action. The first line of this column indicates that as a result of moving the cursor to a file icon and pressing the mouse button, that file is in the state of being selected. The second line of User Action, `~[x,y]* ~[x',y']` indicates movement of the cursor to an arbitrary point `x,y` on the screen in a succession of zero or more arbitrary points about the screen, ending at the point `x',y'`. The Interface Feedback column indicates that the outline of the file icon follows the movement of the cursor. Finally, in the third line the mouse button is released, at which time the file icon is redisplayed at the point `x',y'`, indicating that the file is now at a new location. Additional columns may be added to a task description, including a general notes column, in which references to other figures, and annotations may be placed.

2.3.2 Purpose

The original mission of the UAN was to serve as a vehicle for **communicating an interface design,** usually from interface designer to interface implementer. Communicating the behavioral aspects of an interface design using English prose proves imprecise due to the potential for ambiguity. The brevity and readability of the UAN can make communication of an interface design precise and detailed. Without a notation like the UAN, interface design descriptions rely on written descriptions, rough screen sketches, and sometimes verbal

communication. This method of information transfer fails to capture the detail inherent in an interface design, and leaves a considerable amount of interpretation and decision-making up to the interface implementer.

Consider a situation where the design for a control panel for a communications program is being formulated. This control panel sets the speed and parity of the communications line being used. Below is the design as envisioned by the interface designer.

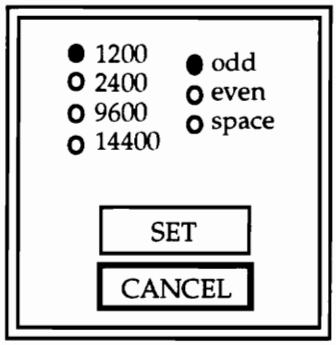


Figure 2.11 Sample control panel as envisioned by interface designer

Further suppose that the accompanying English description for the above design consisted of the following:

> There are two banks of buttons, each bank sets the attribute for line speed and line parity. The left bank lets the user choose among 1200, 2400, 9600, and 14400 baud. The right bank lets the user choose between odd, even, or space parity. The user can choose only one item in each bank. To click on a button, the user moves the pointer over the desired button and presses the mouse button. There is a Cancel button on the bottom that cancels any choices made, an a Set button that enables any choices made.

The description above seems to fit the given design very well. Radiobuttons are a common interface widget type that allows unique selection of one out of a choice set. However, note that the behavior of the bank buttons when clicked was completely unspecified, nor was any mention made of the behavior of the Cancel or Set buttons. The implementer is thus left to make decisions about portions of the design that are unspecified. Below is a possible interpretation of the textual description given above.

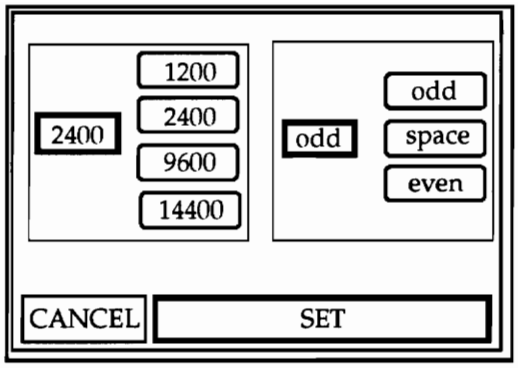


Figure 2.12 Sample control panel as implemented by interface implementer

In this design, the push buttons change the current setting for speed or parity, but the button clicked does not stay depressed. Rather, the new value for speed or parity is displayed in a window to the left of the bank of buttons. This is significantly different from the intended design. Radiobuttons may have been intended by the designer to maintain consistency between similar screens. Even if radiobuttons had been specified in the textual description, the exact behavior (highlighting, effect on other buttons, etc.) of the radiobuttons is left unspecified.

Without explicit description of the user actions and interface feedback, a great deal of interpretation and decision-making is left up to the inter face implementer. The UAN allows the interface designer to be very specific in the behavior of the system by precisely describing user actions with interface feedback as well as changes in interface state in response to those user actions. A task arena allows the interface designer to specify the context within which the task is to be performed, either pictorially or textually. Other categories may be added as desired, depending upon the information that the designer wishes to capture. The symbology of the UAN is completely customizable and extendible, making it adaptable to a number of interaction situations.

The UAN can also be used in **interface evaluation,** by allowing interface evaluators to concisely describe user actions for later analysis. Used in this manner, the UAN offers a detailed view of the users' task execution - down to the primitive user action level - that can aid in uncovering problems in an interaction design such as inconsistencies.

These examples illustrate that the User Action Notation offers a concise and extendible method for describing user tasks. By way of its abstraction mechanisms and high detail, interface designs can be precisely described. However, observations of and interviews with UAN users

have revealed problems in the usability of the notation with regard to its use on pencil and paper. The next chapter outlines these problems and motivates the need for a dedicated software support tool (QUANTUM).

3. The case for a tool

In this section, problems associated with usage of the UAN are presented. The hypothesis is made that development of a software support tool for the UAN will alleviate these problems.

3.1. Problems in using UAN

The UAN was intended from the start to provide user interface designers with a concise and precise notation for describing designs. The UAN offers great flexibility to the interface designer by way of its abstraction mechanisms, temporal notations, and extendibility. However, many of the comments by UAN users center around difficulties in using the UAN. These comments are not directed at the usability of the UAN as a notation or a technique, but rather are directed at the application and usage of the UAN. Many of the comments involve issues of navigation, management, information depth, and integration into other development activities.

3.1.1. Difficult navigation of interface designs

An interface design described in UAN can be made up of tens of hundreds of sheets of paper, each containing a task description. Each task descriptions is linked to descendent task descriptions by way of the task abstraction structure. These links can be thought of as "calls" to subtasks. Further, these descriptions can include links to figures, state transition diagrams, additional explanatory text, and references to other annotations. This task abstraction structure is similar to a k-ary tree where each node represents a task description (with possibility of non-unique termination nodes). An example structure is pictured below.

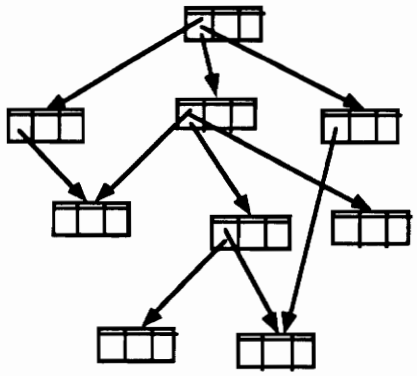


Figure 3.1 Hyperlink task structure of a UAN interface design

This non-linear hyperlink nature of an UAN interface design presents a difficult navigation problem when represented in the sequential format of physical media. Representing the hyperlink structure of an interface design is similar to the representation of a k-ary tree in an array . Each array cell represents a node in the tree. Each array cell holds the data for a node and contains a list of indexes to other array cells representing child nodes. In the case of a UAN interface design, the array is mapped to a labeled sequence of description sheets. This linear representation is pictured below.

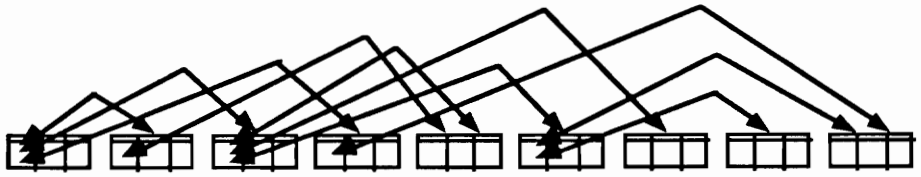


Figure 3.2 Linear representation of hyperlink task structure

Reading of such linearly represented designs is awkward and time-consuming, as navigation of the abstraction structure proves to be tedious. In group walkthrough sessions of interface designs described in UAN, participants were asked to trace through a user task path. Participants flipping through sheets of paper rapidly became confused and lost. A major part of the problems UAN users experience is difficulty in backtracking to "calling" task descriptions from "called" task descriptions. While setting markers in the task path to serve as anchors aided in backtracking, the number of markers needed relatively simple tasks quickly became unmanageable. These participants and other UAN users have remarked on the need for better navigational support for reading UAN interface designs.

3.1.2. Difficult creation and management of interface designs

The description of an interface design in UAN using pencil and paper can become a formidable task. A UAN description of an interface design may be made up of tens or hundreds of individual task descriptions, each requiring the use of paper sheets, each possibly requiring associated arena diagrams, state transition diagrams, or other notations. Some UAN users have reported the use of word processors, drawing programs, and similar software tools to aid in the production and storage of UAN interface designs. However, the use of these independent tools does not easily address the hyperlinked, associative nature of an interface design as they do not readily enable the creation of such links. In addition, these tools must be customized to handle the use of UAN. Templates for UAN task descriptions can be created and stored with word

processors, but such templates are difficult to extend and update, and still do not address the hyperlink nature among task descriptions.

Consistency and reuse of task descriptions among different interface design projects is also a problem in the paper and pencil use of the UAN. Relatively low-level tasks such as menu selection and icon movement generally do not differ significantly between related interface designs. If a set of interface design style guidelines is being enforced, such as Motif (Open Software Foundation, 1989) or Macintosh (Apple Computer Inc., 1987) guidelines, then these tasks are standardized and do not differ. In pencil and paper usage, reuse of these task descriptions across interface design projects can be accomplished by photocopying and insertion, but this solution does not address the possibility of changes being made to these low-level task descriptions (possibly as the result of a style guideline change). Such changes mean manually updating all interface design projects that use those tasks.

3.2. Hypothesis: manual manipulation hampers usability

The driving hypothesis behind this research is that these apparent problems in usability of the UAN are not inherent in the notation. It is not the hypothesis of this research that the UAN has problems in its overall usability. Rather it is the usage of the UAN through manual manipulation of physical media that limits its usability. As with any product, utility is limited and dictated by usability. The reading and writing of interface designs using the UAN is made difficult by pencil and paper. These difficulties hamper the usability of the UAN, leading to dissatisfaction by its users. The use of word processors, drawing programs, and other software tools addresses some of the problems of storage and retrieval, but do not address issues of navigation.

3.3. Solution: interactive support will improve UAN usability

There is a need for a dedicated, integrated interactive support tool that enables easy creation, management, and navigation of UAN interface descriptions. The driving hypothesis behind this research is that such a software tool environment will solve the problems associated with the usage of the UAN, and improve the overall usability of the UAN. Since its creation, UAN users have remarked on the need for such a tool.

An interactive tool for the UAN could aid the overall interface design and implementation process by enabling interface designers to quickly produce interface designs in UAN for transmission to interface implementers. By allowing the designer to describe the behavioral aspects of an interface more quickly and efficiently, the software tool could facilitate interface design by allowing the designer to concentrate on behavioral issues rather than issues in the management of pencil and paper designs. This tool could also provide the interface implementer an interactive tool for examining a proposed interface design on-line. The hyperlink structure of a UAN interface design can be easily represented within software support, and graphic and textual annotations can be easily appended to such designs. The interactive, on-line nature of such a tool can also facilitate productivity with the UAN, in much the same way that the software word processors facilitate productivity in word processing tasks.

The advantages of such a tool are summarized in the following list:

- facilitated navigation of interface designs by readers (interface designers or interface implementers)
- facilitated creation and management of interface designs using the UAN, supporting the hyperlink nature of such designs.
- facilitated productivity due to increased usability of the UAN

The next chapter outlines the design of QUANTUM, a proposed tool environment for the UAN.

4. Proposal: QUANTUM

The proposed design of a software support tool for the UAN was developed called **QUANTUM**, an acronym that stands for **Q**uick **U**ser **A**ction **N**otation **T**ool for **U**ser interface **M**anagement. QUANTUM is designed to offer demonstrable capability and efficiency advantages over conventional pencil-and-paper for using UAN. QUANTUM offers interface designers an integrated on-line environment designed to support the creation of UAN task descriptions, management and navigation of the task abstraction structure of an interface design, easy attachment of annotations at any point in the design, construction of collections of task descriptions for later reuse, and easy storage and access of interface designs. The following diagram illustrates the basic component structure of QUANTUM.

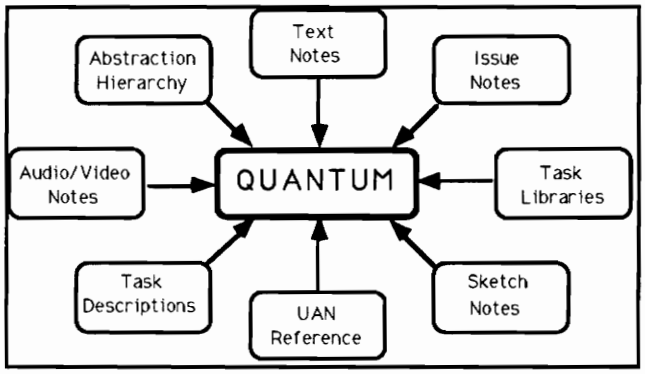


Figure 4.1 QUANTUM component structure

QUANTUM combines the following features into an integrated direct-manipulation, windowed environment to facilitate UAN usage:

- a manipulable graph-based view of the task abstraction structure of an interface design
- individual task description windows allowing easy entry and formatting of UAN
- text annotations, hyperlink-attachable anywhere in an interface design
- graphical annotations, hyperlink-attachable anywhere in an interface design
- ability to highlight and discussion of problems/issues in an interface design
- ability to attach audio/video annotations anywhere in an interface design
- archives of logically-grouped task descriptions for later reuse
- on-line reference sheet for symbology of UAN

A detailed overview of these features will be presented in a later chapter.

QUANTUM is intended primarily for use by user interface designers who wish to use the UAN for describing a design. Other QUANTUM users include interface implementers who wish to read an interface design on-line for the purposes of prototype construction, or usability specialists who wish to evaluate an interface design.

4.1. Advantages of QUANTUM

We claim that *QUANTUM is superior to pencil and paper, and to the use of non-integrated interactive tools* (word processors) for using the UAN. The following features are not easily afforded by pencil and paper media, giving QUANTUM an advantage over such media.

4.1.1. Affords easy navigation of interaction designs

The immediate access to a task description by way of the graph-based representation (called the Abstraction Hierarchy) affords *easy navigation of the task abstraction structure and associated annotations*. A walkthrough of a UAN interface design involves tracing a user task path from upper to lower levels in the task abstraction structure. This navigation becomes difficult when the structure is linearly represented. In QUANTUM, by simply double-clicking on a task's node in the Abstraction Hierarchy, the designer is shown the task description window for that task. Tracing a task path becomes a matter of tree traversal, a task that is naturally supported by the graph-based view. This natural mapping saves time and confusion on the part of design walkthrough participants.

4.1.2. Facilitates creation and manipulation of interaction designs

Interviews with UAN users indicated the use of two styles of interface design development: top-down and bottom-up. Designers who prefer top-down development tend to construct the task abstraction structure before filling out the UAN for the tasks. Designers preferring bottom-up development tend to write the UAN for lower-level task and proceed to build the task abstraction structure as each task description is completed. QUANTUM affords *interleaving of both top-down and bottom-up development* by allowing the interface designer to easily switch between editing of a task description to manipulation of the task abstraction structure.

The *hyperlink nature of an interface design is directly supported* in the graph-based representation of the task structure. Each node, representing a task, is hyperlinked to a window containing the UAN task description for that task. This hyperlink support allows the interface designer to easily view and edit the structure, without having to contend with a less-natural linear representation. In addition, annotation windows (text, graphic, issue, or audio/video) can be attached via hyperlink to any point in the interface design. Hyperlink representation is not easily afforded in pencil and paper or a word processor due to the inherent linear nature.

The on-line nature of QUANTUM promotes faster development of task descriptions as the *designer is relieved from manually management of pencil and paper.* QUANTUM provides the designer with ready-to-use windows for task descriptions containing a UAN table template in a tabular spreadsheet-like format. The designer can easily switch between task descriptions simply by clicking on the desired window. The 'UAN spreadsheet' supports simple text editing within cells, easy navigation between cells, setting the size for a row or column, and cutting and pasting text between UAN descriptions. Editing in this manner requires considerably more effort on pencil and paper. While word processors can be used for these editing functions on a single task description, they are not optimized for editing between task descriptions.

4.1.3. Enables higher detail of interaction designs

QUANTUM treats text and graphic annotations, design issue highlights, and audio/video supplements as classes of notes. Modeled after post-it type sticky notes, these notes can be attached anywhere in the Abstraction Hierarchy window or in Task Description windows, *increasing the level of detail that can be specified in an interface design.* Text annotations (called Text notes) may be used for further English explanation of a particular interaction. Graphic notes (called Sketch notes) can be used to include any form of visual supplement in an interface design, such as a state transition diagram, a rough screen sketch to describe a task arena, or even a screen dump of a prototype interface. Design issue highlights (called Issue notes) can be used to identify areas in an interface design that require further attention - this is especially useful in walkthrough reviews of a design. Audio/video annotations (called Audio notes and Video notes) enable the designer to add supplements such as voice recordings of the designer explaining an area of the design, or a video recording of a user interacting with a prototype. As described previously, QUANTUM is designed to be integrated into the IDEAL (Ashlund, 1991) tool lattice. Notes as implemented in QUANTUM will be treated as a separate component of IDEAL.

4.1.4. Provides better document management for interaction designs

The on-line nature of QUANTUM enables *efficient storage, access, management, and distribution of interface designs.* Interface designs can be stored electronically rather than in paper file folders. While this may not be an issue in the consideration of the low cost of paper, electronic storage offers faster transmission and access, making distribution of designs easier. In the case of team development of a user interface design, a software tool offers better project management support by allowing team members to 'check out' portions of an interface design for editing or evaluation.

QUANTUM allows designers to build collections of task descriptions (called Task Libraries) for later reuse. Reuse of task descriptions promotes more rapid development as frequently used task description need not be regenerated for different interface designs. An example of a useful application of Task Libraries is the adherence of an interface design to a set of interface guidelines such as Motif (Open Software Foundation, 1989) or Macintosh (Apple Computer Inc., 1987) guidelines. Designers can build Task Libraries of low-level tasks such as menu selection or icon dragging that follow a prescribed guideline set. These tasks can then be used in later design projects, *ensuring consistency across interface designs.*

4.1.5. Support for new users of UAN

As mentioned before, one of the primary missions of the UAN is to communicate a user interface design from interface designer to interface implementer. Support is needed for easy reading of an interface design by interface implementers, who may be first-time UAN readers. A UAN Reference Sheet lists the UAN symbology in an easy-to-use table for quick lookup. On-line Help for both QUANTUM and UAN is available to help users (interface designers or implementers) should they need it.

4.2. Goal : improve usability of UAN for writers and readers

The goal of QUANTUM is to improve the usability of the UAN. This is accomplished by providing the interface designer with an interactive and powerful environment to easily produce interface designs using the UAN. The increased usability should lead to better quality UAN as the designer is relieved from the concerns of manual manipulation such as formatting and paper management. The researchers do not claim that using QUANTUM will lead to better quality

interface designs, only that UAN produced with QUANTUM has a higher degree of quality due to the increased usability of UAN. The quality of a UAN interface design representation is broadly defined here as a subjective measure of the ease of navigation, expressiveness, and appearance of such representations.

The next chapter outlines the development process of the QUANTUM interface design.

5. The development process for QUANTUM

This chapter outlines the three-stage development process for QUANTUM, detailing early analysis, design stages, and prototype implementation issues. The following diagram illustrates these three stages.

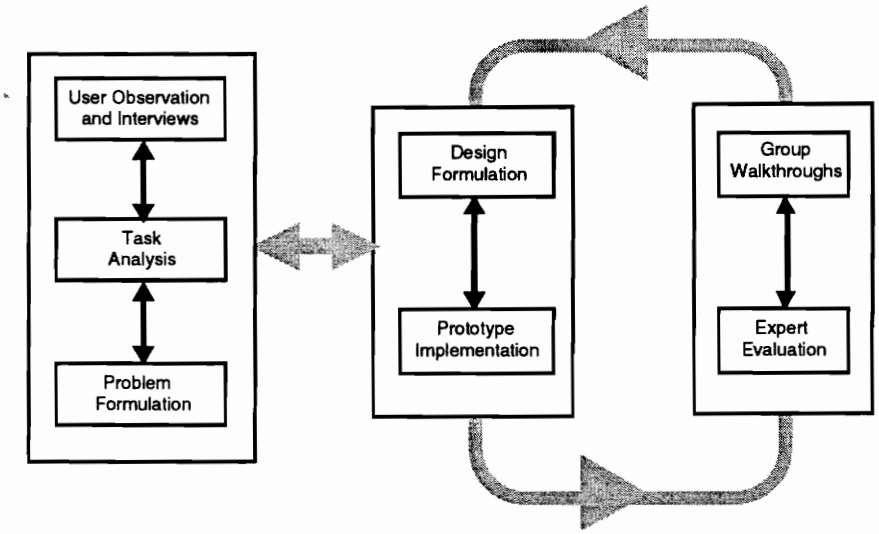


Figure 5.1 QUANTUM development process

5.1. Early analysis

Interviews with UAN users from both academia and industry confirmed the need for a tool to support user interface design with the UAN. Further in-depth interviews led to a better understanding of the tasks and problems involved in using the UAN to represent interface designs. From these interviews, a task analysis hierarchy was created. This section outlines the results of the early activities.

5.1.1. Needs analysis

Needs analysis is the process of identifying the need for a new system to address some particular task or problem and determining the basic goals, purpose, and features of that system (Hix & Hartson, 1993). Since the creation of UAN nearly 5 years ago, many UAN users have remarked on the need for tool support. As described in Chapter 3, interface designs represented in UAN can be quite complex and large, with the number of task descriptions in an interface

design possibly reaching over 100. This does not take into account the management of the linear representation of the task abstraction structure, nor the annotations that may be attached throughout the design. Manual management of such designs becomes impractical, making the need for software tool support clear.

5.1.2. User interviews

Having established the need for an interactive tool for the UAN, users from both academic and commercial settings were interviewed in order to gain a better understanding of the problems inherent in UAN usage. In the interviews, UAN users were asked to describe how they used the notation, list the difficulties they were having, and suggest possible solutions to these problems within the frame work of a software tool.

The use of a graph-based representation of the task abstraction structure was observed at several sites. One industrial site reported the use of a large sheet of paper on which the UAN for the interface being designed was drawn. Each user task was represented as node in a directed graph, and color coding was used to classify different types of user tasks. Other users, especially those new to UAN, drew similar graphs to represent the hierarchical decomposition of user tasks. Discussion with these users revealed that these graphs served as a visual aid that enabled them to better understand the task abstraction structure. Those users unfamiliar with the notation preferred to begin an interface design by drawing the task abstraction structure, and proceed in a top-down manner. Many of the interviewees expressed support for the idea of a software support tool, and several offered suggestions for the design. The following list is a summary of the major tasks that UAN users wanted support for in a software tool.

- Easily write UAN task descriptions
- Use predefined UAN table templates
- Store task descriptions in a reusable format
- Work in a top-down manner starting with the task abstraction structure
- Attach text annotations to complex portions of an interface design
- Easy translation of notation for UAN novices
- Easy navigation among task descriptions

5.1.3. Task analysis

After analyzing the comments made by UAN users, a set of hierarchical task structures was created to drive the QUANTUM user interface design process. These task hierarchies define how the users will use QUANTUM by describing the decomposition of tasks from high abstraction levels (goals, intents) down to the lowest level (user action). This procedure of creating this task structure is described in (Hix & Hartson, 1993), and exactly matches the kind of top-down development UAN users identified in the interviews - designers begin with a graphic representation of the user task structure and later create the UAN for each task.

The following figures illustrate these task hierarchies that were used to create the QUANTUM user interface. Some of the tasks identified in the diagrams (library tasks, note tasks, and abstraction hierarchy tasks) will be explained in Chapter 5.

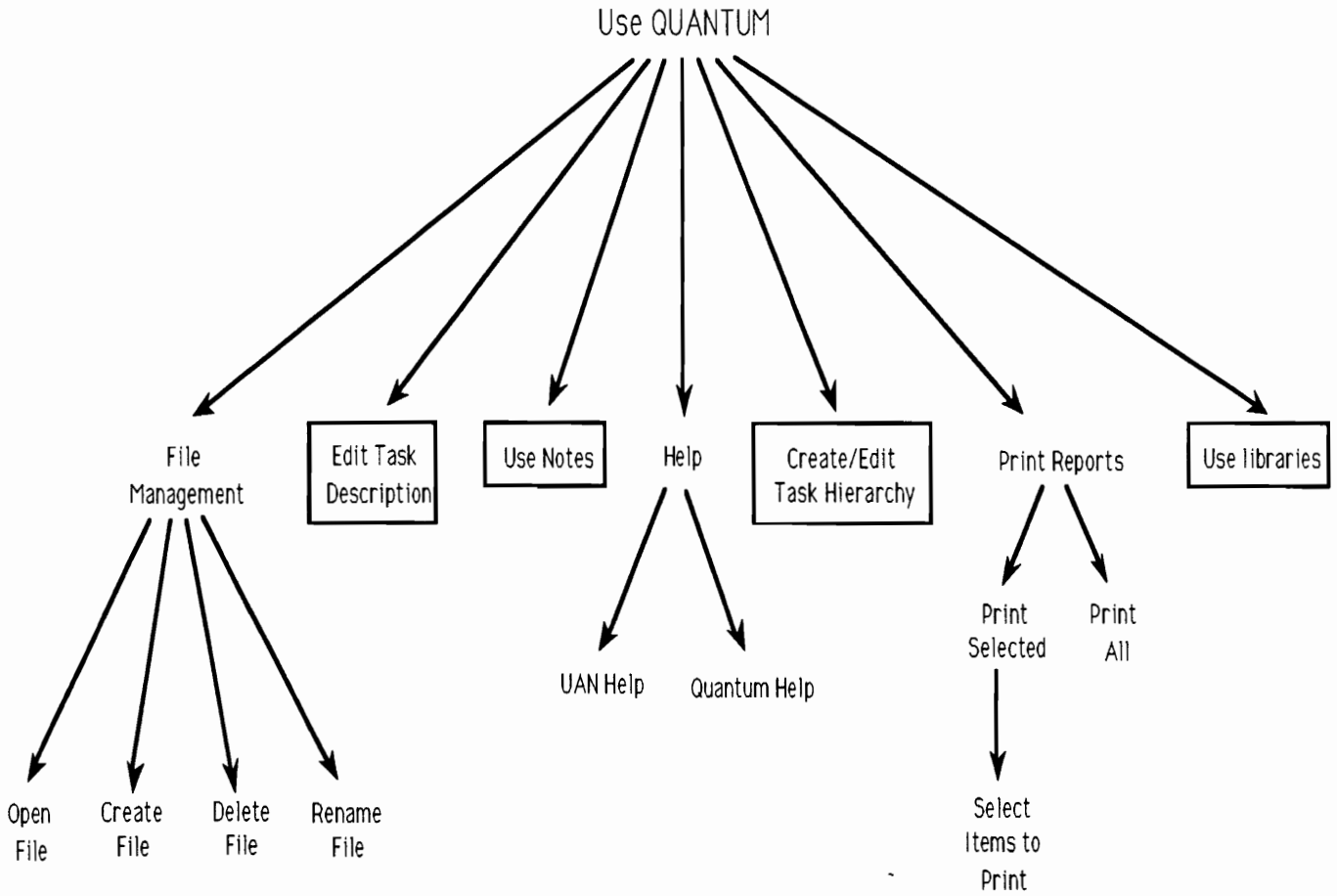


Figure 5.2 Top level hierarchy

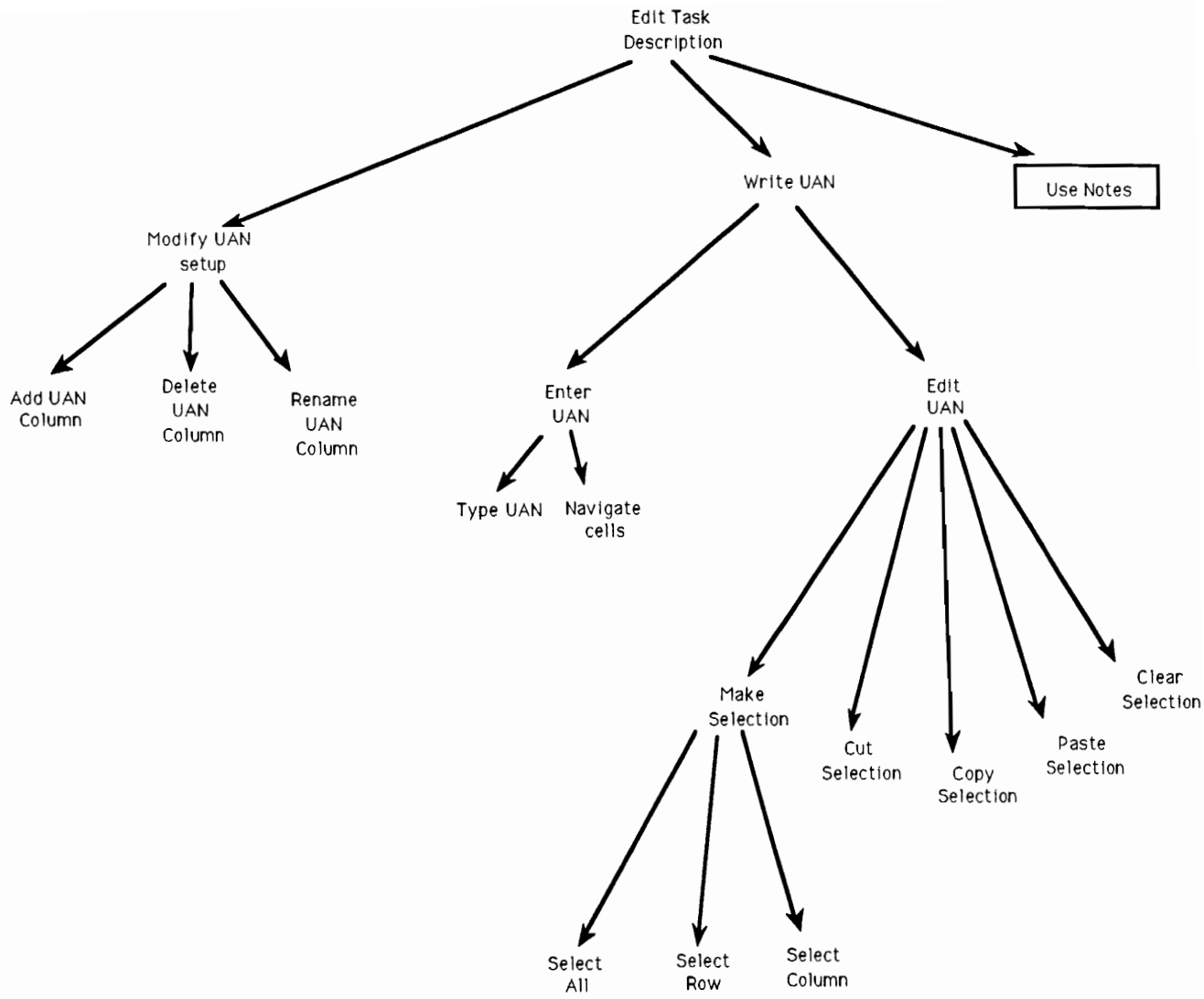


Figure 5.3 UAN task description hierarchy

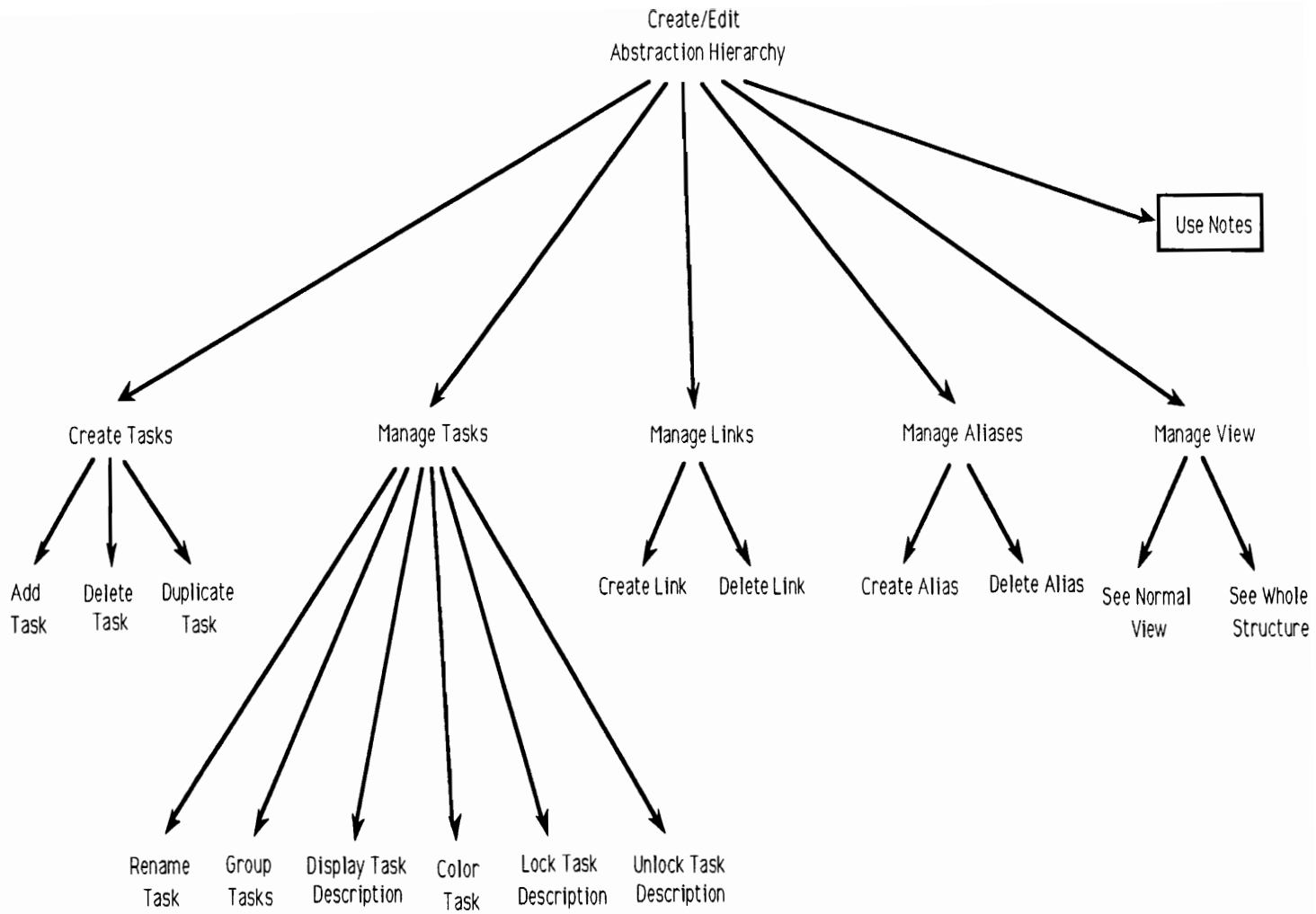


Figure 5.4 Abstraction structure hierarchy

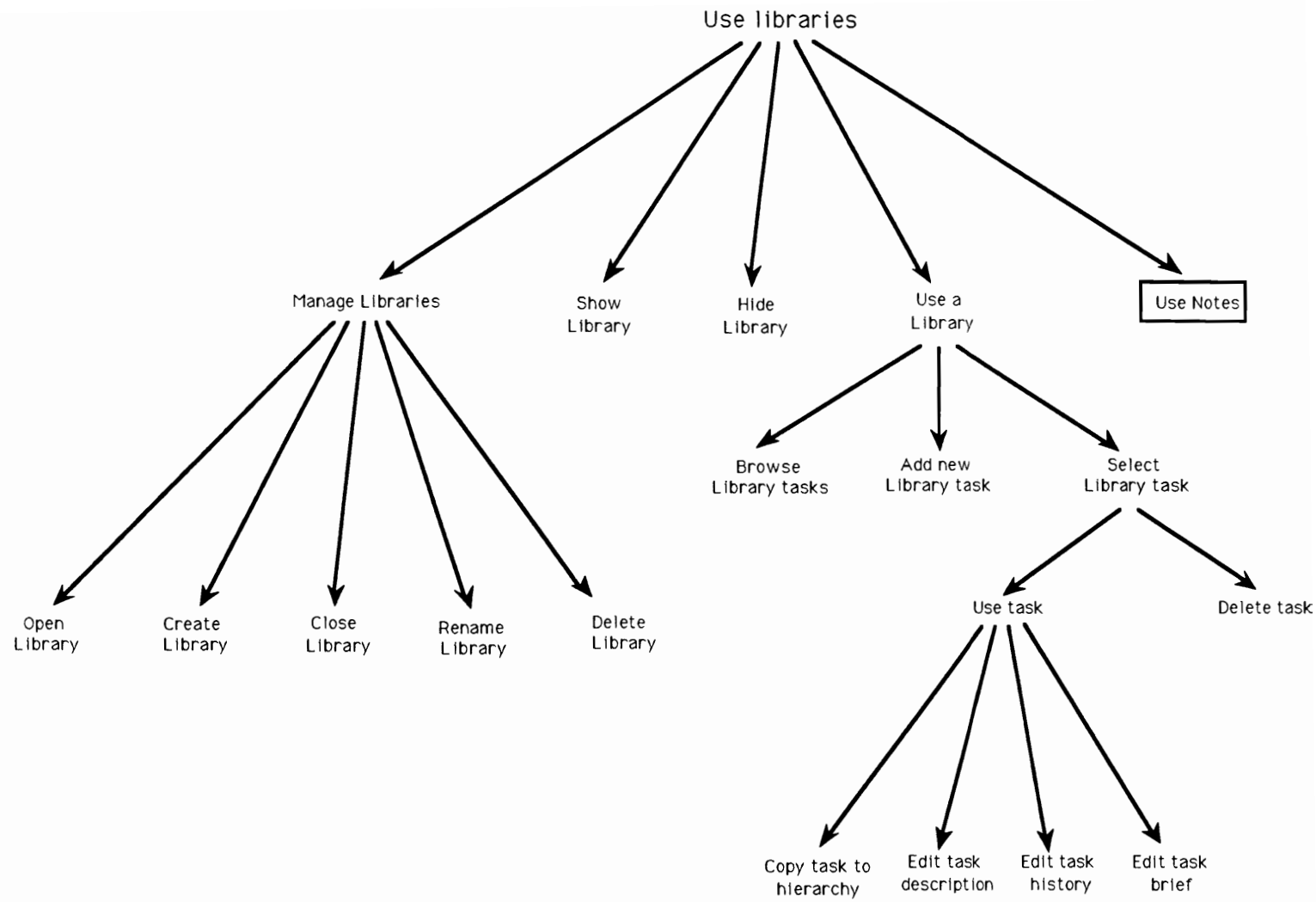


Figure 5.5 Task Library hierarchy

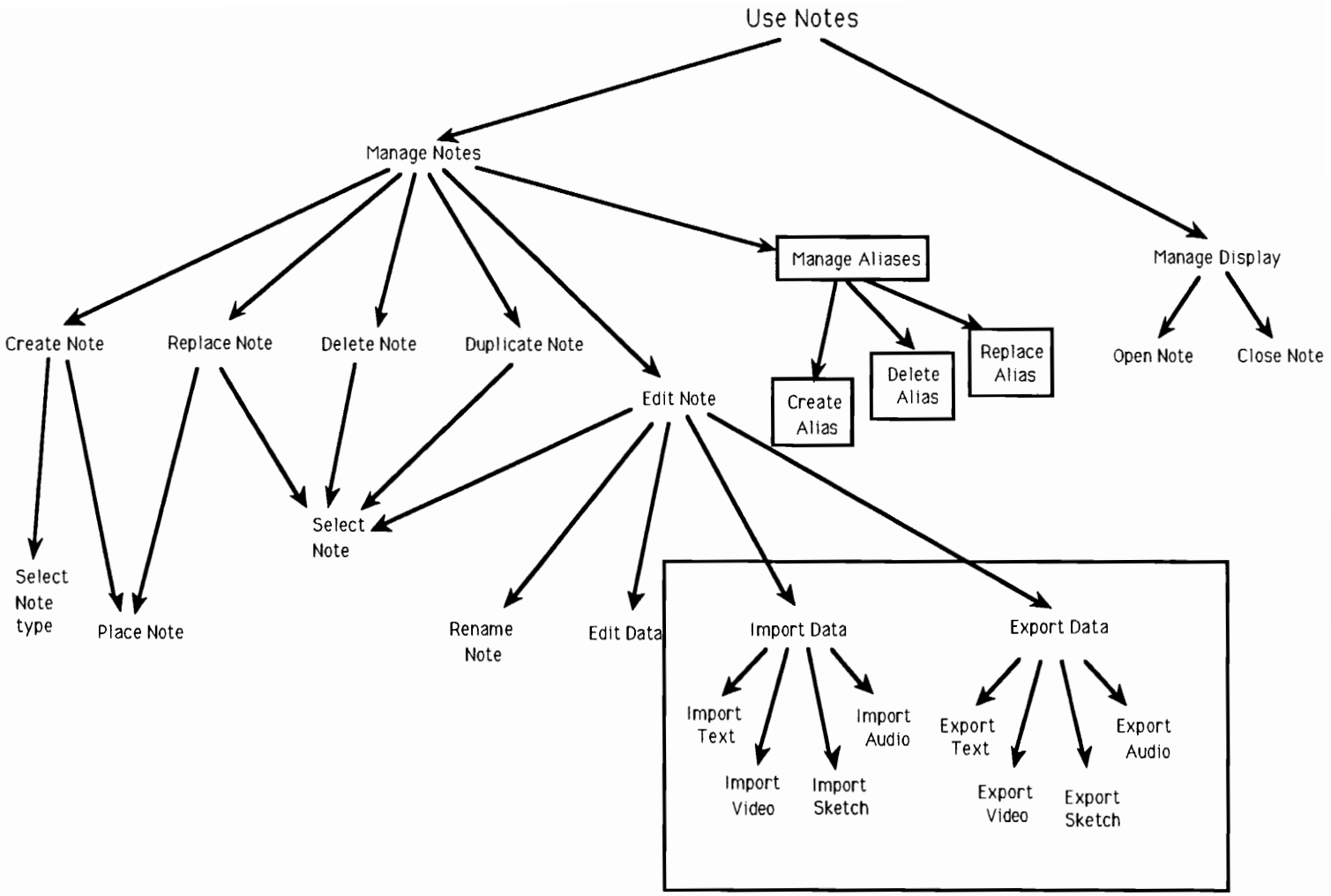


Figure 5.6 Notes hierarchy

5.2. Iterative refinement

User interface design is inherently iterative, meaning that frequent evaluation and redesign are necessary to improve usability (Hix & Hartson, 1993). The QUANTUM interface design was subjected to numerous evaluation by way of group walkthroughs and expert consultation. This section highlights some of the stages of the iterative refinement of the QUANTUM interface design.

Early attempts at a design for a UAN tool (Hartson, Hix, & Siochi, 1989) were made by members of the Dialogue Management Project at Virginia Tech, a group of HCI researchers investigating the achievement of quality user interface systems through development of tools, methodologies, and processes. One of the first issues addressed was the need for easy entry of UAN in some predefined template. Initial designs incorporated a scrolling window in which UAN was formatted into three columns: User Action, Interface Feedback, and System State. Screen sketches were closely tied to the UAN description for a task. Below is an example of an early attempt at the design for a task description window.

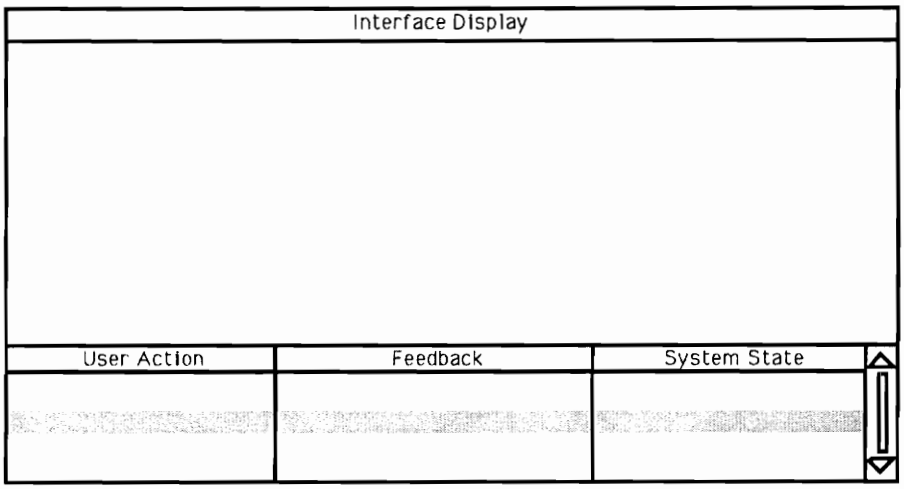


Figure 5.7 Initial design for Task Description Windows, from (Hartson, et al., 1989)

The user was afforded the ability to 'play back' a UAN task description. The tool was to step through the task description one line of UAN at a time, updating an action screen at each step. The action screen occupied the upper two-thirds of a movable window, with the other third occupied by a scrollable UAN description. A highlight bar across the UAN description was to highlight the action being represented at any particular time on the action screen. Several of

these UAN windows were to be displayed on the screen at once, allowing the user to work in any window by simply selecting it. This design had several problems, including the necessity of the user to construct a screen sketch for each point in a task execution, the use of too much screen real estate for such windows, and the complexity of implementation.

The idea of a screen sketch was then incorporated into a separate window from the UAN task description. Other refinements were made to the design, including a window to represent the task abstraction structure in an easily recognizable format. This Abstraction Hierarchy window came directly out of user remarks on the difficult navigation of a UAN interface design. The task abstraction structure for an interface design is an instance of a k-ary tree, which is represented in the Abstraction Hierarchy window. Below is an example of an early design for this window.

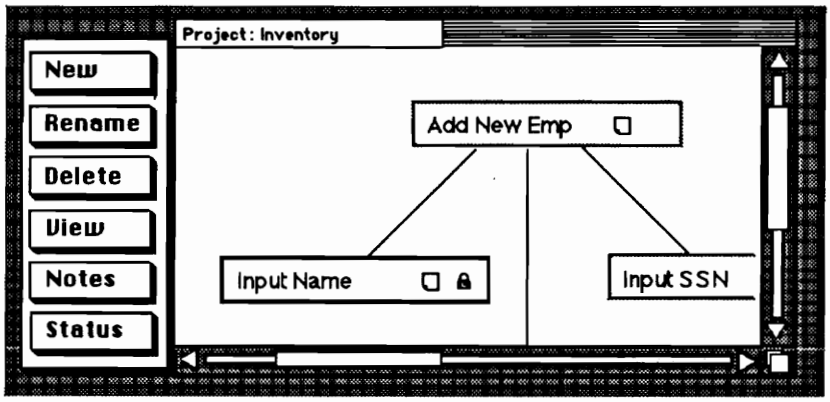


Figure 5.8 Early Abstraction Hierarchy Window and Palette

A palette of buttons to the left of this Abstraction Hierarchy window offered the designer a set of commands applicable to the window. The designer could create a new task simply by clicking on a button labeled 'New.' Other buttons on the palette allowed the designer to rename tasks, delete tasks, view task descriptions, view annotations, and check the completion status of a task. One problem with this design was the ambiguity of the text labels used on the command palette. In addition, this palette was also attached to the window, and could not be moved separately. After evaluation by interface design experts, this palette was moved into its own window, and visually onomatopoeic icons were used in place of the text labels.

Several other portions of the QUANTUM interface design were changed, due in large part to the use of frequent **design walkthroughs**. Walkthroughs offer a relatively low-cost and efficient way to evaluate a design (Lewis, Polson, Wharton, & Rieman, 1990) (Karat, Campbell, &

Fiegel, 1992)(Rowley & Rhoades, 1992). Over the course of one year, QUANTUM interface designs were subjected to numerous walkthroughs by a panel consisting of both interface design experts and UAN users. The design experts brought their experience with interface design guidelines to the walkthrough sessions, while UAN users offered their domain-specific knowledge of the tasks involved in UAN usage. In these walkthroughs, the panel was guided through a set of sample tasks within the proposed interface. The researchers used overheads of screen sketches and verbal description to simulate the interface for members of the panel playing the part of QUANTUM users. Through this interaction, the panel identified problems in the interface design and suggested possible approaches to solving them. Many of the comments that came out of these walkthroughs focused on issues such as window management, clarity of icons used, and feedback details such as highlighting methods. The overall result of these walkthroughs was the efficient identification and correction of many fundamental design errors.

5.3. Prototype implementation

While walkthroughs can identify major problems in an interface design, it is commonly accepted that the most efficient manner for ensuring the usability of a design is to construct a prototype that can be subjected to evaluation by users. Prototyping and evaluation offers the advantages of early detection of interface design problems and early opportunity for refinement (Hartson & Smith, 1991) (Wasserman & Shewmake, 1985). After the QUANTUM interface design was put through 2 walkthrough sessions, work began on the construction of a prototype. Two undergraduate researchers were hired to aid in the programming required for prototypes of both the QUANTUM interface design and the IDEAL interface design. This simultaneous development of the prototypes and designs was intended to ensure consistency between QUANTUM and IDEAL.

Several prototyping systems were investigated for their suitability as platforms for the prototype implementation. In principle, an ideal prototype platform should have no effect on the implementation interface design. In practice however, the limitations of the platform used affect the speed of the implementation and dictate what can and cannot be prototyped. After consideration of several Macintosh-based systems (HyperCard, SuperCard, and Wingz) and a UIMS (ezX(Sunrise Software International Incorporated, 1992)), the Tcl/Tk system was chosen. **Tcl** and **Tk** (Ousterhout, to be published 1993) are two parts of a public-domain system that provides a programming platform for developing and using windowing applications. Tcl (tool command language) is an interpretive programming language implemented as a library of C

procedures. Tk is a toolkit for the X11 window system, and is also implemented as a library of C procedures. Using Tcl and Tk, programmers can easily create windowing applications by letting Tcl/Tk handle the code for managing the interface. One of the most powerful features of the Tcl/Tk system is the ability to write executable scripts in a windowing shell called `wish`, which incorporates both Tcl and Tk. Similar to scripts for shell programs such as `csh` under UNIX, these scripts are so efficiently and quickly interpreted by the `wish` shell that they can function as useful applications. This approach was used for the prototyping of the QUANTUM design. The Tcl/Tk system, including the `wish` shell, was installed on a DECstation 5000 Model 100 running ULTRIX version 4.3 and the X11R5 window system.

Through the efforts of the two undergraduate researchers and the author, a prototype was constructed by writing a collection of `wish` scripts that created the QUANTUM interface design. The speed of prototype implementation using the Tcl/Tk system enabled frequent walkthroughs of the design via interaction with the partially constructed prototype. Because the designs for individual artifacts such as icons and windows could be quickly implemented, they could be immediately evaluated by informal interaction sessions with expert interface designers. These frequent interactions led to further identification of design problems that were discussed at the walkthrough sessions. This efficiency enabled a large number of necessary changes to be quickly and easily made to the QUANTUM interface design. Without this efficiency, the QUANTUM interface design would have taken significantly longer to bring to a point where it was ready for expert evaluation.

The next chapter outlines the QUANTUM interface design as presented to the expert evaluators.

6. Overview of QUANTUM prototype

This chapter presents an overview of the QUANTUM interface prototype as implemented and presented to the expert evaluators. This overview is not intended to serve as a detailed explanation of the operation of QUANTUM. For details on specifics of usage, the reader is referred to a separate User's Manual document. Some of the items described in this design overview were not functionally implemented in the prototype, such as global editing functions, audio/video capabilities, and full saving and loading features. These functions were not implemented due to time and resource constraints, but where applicable 'stubs' representing a non-functional item were used so that the evaluators could still analyze the interface.

QUANTUM is intended to provide user interface designers with an integrated environment to facilitate UAN usage. QUANTUM can also be used by interface implementers or usability specialists who may wish to examine an interface design for purposes of implementation or evaluation. The QUANTUM user interface was designed to afford a high degree of flexibility and power, while maintaining an interaction style familiar to users experienced with direct-manipulation interfaces. To offer this high degree of flexibility, the QUANTUM interface is designed to maximize the user's control of flow (Norman, 1991), a concept that addresses whether the user or computer directs the course of events in the cooperative performance of tasks. In computer-guided systems, the user may be directed through a structured series of prompts and steps in order to accomplish a task. In QUANTUM, the user can start, stop, and resume tasks at any point. This behavior more naturally matches the usage of UAN as observed during the needs and task analysis stages of the QUANTUM design development. Several different types of windows are used to present different components of a UAN interface design such as task descriptions, the user task abstraction structure and annotations. Other windows are used for QUANTUM-specific artifacts such as task libraries, notation reference sheets, and five different types of annotations. The rest of this chapter presents these windows in detail.

Several assumptions were made regarding the platform upon which a UAN support tool would run. QUANTUM was designed to run on a UNIX platform running XWindows. The QUANTUM interface design was built on the assumption that user interface designers would have access to computers that have color capability and employ a large (16 inch or greater) screen. Considering the shrinking cost of computer hardware, this assumption was deemed

reasonable. It was also assumed that user interface designers would have access to computer supported audio and video recording tools. In this regard, the notion of audio/video capability was integrated into the design in the form of audio/video notes which are described later in this chapter.

6.1. Abstraction Hierarchy window

QUANTUM centers around a graphical representation of the user task abstraction structure, displayed in a window entitled "Abstraction Hierarchy." This window, like all windows within QUANTUM (with the exception of dialogs), is resizable and positionable anywhere on the screen. The diagram below illustrates the Abstraction Hierarchy window.

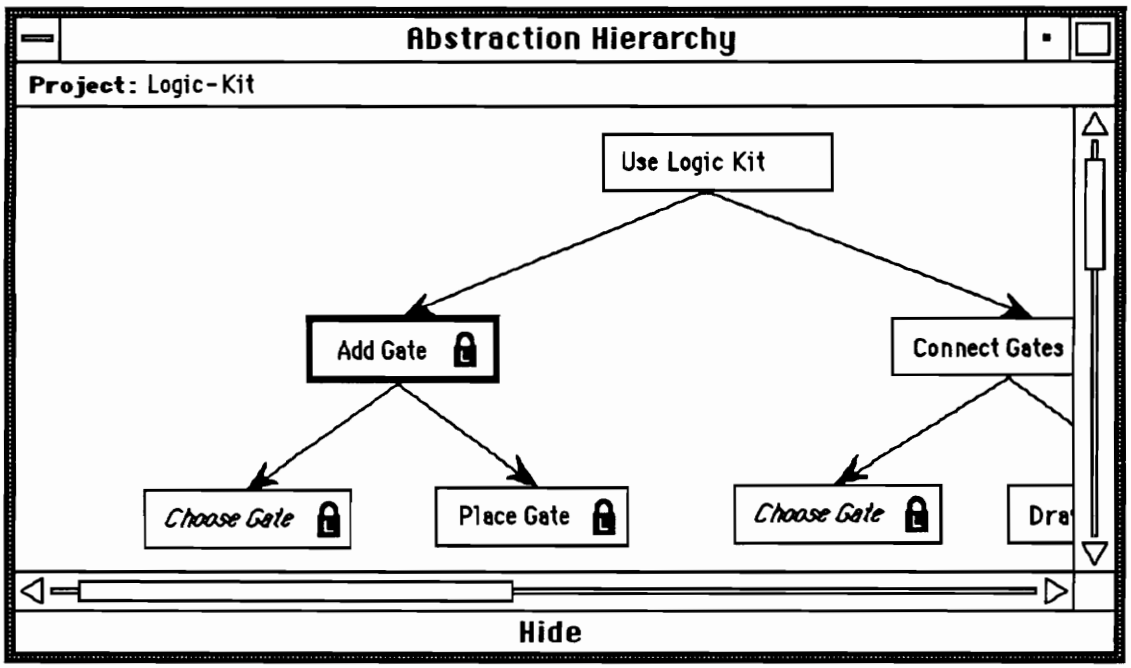


Figure 6.1 Abstraction Hierarchy Window

All information such as task description, annotations, and task hierarchy structures are abstracted into the concept of a user interface design project. A line at the top of the window displays the name of the interface design project that is currently loaded. In this first version of QUANTUM, only one interface design project can be worked on at one time. All windows (with the exception of dialog windows) contain a Hide button at the bottom of the window that removes the window from view - these windows can later be redisplayed. This ability to remove

unneeded windows from the screen enables the user to keep the number of windows that must be managed down to a reasonable amount.

Each user task is represented as a node in a directed graph resembling a k-ary tree. This graph is displayed on a virtual canvas, a portion of which is shown in the window. By either resizing the window or using the scrollbars attached to the sides of the canvas, the user can see as little or as much of the task structure as desired.

Figure 6.2 Example of locked task description node

Nodes in the task structure are called task icons, and are labeled with the name of the task being represented. QUANTUM allows the designer to color-code tasks in whatever manner they desire. The designer may elect to let color represent the completion status of a task, or represent the type of a task (input/output, dialog, etc.). Each task icon is hyperlinked to a window containing the appropriate task description. In the above example, the task icon for the task entitled "Add Gate" is shown. The image of a padlock is used to indicated that this task description is read-only. The ability to lock task descriptions in this manner supports the development of portions of an interface deign while keeping other portions locked. Double-clicking on a task icon with the mouse opens the task description window associated with that task. This window is described later.

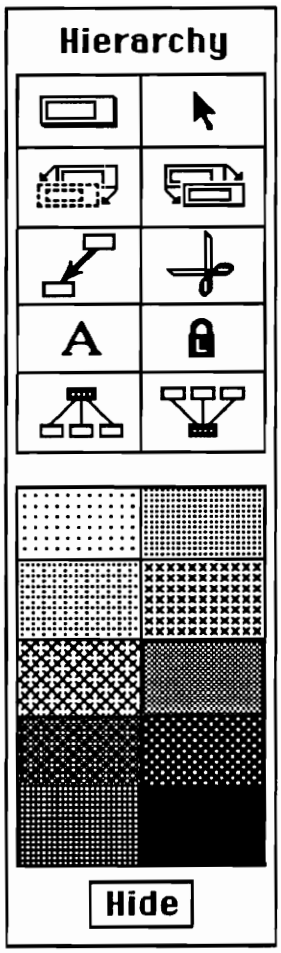


Figure 6.3 Hierarchy Palette

The **Hierarchy palette** (picture above) presents the user with a set of operations that are applicable only on the Abstraction Hierarchy Window. The icons used on this palette were the result of several iterations through candidate symbols. Hemenway (Hemenway, 1982) suggested that when icons share common characteristics, the user has less to learn since familiarity with one icon transfers to a new icon. With this in mind, most of the icons on this palette use a task node as an element of the symbol.

The QUANTUM user can the task tool, located at the top left of the palette, to create new tasks. After choosing the tool from the palette, the user placing a new task icon within the Abstraction Hierarchy by clicking in the desired position. A task icon will appear at that point.

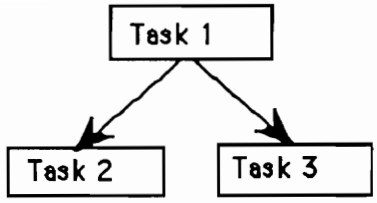


Figure 6.4 Example of task links

The user can then use the link tool to illustrate a dependency link between two tasks. Links may be created and deleted as desired. A link between a task and a subtask is represented by a directed arrow. In the above diagram, Task1 is a parent to child tasks Task2 and Task3. Task1 is a task at a higher level of abstraction than its child tasks Task2 and Task3, which can be thought of as subtasks in the description for the supertask Task1. The set of all tasks that are parent to some task A is called the 'Fan-in' of task A. Likewise, the set of all tasks that are child to some task A is called the 'Fan-out' of task A. Views of these sets for any task are available to the user by way of a separate window containing a scrolling list of task names. By adding task icons and defining dependency links, the user interface designer constructs a tree of task dependencies. By traveling to deeper levels in this tree, the user tasks are at lower levels of abstraction. Direct manipulation allows the user to move task icons within the window, while the dependency links remain attached. This enables designers to rearrange the abstraction structure in order to create a more appealing layout.

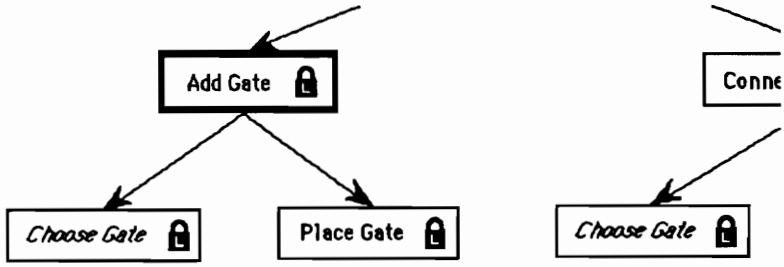


Figure 6.5 Example of aliasing

Certain low level tasks such as keyboard entry of a string or selection from a menu are usually used by several tasks. Aliases allow the designer to use multiple icons to represent one task at several places in the Abstraction Hierarchy, reducing the complexity of the graph representation. In the above figure, the task 'Choose Gate' is aliased. This allows it to be shown as a subtask to both 'Add Gate' and 'Connect Gate'

Other commands available on the Hierarchy palette enable the user to make a separate copy of a task and to rename a task.

6.2. Task Description windows

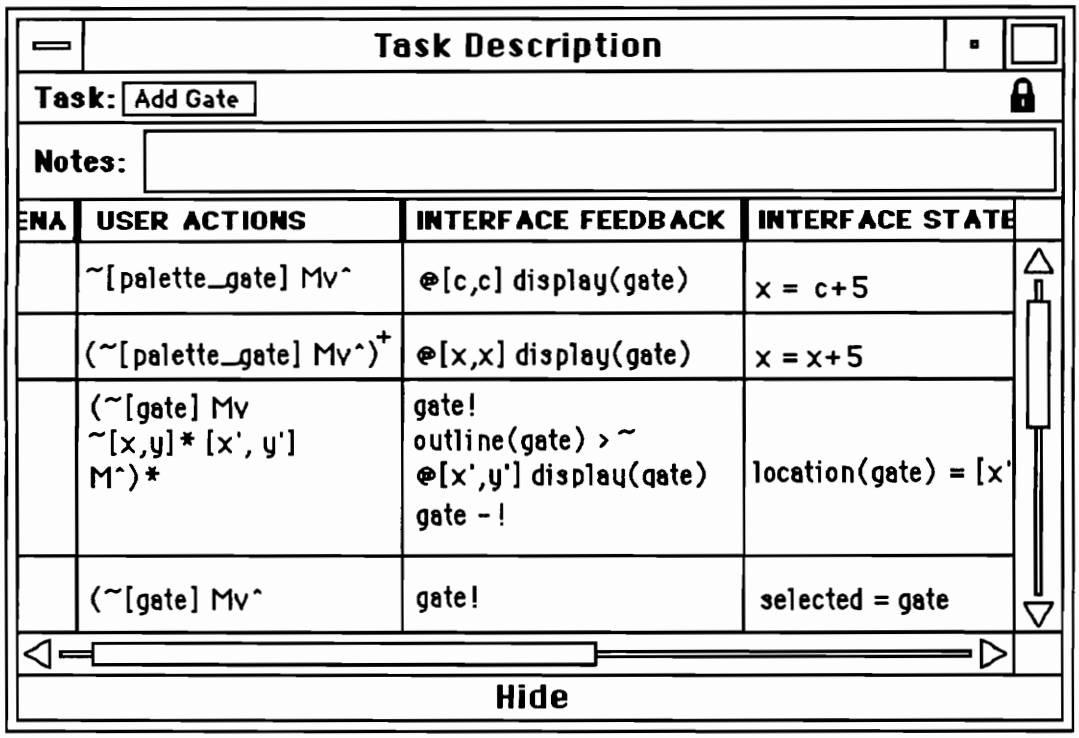


Figure 6.6 Task Description Window

Task Description windows hold the UAN task description for user tasks. The name of the task is displayed at the top of the window. The color of the task icon(s) in the Abstraction Hierarchy is used as a background color for the task name. If the color used for the task icon is changed, this background color will change to match. If the task is locked against change from the Abstraction Hierarchy window, a lock symbol will appear at the upper right of the window.

The main portion of the Task Description window is the Spreadsheet area(pictured below) This Spreadsheet area is similar to spreadsheets used in accounting programs. Each cell holds UAN, and navigation within and among cells is accomplished with both the keyboard and the mouse. The user can select across columns for pasting into other locations, and column widths are resizable. Spreadsheets are pre-configured with four columns, Arena, User Action,

Interface Feedback, and System State. The user can rename columns or add new columns as desired.

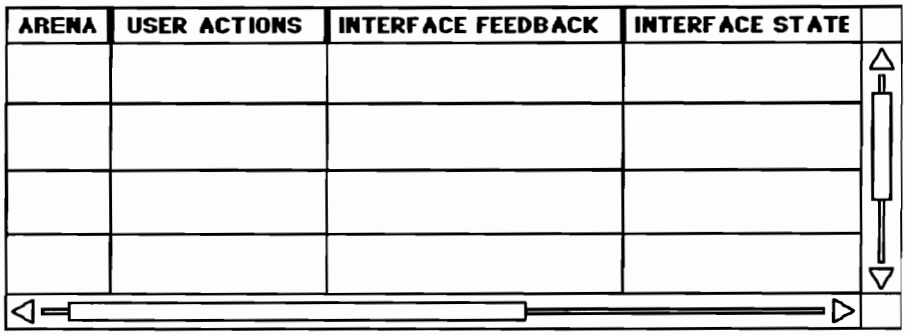


Figure 6.7 Spreadsheet area for a Task Description Window

The **UAN palette** enables the user to insert predefined UAN strings into task description windows. These strings are user-definable, allowing, the designer to create a set of frequently used UAN strings to save time while creating task descriptions. This palette is pictured below.

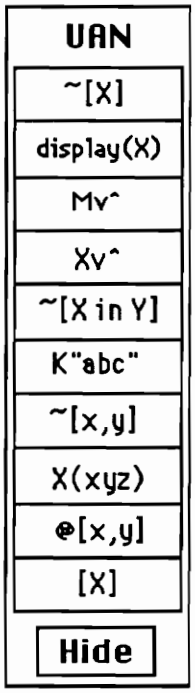


Figure 6.8 UAN Palette

This palette, as well as the Hierarchy palette described earlier, and the Note palette (described later) can be hidden to save screen real estate by clicking on the Hide button.

6.3. Libraries

Task Libraries windows (pictured below) provide access to task library contents. Task libraries allow the user to build archives of pre-defined UAN task descriptions. The libraries can store frequently used low-level tasks such as file selection or menu selection. The stored tasks can be reused later in the design of an interface, or in the interface design for a different project. This ability to share task descriptions across interface design projects can ensure consistency of behavior across applications, as common low-level tasks in these application have the same behavior. Another application of these archives is ensuring the adherence of an interface design to a particular user interface guideline specification, such as the Macintosh desktop (Apple Computer Inc., 1987), or Motif (Open Software Foundation, 1989). Low-level tasks designed to adhere to a standard such as these can be retrieved from a 'guideline library', saving the designer from having to recreate the task.

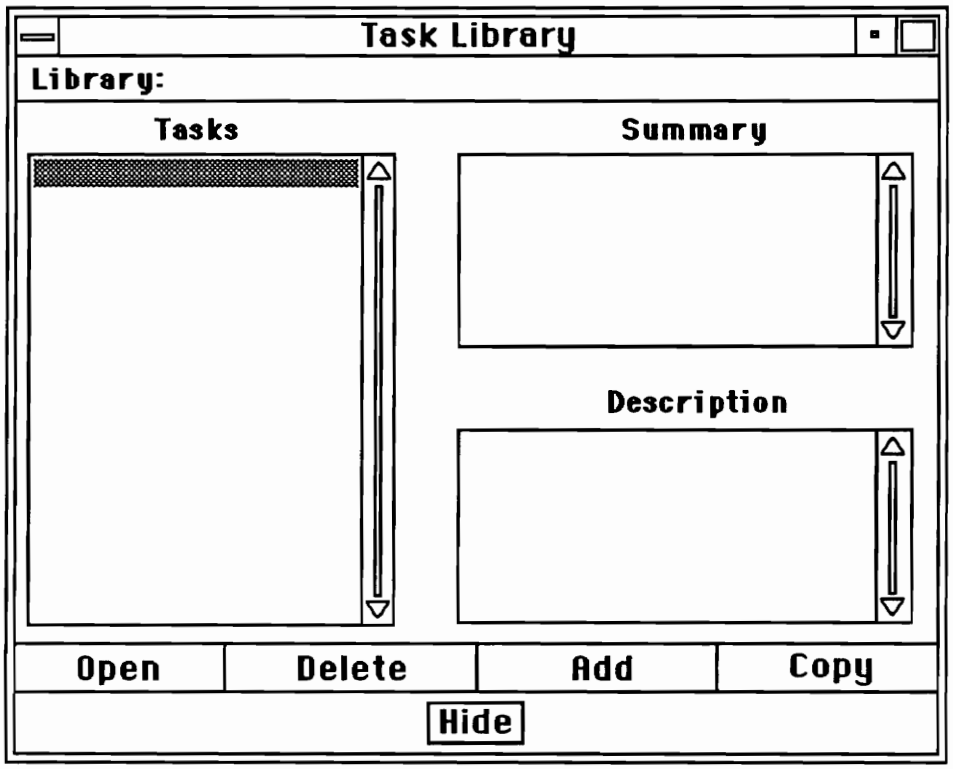


Figure 6.9 Task Library window

A scrolling list on the left of the window called the Task list, lists the tasks contained in the library. Selection of a task is made with either the arrow keys or the mouse. When a task is selected, a brief summary of the task is displayed in the Summary field, and the usage history for

that task (date checked into library, date created, date last used, etc.) is displayed in the History field. A bank of buttons offer operations on the selected tasks. Opening a task allows the user to open the task description window for the selected task in order to view the UAN for that task. This task description window is read-only, and cannot be modified. Deleting a task removes it from the library. Adding a task adds the currently selected task in the Abstraction Hierarchy to this library. Copying a task adds this task to the current project, and creates a new task icon for this task in the Abstraction hierarchy window.

6.4. Notes

Modeled after 'post-it' type notes, these notes can be attached at three different places within the Abstraction Hierarchy window or Task Description windows. Five different types of notes are offered, **Textnotes, Sketchnotes, Issuenotes, Audionotes,** and **Videonotes.** When a note is 'attached', an icon representing the note is shown at the attachment point. This icon can be moved only within the window it is placed. When the note is opened (with a double click), the user is presented with a window for that note for editing and manipulation.

Notes can be attached to a Task Description window in two areas : the Notebar, or within the Spreadsheet area.

Figure 6.10 Detail of Task Description window : Notebar

The Notebar allows the user to attach notes to a task description at the top level, rather than 'in-line'. This Notebar allows the user to specify that a particular relates to the task as a whole, and not 'in-line'.

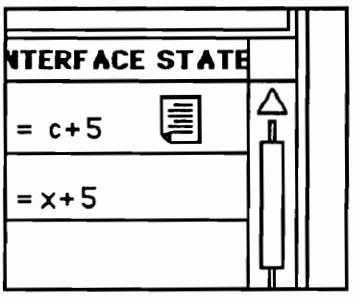


Figure 6.11 Textnote attached within a Task Description window

The above figure illustrates a Textnote that has been attached within the Spreadsheet of a Task Description window.

Notes are created by choosing a note tool from the **Note palette** (pictured below). This palette contains buttons labeled with the icon for each note type. When a note tool is chosen, the cursor changes into the shape of the appropriate note icon, and the user places the note by clicking in either the Abstraction hierarchy or the Notebar or Spreadsheet for a Task Description window. the note icon may be moved within the window after it is placed.

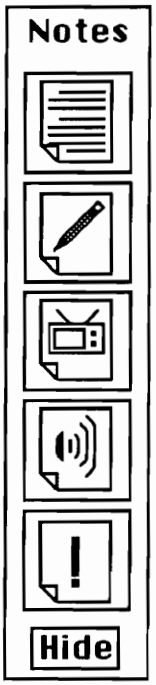


Figure 6.12 Note Palette

Textnotes are the simplest note type, allowing the user to place textual annotations anywhere in an interface design. The window holds a field for simple text entry and editing. a Textnote window is pictured below.

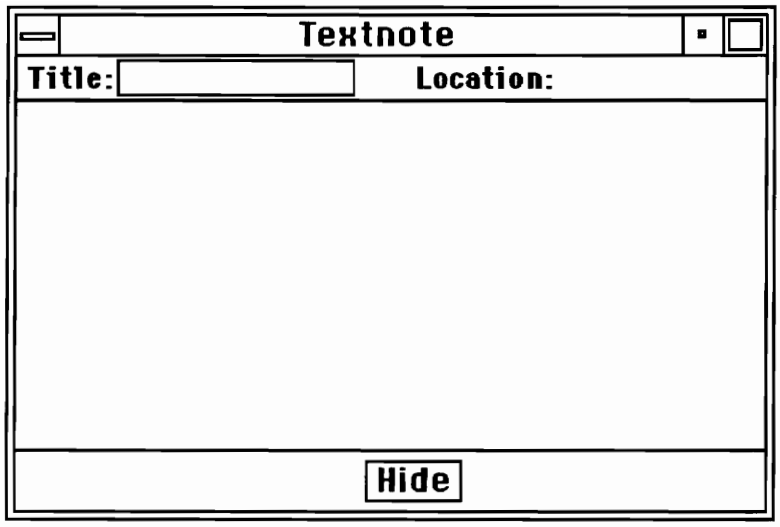


Figure 6.13 Textnote window

Sketchnotes give the user the equivalent of a drawing pad, allowing the user to incorporate graphical annotations into an interface design, such as screen sketches (to describe task scenarios) or state transition diagrams (to augment task descriptions). A bank of tools is available on all Sketchnote windows, providing the user with simple object drawing functions. Objects can be moved, colored, and resized in any manner. A Sketchnote window is shown below.

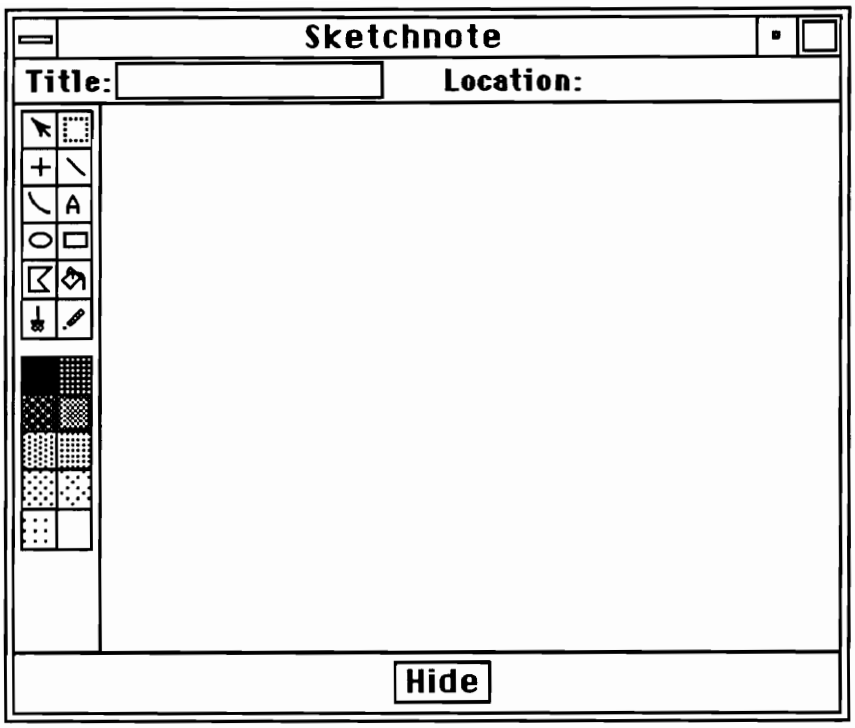


Figure 6.14 Sketchnote window

Issuenotes allow the user to highlight portions of a user interface design that need further attention or redesign. The idea for this note type came from the observation that UAN users attached 'post-it' type adhesive notes to paper UAN interface descriptions. These adhesive notes would perhaps highlight a problem in the design of a particular interface widget. Issuenotes provide a Problem field for a textual description of the deign issue, and a Discussion field for problem discussion. An Issuenote window is shown below.

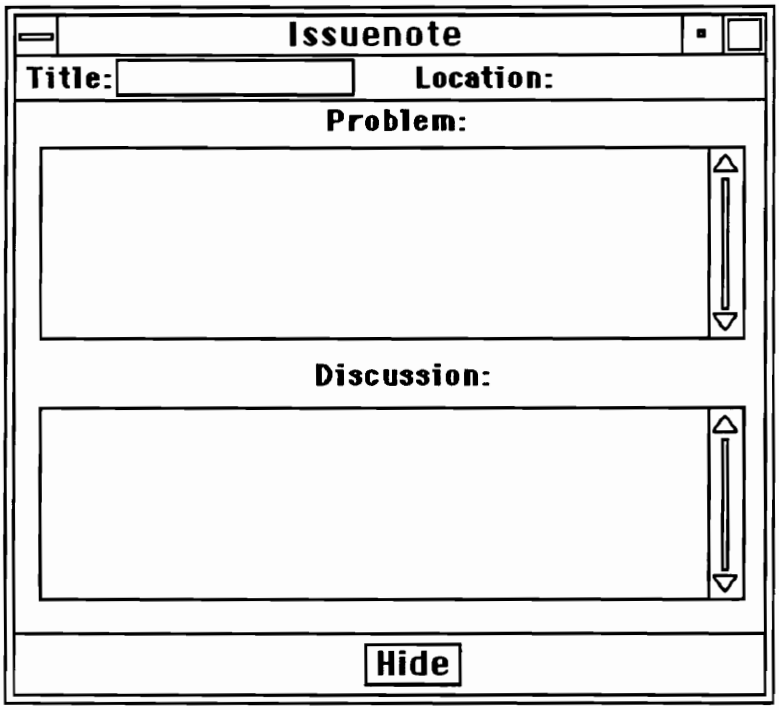


Figure 6.15 Issuenote window

Audionotes and Videonotes provide the user with a means for attaching audio/video recording clips to any point in an interface design. As discussed in Chapter 2, Notes are designed to be an artifact of the IDEAL architecture. Results of their design and prototyping within the QUANTUM design will propagate upwards into the design and prototyping of IDEAL. Several possible uses of audio/video clips to an interface design exist, for example an interface designer could include a voice description of a particular interaction sequence written in UAN to enrich the interface description. A video recording of a user interacting with a prototype could be attached to the UAN description of the task being performed. Audionote and Videonote windows are shown below.

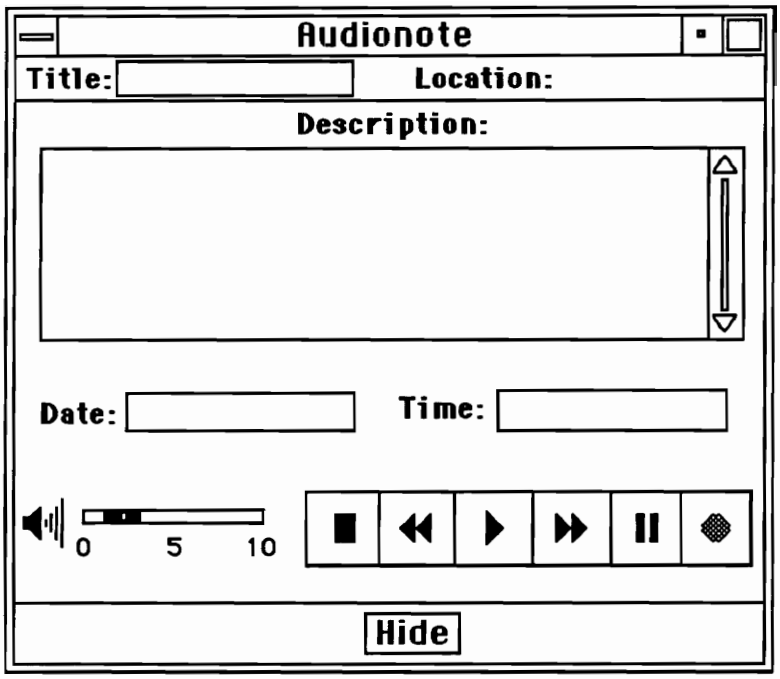


Figure 6.16 Audionote window

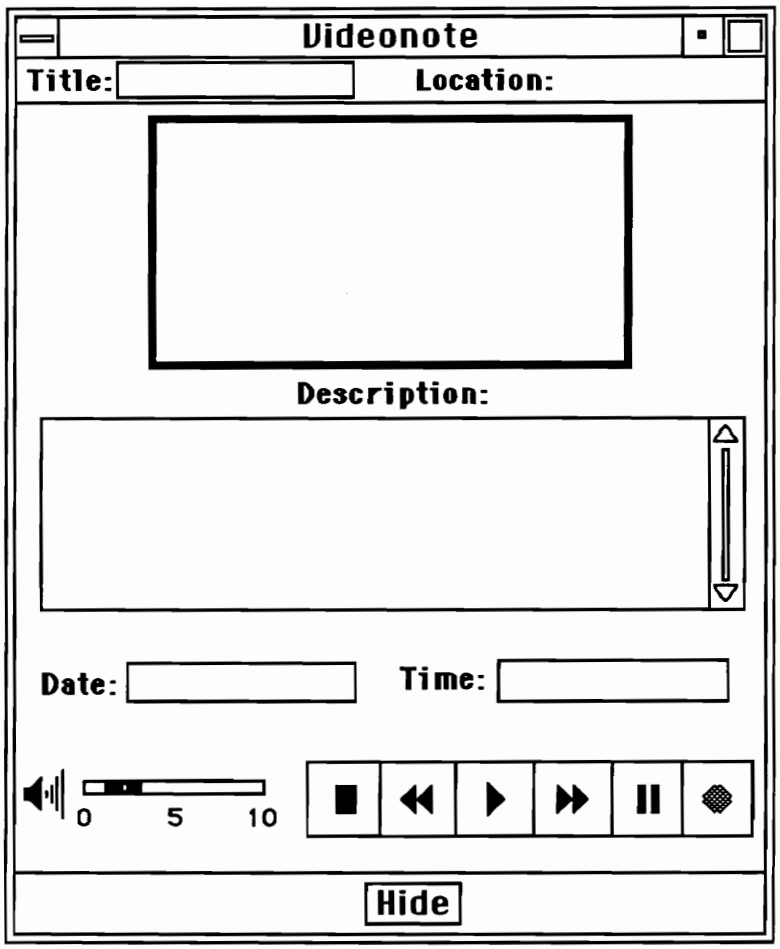


Figure 6.17 Videonote window

The Audionote and Videonote windows are identical with the exception of a video screen in the Videonote window. A Description field allows the entry of a text describing the recording. A volume control lets the user set the speaker level, and a bank of buttons along the bottom of the wind allow the user to respectively stop, rewind, play, fast forward, pause and record either a video clip or a sound clip.

The **UAN reference** (pictured below) is a window displaying the current symbology of the UAN, along with the meaning and an usage example for each symbol. A scrolling Symbol list on the left lists the UAN symbols. When a UAN symbol is selected, its meaning and an example of how to use that symbol is displayed in the Meaning and Example fields on the right of the window.

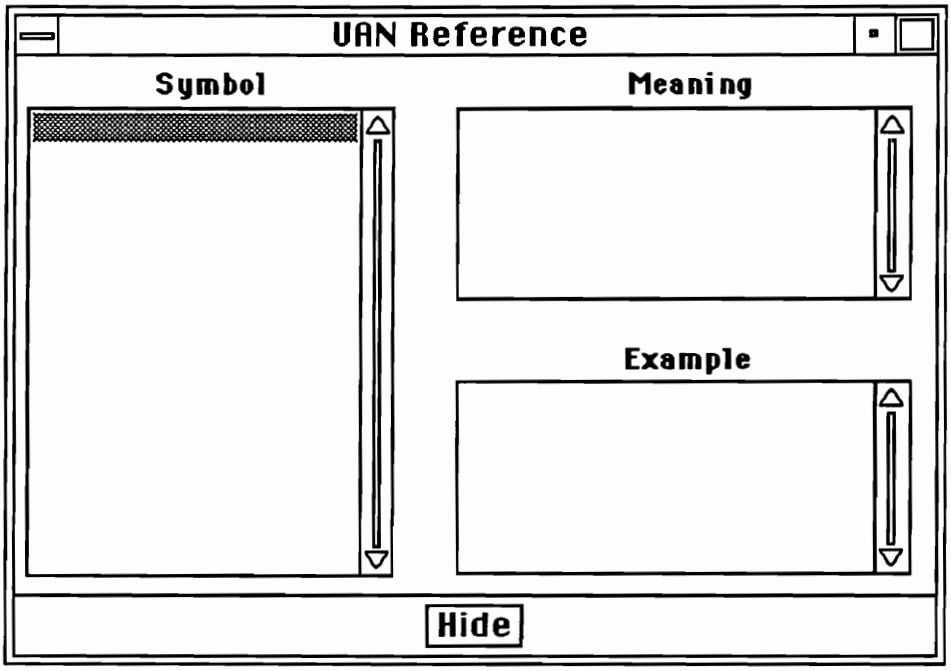


Figure 6.18 UAN Reference window

The UAN Reference serves as a helpful reference for new UAN users who may be unfamiliar with the notation.

6.5. Menubar

The QUANTUM Menubar is made up of 9 menus: Project, Edit, Description, Hierarchy, Notes, Library, Report, Utilities, and Help. Commands within these menus were grouped in this manner to provide semantic organization of the tasks. Guidelines taken from (Norman, 1991) and (Shneiderman, 1987) were applied in the organization of these menus. A detailed description of the menu contents is given in a separate QUANTUM User Manual document. All of the commands available from the Hierarchy palette and the Note palette are duplicated in the Hierarchy and the Notes menus, respectively. This provides redundancy for the user, allowing them more than one way to perform these tasks such as creating a new task in the Abstraction Hierarchy and creating a new Note.

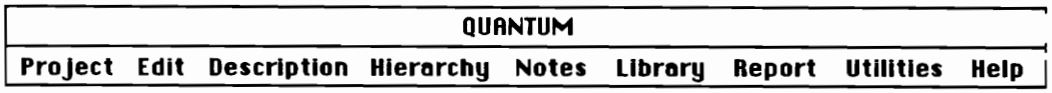


Figure 6.19 QUANTUM Menubar

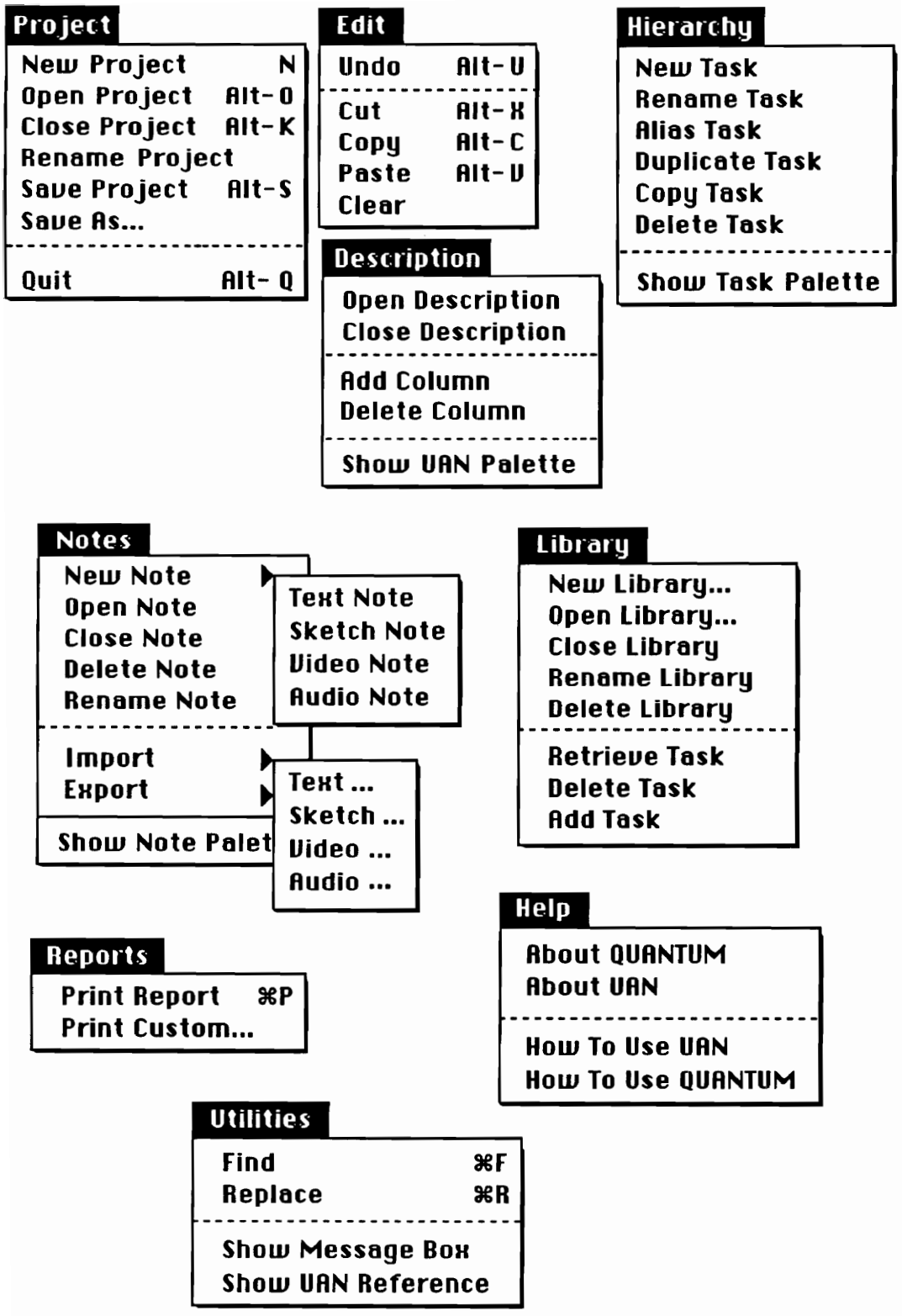


Figure 6.20 QUANTUM menus

The Project menu provides the user with commands to create a new project, open a project stored on disk, and save and close the current project. The Edit menu offers undo, cut, copy, paste, and clear operations that are applicable to all QUANTUM windows. (These functionalities were not implemented in the prototype due to time constraints.) From the Description menu, the user can add and delete columns to the Task Description Spreadsheets. The Hierarchy menu duplicates the contents of the Hierarchy palette, and adds the ability to delete the currently selected task within the Abstraction Hierarchy window. The Notes menu duplicates the contents of the Notes palette, and adds the ability to import and export data (such as text, graphics or audio/video) from other sources. The Library menu contains commands that duplicate those found along the bottom of each Library window. The Reports menu allows the user to generate a printed report of the entire interface project or selected portions. The Utilities menu gives the user access to the UAN Reference and to the Message window. Finally the Help menu offers access to Help windows for QUANTUM or UAN, and windows giving generic information about QUANTUM and the UAN.

6.6. Other features

Along with the artifacts listed above, the design for QUANTUM includes other features aimed at expanding the usability of the UAN and QUANTUM.

There are two **Help** windows, one for help with using the QUANTUM environment, and help with using the UAN. Both windows have the same layout for consistency. The Help window is split into two fields. The bottom field holds the help text. The top field holds either an Index view of the Help text (listed by keyword), or Table of Contents view (listed by concept). A pair of buttons in the middle of the window, labeled Index and Contents, control the view displayed in the top field. Two different views are used so that users needing help on a particular item can quickly find the appropriate help using an index search, and those users needing more general help can use the table of contents view to show all subjects. This Help window is shown below.

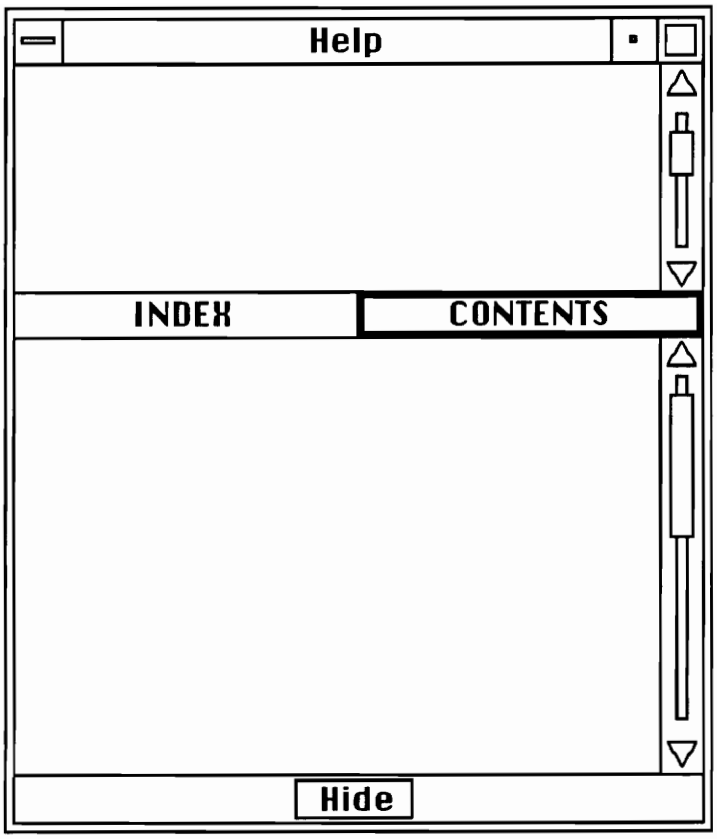


Figure 6.21 Help window

The Message window displays a one-line description of the application of current tool. This feedback will aid new users who will be unfamiliar with QUANTUM usage.

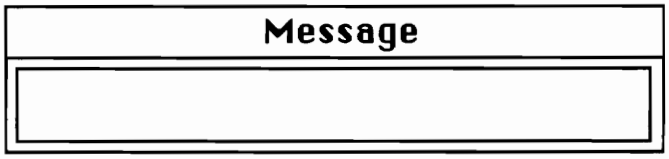


Figure 6.22 Message window

All three palettes (Hierarchy, UAN, and Note), the Message window, and the Menubar float above all other windows such as Task Description windows, Task Library windows, and Note windows. This ensures that they are always visible to the user.

When a tool is chosen from any palette, the cursor changes into the shape of the icon used for the palette button, indicating to the user what tool they are using at any given time. If the user attempts to apply a tool to a window within which its usage is undefined, the system will

emit a beep. When the cursor is within a window where the current tool is not applicable, the cursor changes into the shape pictured below:

Figure 6.23 'Tool not applicable' cursor

The next chapter outlines the expert evaluation process for the QUANTUM interface design.

7. Expert evaluation

While panel walkthroughs and informal interactions led to significant refinement of the QUANTUM user interface design, further evaluation was deemed necessary to ensure usability. Expert evaluation was chosen as the evaluation method, as it offers several advantages:

- Experts bring extensive real-world interface design experience to an evaluation, giving them the ability not only to precisely identify problems but to suggest solutions
- It is usually cheaper to have an interface evaluated by small number of experts than a large number of subjects (who must be compensated)
- Expert evaluations can result in both quantitative and qualitative data, for analysis

The purposes of expert evaluation as used in this research were 1) to include formative evaluation as part of the development process, and 2)

It is noted here that the researchers are not claiming that expert evaluation can be used a substitute for evaluation with end-users, rather that expert evaluation provides the designer with an efficient method of usability assessment which can complement and augment other methods of usability evaluation.

7.1. Method

Three expert evaluators were chosen as subjects to evaluate the QUANTUM design. These experts were human-computer interaction researchers, each with over 4 years of industrial and academic experience in software user interface design. Two of the evaluators were authors of the UAN, and the third expert has performed significant research on the usability of, and extensions to the UAN. Nielsen (Nielsen, 1990) showed that evaluation by 3 subjects with minimal human factors training identifies an average of 49% of all usability problems in an interface. Virzi (Virzi, 1992) later showed that evaluation by 3 users without domain knowledge is likely to uncover 65% of the usability problems in an interface design. These results lead to the assumption that the expert evaluators should be able to identify more than 65% percent of any errors in the interface design, due to their extensive domain and interface design knowledge. Under this assumption, the expert evaluation method becomes an ideal choice in terms of efficiency and accuracy. The expert evaluators were asked to subjectively evaluate QUANTUM on the basis of two separate issues:

1) the usability of the QUANTUM user interface

2) a comparison of the ease of UAN usage among pencil and paper, word processors and other on-line tools, and a UAN support tool such as QUANTUM

The first evaluation issue deals only with usability of the user interface, and does not address the value of a UAN support tool such as QUANTUM, which is the point of the second evaluation issue.

7.1.1. Evaluation sessions

Three separate evaluation sessions were performed using one user per session, using the same version of the QUANTUM prototype. Each evaluation session was videotaped with the prior permission of the subject. The researcher was present during each session to answer questions and give instructions about usage of the prototype. The diagram below illustrates the arrangement of the participants in the evaluation sessions.

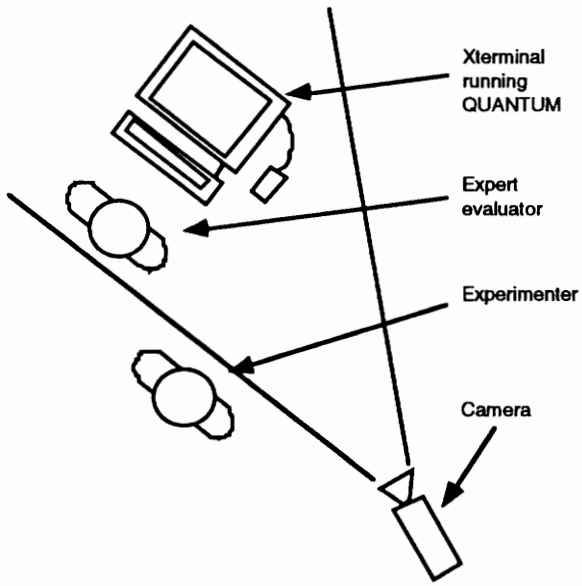


Figure 7.1 Seating arrangement for evaluation session

In each session, the user was presented with the QUANTUM prototype and instructed to evaluate the user interface by performing the general task of constructing a user interface design using QUANTUM. Two of the subjects chose to transcribe prewritten UAN for the user interface of an existing application. The third subject chose to create the UAN for the user interface of an imaginary system. By constructing an entire user interface design, each expert was able to

interact with a majority of the items in the QUANTUM user interface. For each item that was not used during the session, the researcher demonstrated the item and explained its usage. The researcher observed the experts in their use of QUANTUM and noted any critical incidents that occurred. The videotape for each session was later analyzed to identify the steps leading up to the reported critical incidents, and to detect other critical incidents that were missed during the evaluation.

7.1.2. User questionnaires

After each user completed interaction with all of the QUANTUM interface items, a three section questionnaire was administered. Based in part on QUIS (Chin, Diehle, & Norman, 1988), the first section of the questionnaire was designed to estimate the satisfaction of the subjects with the QUANTUM user interface. The second part of the questionnaire asked the subjects directed questions about areas of the QUANTUM user interface in order to elicit general comments. This section allowed the subjects to suggest solutions to problems they might identify.

7.1.3. Expert comparative ranking matrix

The third section of the questionnaire was designed to let the subjects comparatively rank methods for using the UAN. The three methods of choice were (pencil and paper), non-integrated software (word processors), and an integrated, dedicated support tool (QUANTUM). Based on their extensive experience with UAN usage in a variety of methods, subjects were asked to rate each of the above methods for ease of use in the performance of a set of tasks. These tasks were laid out along the vertical axis of a matrix, with the three methods listed across the horizontal axis. A portion of the evaluation matrix is reproduced below.

Task	pencil + paper	word processor	QUANTUM
Write UAN task description for a single, low-level task (example: click on a button)			
Write UAN task description for a single, high-level task (example: create a word processing document)			
Associate the UAN task description for a super-task to that of one of its sub-tasks			

Figure 7.2 Portion of evaluation matrix

For each cell in the matrix, subjects were asked to place a number from 1 to 3 that reflected their preference for that method in the performance of the given task (as compared to the other listed methods). Methods could be assigned equivalent ratings if they were of equal preference to the evaluators.

7.2. Results and discussion

Analysis of the feedback obtained during the evaluations and the questionnaires indicated that although several problems were identified, overall the expert evaluators reacted positively to the user interface of QUANTUM. Answers from the first section of the questionnaire from each expert were averaged, indicating that the experts found the overall interface to be satisfying, easy to use, adequate in capability, and flexible. The comments from the second part of the questionnaire discussed issues such as tool modality, window layout, and feedback; however the majority of the comments indicated a positive reaction. Analysis of the evaluation matrix indicated that for a majority of UAN tasks, QUANTUM is preferable to the use of word processors and to the use of pencil and paper when working with the UAN. Pencil/paper and word-processing methods were on the average equal in preference for use for UAN, but in 90% of the time, QUANTUM is preferable to both methods. Several of the comments from both the evaluations and the questionnaires are briefly discussed below. A detailed summary of all comments from the expert evaluators is included in Appendix C of this thesis.

The evaluators identified several problems in the interface design, most of which related to window management issue. Experts observed that the number of windows displayed at once on the screen can grow to an unmanageable level. The use of floating palettes was found to be annoying and distracting by some evaluators. One expert noted that the placement of the Hide button at the bottom of all windows makes it too easy to hit accidentally, resulting in unexpectedly closing a window. Another problem identified in the evaluations was the modality of the tools. The current tool setting persists until the user chooses another tool. All evaluators noted that there are some times when the user will want to create a new task and begin typing in the UAN immediately, rather than make two different tool choices (new task, and pointer) before typing. When the experts were presented with a dialog box containing Preferences settings, all agreed that the best strategy for the modality issue is to let the user control the setting.

Other suggestions for improvements included the addition of a window tiling command that would automatically align all windows open on the screen. The use of Courier (a monospaced font) for the Task Description Spreadsheets was called into question, and it was suggested that the use of a sans serif font should be investigated. All three experts agreed that Notes should be draggable between windows, and one expert suggested the ability of placing a note within the window of another note. The experts strongly expressed the desire to be able to create aliases of notes in the same way tasks are aliased in the Abstraction Hierarchy window. One expert noted that the import of graphics was a key task overlooked in the layout of the Sketchnote window.

8. Future work

QUANTUM will undergo many revisions as design problems are identified, and extensions added. The goal of this research was not to generate a final, complete design for a UAN tool, but to prove that such a tool is feasible and would have success in solving some of the usability problems reported by UAN users. This chapter outlines suggested future work for the QUANTUM design.

Further evaluation of the QUANTUM user interface should be performed. While the use of frequent walkthroughs and expert evaluation successfully identified several problem in the user interface, a formal experiment using a statistically significant number of subjects should be performed. Due to time constraints, this was not accomplished during this research.

The following list summarizes some of the proposed extensions to the QUANTUM design. Many of these extensions came directly out of the evaluation sessions, others came from suggestions by walkthrough participants, and the remaining suggestions were derived by the researcher.

Collapsing of task structure

The task abstraction structure for an interface can be made up of tens or hundreds of task descriptions. Representation of this graph within the Abstraction Hierarchy window solves some of the navigation problems, but such a tree graph increases in complexity at a geometric rate. The ability to 'collapse' the tree in sections would make the Abstraction Hierarchy window easier to use.

Automatic consistency checking

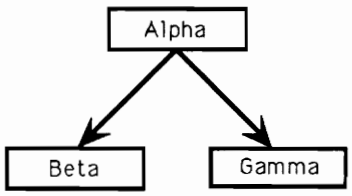


Figure 8.1 Dependency example

The above structure indicates that within the task description of task Alpha, calls to tasks Beta and Gamma are included. Currently it is left up to the user to make sure that those calls are actually written within the task description of Alpha. The problem of

making sure that those calls are included is not solved by simply inserting appropriate calls within the description of Alpha, since it cannot be determined where to place the calls. Techniques to ensure this kind of consistency would aid the interface designer greatly.

Version history

The iterative refinement of an interface design generates many new interface versions. A system for tracking the different versions of an interface design would allow designers to review descriptions of earlier version in order to gauge the evolution of a design.

Documentation generation

Industrial sites using the UAN reported its use for generation of user documentation. A traversal of the task abstraction structure naturally creates a documentation structure that is built from a user task view. Towe (Towe, 1992) investigated the overall applicability of UAN to the documentation process. QUANTUM could automatically generate this documentation structure for use by documentation specialists.

UAN generation by demonstration

Myers (Myers, 1988) and others (Myers & Rosson, 1992) have reported work on the creation of user interfaces by demonstration. Since the UAN is a representation of user action and interface feedback, it should be possible to generate UAN from the analysis of a stream of user action. Siochi (Siochi & Ehrich, 1989) investigated the use of user transcripts for usability evaluation. The same techniques used to capture the user transcripts in those experiments could be used by QUANTUM to automatically generate low-level task descriptions for existing applications.

Multiple platform availability

In its current state, QUANTUM is designed around an XWindows platform. Given the use of many different platforms (Macintosh, DOS, Windows, VAX, etc.) it is reasonable to assume that interface designers will want QUANTUM to be implemented across several platforms.

MRP analysis

Siochi (Siochi & Ehrich, 1989) (Siochi & Hix, 1991) applied the MRP (Maximal Repeating Pattern) analysis method to user interfaces in order to identify possible usability problems. An extension to QUANTUM could automatically perform this analysis and identify portions of a design that could have problems.

'Live' playback of UAN

If the UAN can be thought of as an recording of the user-computer interaction in an interface, then it should be possible to 'play' this recording. Using the UAN in the task descriptions, QUANTUM could use animation of a screen sketch to show the designer what an interaction would look like to the user.

Embedded UAN and QUANTUM tutorials

Bhattarai (Bhattarai, 1992) designed a UAN Tutor aimed at presenting a workbook approach to learning the UAN. This tutorial could be embedded within the tool, making QUANTUM even easier to use by UAN novices. A similar tutorial for QUANTUM could be provided.

Interchange and cooperation with other UIMS

Many UIMS (Myers, 1989) and usability evaluation tools (Weiler, et al., 1993) exist to support user interface designers and evaluators. As the UAN is a powerful tool for representing behavioral designs, QUANTUM should be able to import and export data to and from as many of these systems as possible, making QUANTUM an integral part of the design process

Interface code generation

Low-level UAN task descriptions map to 'sentences' of user action. These descriptions precisely capture the behavior and state of the interface in response to user actions. By linking to predefined libraries of interface code (code to create and manage window, widgets, etc.), QUANTUM could produce the low-level code necessary to implement the described interface. This possible extension would enable QUANTUM to bridge the gap between the behavioral and constructional process domains.

Intelligent on-line agent

An on-line agent could aid the user interface designer by providing helpful UAN usage suggestions, confirmation of intent of actions, and consistency checks. This feature could be turned off as necessary.

9. Summary and conclusion

UAN users have reported difficulty with usage of the notation. An analysis of the usage problems reported led to the hypothesis that the usability problems are not inherent in the UAN, but that manual manipulation as a method for working with the UAN is inherently difficult., leading to dissatisfaction on the part of UAN users.

The researcher proposes that a dedicated software support tool will solve some of the identified problems. The researcher produced a design for such a tool (QUANTUM) as an answer to this problem. Interviews with UAN users produced a list of goals for QUANTUM, which drove the design of the user interface. The interface design for QUANTUM was produced by iterative refinement through the use of design walkthroughs by interface design experts and interaction with a prototype.

Three experts in UAN and user interface design were selected to perform an expert evaluation of QUANTUM on the basis of two issues: 1) an evaluation of the QUANTUM user interface, and 2) an evaluation of the usability of the tool as compared to manual methods (pencil and paper) and non-integrated software tools (word processors). The expert evaluators indicated that although there were minor problems regarding tool usage and window management, the QUANTUM user interface offers more than adequate usability. Expert evaluators also indicated a preference for QUANTUM over manual and non-integrated methods for UAN usage.

The overall goal of this research was the investigation of whether or not a tool dedicated to the UAN would solve some of the problems reported by UAN users. QUANTUM addresses those problems and offers significant advantages over manual or non-integrated software methods for using UAN. QUANTUM improves the usability of the UAN by making it easier to use.

References

Alexander, H. (1987). *Formally-based Tools and Techniques for Human-Computer Dialogues.* New York: John Wiley & Sons, Inc.

Apple Computer Inc. (1987). *Human Interface Guidelines: The Apple Desktop Interface.* Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Ashlund, S. L. (1991). *IDEAL: A Tool to Enable Usability Specification and Evaluation.* Thesis Report, Virginia Polytechnic Institute and State University.

Bhattarai, H. R. (1992). *UAN (User Action Notation) Tutor.* Project Report, Virginia Polytechnic Institute and State University.

Boehm, B. W. (1988). A Spiral Model of Software Development. *IEEE Computer, 21*(5), 61-72.

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

Chin, J. P., Diehle, V. A., & Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. In E. Soloway, D. Frye, & S. B. Sheppard (Eds.), *Human Factors in Computing Systems, CHI '88 Conference,* (pp. 231-218). Washington, D.C., May 15-19: ACM.

Foley, J. D. (1980). The Keystroke-Level Model for Performance Time with Interactive Systems. *Communications of the ACM, 23*(7), 396-409.

Gould, J. D., Boies, S. J., & Lewis, C. (1991). Making Usable, Useful, Productivity-Enhancing Computer Applications. *Communications of the ACM, 34*(1), 75-85.

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

Hartson, H. R. (1989). User-Interface Management Control and Communication. *IEEE Software*(1), 62-70.

Hartson, H. R. (1991). Temporal Aspects of Tasks in the User Action Notation. *Human-Computer Interaction, 7*, 1-45.

Hartson, H. R., Brandenburg, J., & 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 and Bartlett Publishers.

Hartson, H. R., & Hix, D. (1989). Human-Computer Interface Development: Concepts and Systems for its Management. *ACM Computing Surveys, 21*(1), 6-92.

Hartson, H. R., Hix, D., & Siochi, A. (1989). *Notes for a UAN tool*

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

Hartson, H. R., & Smith, E. J. (1991). Rapid Prototyping in Human-Computer Interface Development. *Interacting with Computers, 3*(1), 51-91.

Hemenway, K. (1982). Psychological Issues in the Use of Icons in Command Menus. In *Human Factors in Computing Systems, CHI '82 Conference,* (pp. 20-23). Gaithersburg, Maryland, March 15-17: ACM.

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

Hoare, C. A. R. (1985). *Communicating Sequential Processes*. New Jersey: Prentice-Hall International.

Johnson, P., Wilson, S., Markopoulos, P., & Pycock, J. (1993). ADEPT: Advanced Design Environment for Prototyping with Task Models. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, & T. White (Eds.), *Human Factors in Computing Systems, INTERCHI '93 Conference,* (pp. 56). Amsterdam, The Netherlands, April 24-29: Addison-Wesley Publishing Company.

Karat, C.-M. K., Campbell, R., & Fiegel, T. (1992). Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), *Human Factors in Computing Systems, CHI '92 Conference,* (pp. 397-404). Monterey, California, May 3-7: ACM.

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

Lewis, C., Polson, P., Wharton, C., & Rieman, J. (1990). Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces. In J. C. Chew & J. Whiteside (Eds.), *Human Factors in Computing Systems, CHI '90 Conference,* (pp. 235-242). Seattle, Washington, April 1-5: ACM.

Macleod, M., & Bevan, N. (1993). MUSiC Video Analysis and Context Tools for Usability Measurement. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, & T. White (Eds.), *Human Factors in Computing Systems, INTERCHI '93 Conference,* (pp. 55). Amsterdam, The Netherlands, April 24-29: Addison-Wesley Publishing Company.

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,* 3-50.

Myers, B. A. (1988). *Creating User Interfaces by Demonstration.* Boston: Academic Press Incorporated.

Myers, B. A. (1989). User-Interface Tools: Introduction and Survey. *IEEE Software, 6*(January), 15-23.

Myers, B. A., & Rosson, M. B. (1992). Survey on User Interface Programming. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), *CHI' 92 Conference on Human Factors in Computing Systems,* 1 (pp. 195-202). Monterey, CA: ACM Press.

Nielsen, J. (1990). Evaluating the Thinking-Aloud Technique for Use by Computer Scientists. In H. R. Hartson (Ed.), *Advances in Human-Computer Interaction* (pp. 197-216). Norwood, New Jersey: Ablex Publishing Corporation.

Norman, D. A., & Draper, S. W. (Ed.). (1986). *User Centered System Design : New Perspectives on* (xiii ed.). Hillsdale, N.J.: Lawrence Erlbaun Associates.

Norman, K. L. (1991). *The Psychology of Menu Selection : Designing Cognitive Control at the Human/Computer Interface* (1 ed.). Norwood, New Jersey: Ablex Publishing Corporation.

Open Software Foundation (1989). *OSF/Motif Style Guide*. Cambridge, Massachusetts: Open Software Foundation.

Ousterhout, J. K. (to be published 1993). *An Introduction to Tcl and Tk* (draft ed.). Reading, Massachusetts: Addison-Wesley Publishing Company, Inc.

Payne, S. J. (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*(2), 229-240.

Rowley, D. E., & Rhoades, D. G. (1992). The Cognitive Jogthrough: A Fast-Paced User Interface Evaluation Procedure. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), *Human Factors in Computing Systems, CHI '92 Conference*, (pp. 389-396). Monterey, California, May 3-7: ACM.

Shneiderman, B. (1987). *Designing the User Interface: Strategies for Effective Human-Computer Interaction* (First ed.). Reading, Mass: Addison-Wesley Publishing Company.

Siochi, A., & Ehrich, R. (1989). *Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions* (Technical Report No. 90-15). Computer Science Department, Virginia Polytechnic Institute and State University.

Siochi, A., & Hix, D. (1991). A Study of Computer-Supported User Interface Evaluation Using Maximal Repeating Pattern Analysis. In S. P. Robertson, G. M. Olsen, & J. S. Olsen (Eds.), *Human Factors in Computing Systems, CHI '91 Conference*, (pp. 301-305). New Orleans, Lousiana, April 27-May 2: ACM.

Sunrise Software International Incorporated (1992). ezX. Middletown, RI: Sunrise Software International Incorporated.

Towe, J. B. (1992). *Integration and Iteration of Documentation and Interactive Systems Development via the User Action Notation (UAN)*. Master of Science, Computer Science Department, Virginia Polytechnic Institute and State University.

Virzi, R. A. (1992). Refining the Test Phase of Usability Evaluation: How Many Subjects is Enough? *Human Factors*, *34*(4), 457-468.

Wasserman, A. I., & Shewmake, D. T. (1985). Advances in Human-Computer Interaction. In H. R. Hartson (Ed.), *Advances in Human-Computer Interaction* (pp. 191-210). Norwood, New Jersey: Ablex Publishing Corporation.

Weiler, P., Cordes, R., Hammontree, M., Hoiem, D., & Thompson, M. (1993). Software for the Usability Lab: A Sampling of Current Tools. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, & T. White (Eds.), *Human Factors in Computing Systems, INTERCHI '93 Conference,* (pp. 57-60). Amsterdam, The Netherlands, April 24-29: Addison-Wesley Publishing Company.

Appendix A: QUANTUM sample windows

This appendix contains Postscript pictures of the following QUANTUM windows: Abstraction Hierarchy, Task Description, Task Library, Hierarchy palette, UAN palette, Note palette, Menubar, and Message window.

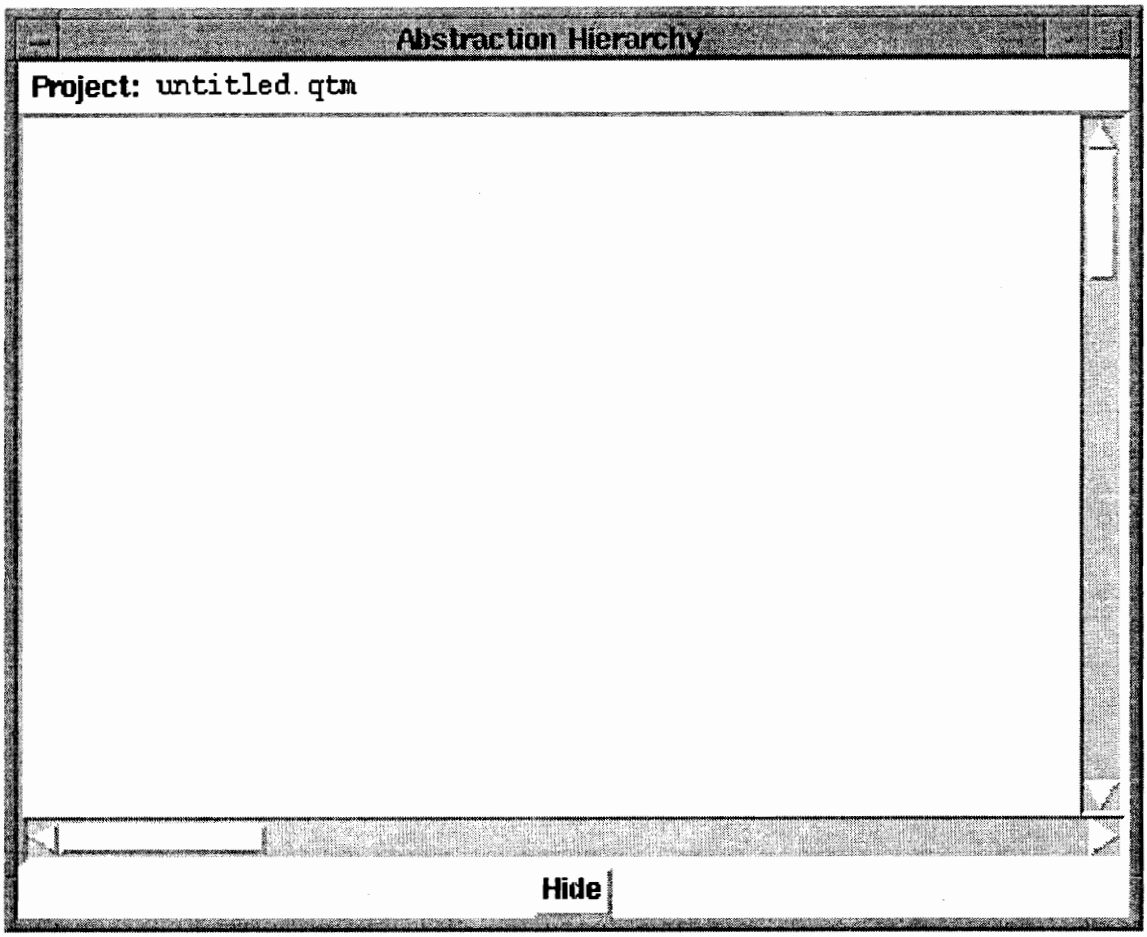


Figure A.1 Abstraction Hierarchy window

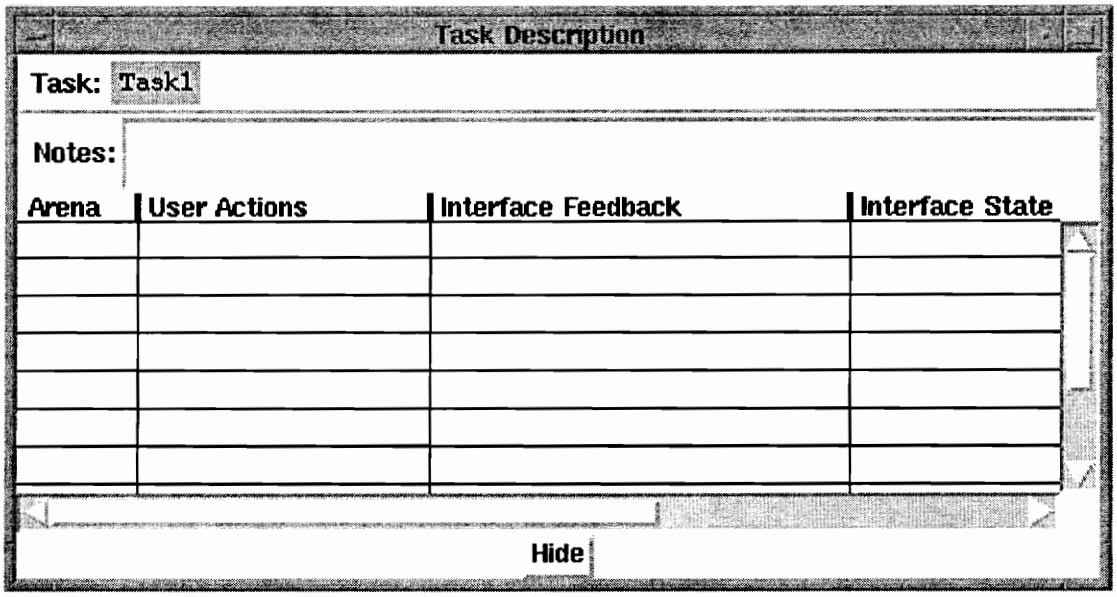


Figure A.2 Task Description window

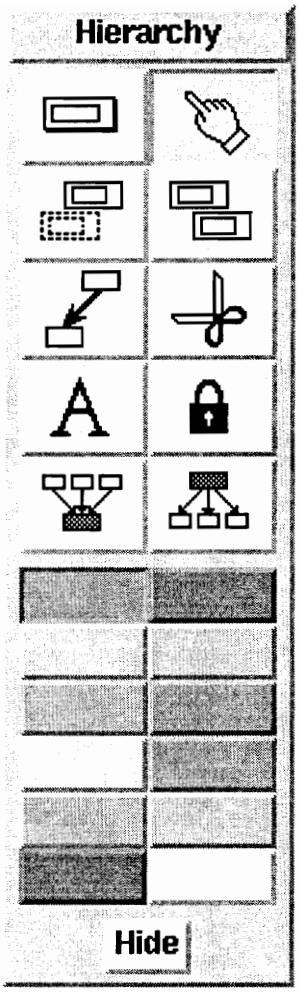


Figure A.3 Hierarchy palette

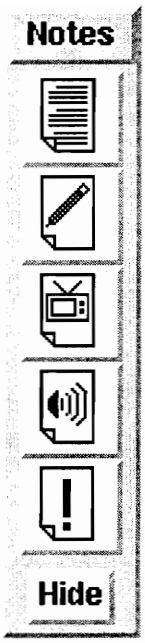


Figure A.4 Note palette

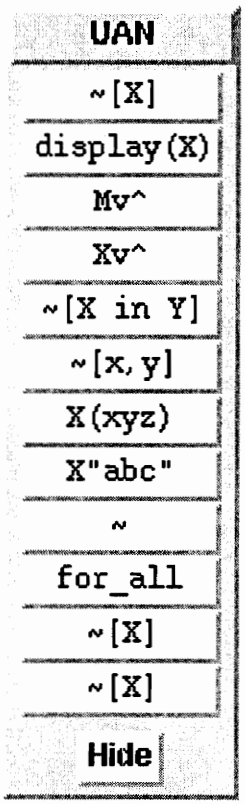


Figure A.5 UAN palette

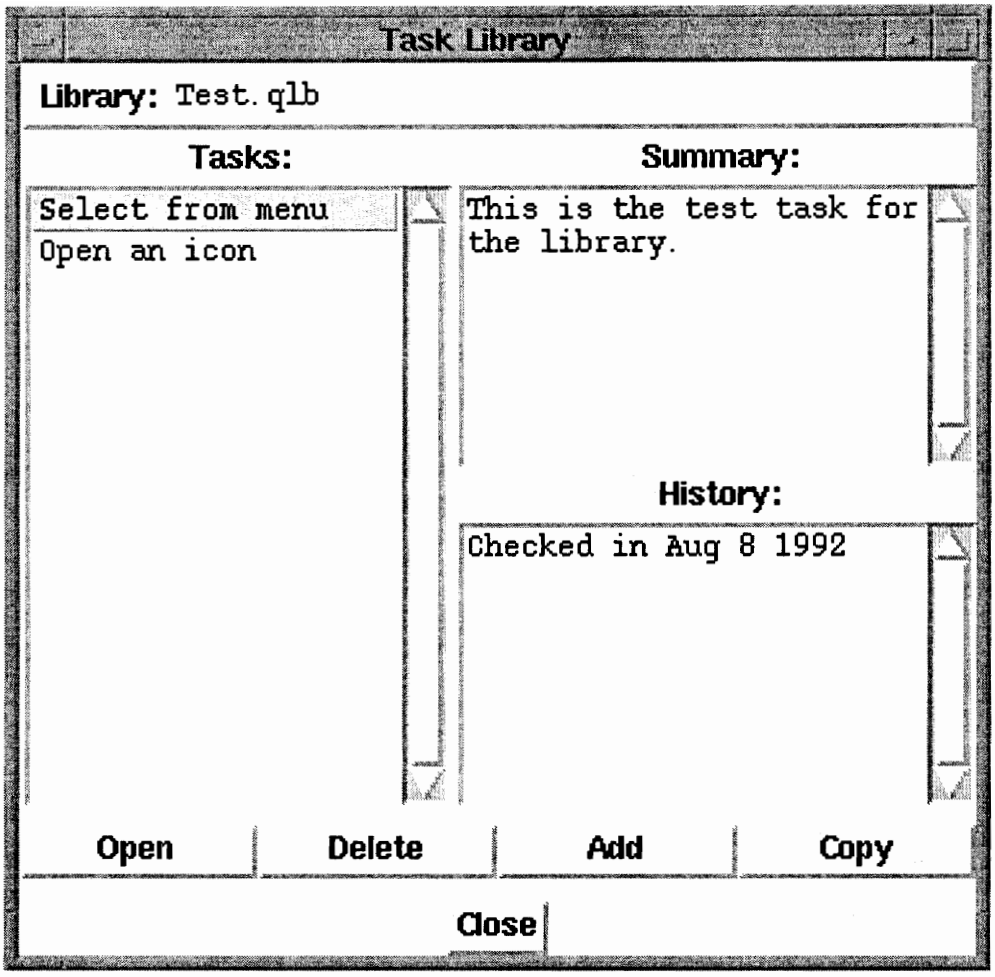


Figure A.6 Task Library window

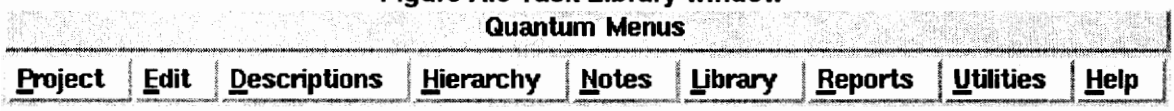


Figure A.7 Menubar

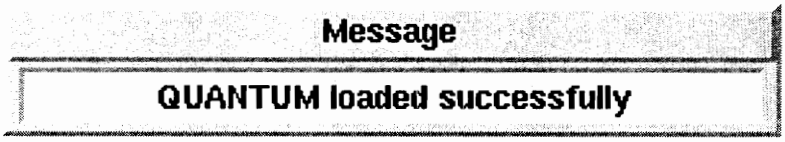


Figure A.8 Message window

Appendix B: Expert evaluation questionnaire

Evaluation Questionnaire for QUANTUM Prototype

Identification number : _____
QUANTUM Prototype Version : _____

This questionnaire is intended to supplement your videotaped record of your evaluation session. You are entitled to a copy of all data collected during your session, including the videotape record and experimenter's notes.

Your participation in this evaluation is voluntary. You may withdraw at anytime from this evaluation without penalty of any form. Thank you for your help.

Arcel Castillo
231-6470/382-5850

Section 1. Reaction to using QUANTUM

For each statement, please circle the number on each of the given scales that makes the statement appropriately express your overall reaction. After each statement, there is space for your written comments. Use extra paper if necessary. If you cannot comment on a statement, circle N/A. Please note that the scales may be slightly different for each question.

1. Creation and manipulation of tasks within the Abstraction Hierarchy is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

2. Entering and editing UAN task descriptions within Task Description windows is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

3. Storing and retrieving tasks in a Task Library is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

4. Creating and attaching a Text Note is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

5. Creating and attaching a Sketch Note is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

6. Creating and attaching an Issue Note is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

7. Use of the tools from the separate palettes (Hierarchy, Note, UAN) is/has:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult	-3	-2	-1	0	+1	+2	+3	NA	easy
inadequate capability	-3	-2	-1	0	+1	+2	+3	NA	adequate capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

8. Using the UAN Palette to insert UAN strings in Task Descriptions is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

9. Using QUANTUM Help and UAN Help is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

10. Interpreting the icons used is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to interpret	-3	-2	-1	0	+1	+2	+3	NA	easy to interpret
inadequate in conveyance	-3	-2	-1	0	+1	+2	+3	NA	adequate in conveyance
inconsistent	-3	-2	-1	0	+1	+2	+3	NA	consistent

Please explain:

11. Predicting the results of actions (ex: tools chosen from menus or palettes) is/has:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to predict	-3	-2	-1	0	+1	+2	+3	NA	easy to predict
inadequate effect in result	-3	-2	-1	0	+1	+2	+3	NA	adequate effect in result
inconsistent	-3	-2	-1	0	+1	+2	+3	NA	consistent

Please explain:

12. Tracing a task execution path using QUANTUM is:

frustrating	-3	-2	-1	0	+1	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	+2	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	+2	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	+2	+3	NA	flexible

Please explain:

Section 2: Comments about QUANTUM

Please list any comments you have about the system according to the following questions. Use extra paper if necessary.

Did you have any problems with navigation among the different windows?

What is your opinion on the window management in QUANTUM (showing/hiding windows, etc.)?

Was the layout and format of windows consistent?
If not, why?

Did you find it hard to learn how to operate/use the system?
If so, what was difficult?

Were there any tasks that you wanted to be able to perform but could not within QUANTUM? Please consider the limitations of the prototype in your answer.

Do you have any suggested modifications to the QUANTUM interface design?

Do you have any suggested extensions to QUANTUM (as a tool environment)?

Do you have any other comments?

Section 3: Comparison to other methods

In this section, you are asked to rank the relative ease of performance of several tasks using the UAN. Each task can be performed using three different vehicles: pencil and paper, word processor, and QUANTUM. Based on your experience with the UAN, please estimate the utility of each vehicle for each task, and rank the vehicles against each other in preferred usage for that task.

Use the following scale:

1 **2** **3**

least preferred **most preferred**

For example, if for some example task you prefer a word processor to QUANTUM, and QUANTUM to pencil and paper, you would fill in the table like this:

Task	pencil + paper	word processor	QUANTUM
example task	1	3	2

In cases where vehicles are equally preferred, use the same rank level, like this:

Task	pencil + paper	word processor	QUANTUM
example task	1	2	2

If you cannot assign a rank to a vehicle, write "NC" (No Comment) in that table cell.

Task	pencil + paper	word processor	QUANTUM
Write UAN task description for a single, low-level task (example: click on a button)			
Write UAN task description for a single, high-level task (example: create a word processing document)			
Associate the UAN task description for a super-task to that of one of its sub-tasks			
Navigate the task abstraction structure for an interface design described in UAN			
Modify the task abstraction structure for an interface design described in UAN			

Attach and edit a textual annotation to a UAN task description			
Attach and edit a screen sketch to a UAN task description			
Attach and edit a state transition diagram to a UAN task description			
Highlight a portion of an interface design that needs further evaluation or attention			
Associate video or audio clips to a particular portion of an interface design			
Trace an task execution path through an interface design described in UAN			
Reproduce an interface design written in UAN for a colleague			
Conduct a group walkthrough of an interface design written in UAN			
Work with a group of designers on the same interface design using UAN			
Ensure the adherence to a set of interface guidelines across several interface designs written in UAN			
Update several existing interface designs written in UAN as the result of an interface guideline change			
Find the places in an interface design where a particular task is used.			
Look up the current UAN notation for a temporal relation			
Build an archive of related UAN task descriptions for later use in other interface designs			

Appendix C: Summary of evaluation results

This appendix summarizes the results of the expert evaluations, and is divided into three sections. Comments made during the evaluation sessions are summarized in the **Evaluation Comments** section. Comments and responses made on the questionnaires are summarized in the **Questionnaire Results** section. Results of the ranking matrix are presented in the **Matrix Results** section.

Evaluation Comments

evaluator 1

- User wanted to be able to place a note within another note
- User expressed need for note aliases
- Wanted to import a screen sketch/dump in a Sketchnote
- "Can I go from one note to another directly?"
- User wanted to place note directly on task icon
- Intent of Arena field (hold Note icons) was unclear
- If a task B is referred to in the Task Description of a task A, user wants to be able to go directly to the Task Description window for task B by double-clicking on the name
- Insertion cursor blink rate is too low - easy to lose
- When moving a task icon within the Abstraction Hierarchy, the window should automatically scroll.
- All Hide/Close buttons should be the same size
- While renaming a task was consistent and easy, user should not have to hit return to complete renaming
- Long Note names and task icon names are hard to read and take up too much space - can two lines be used?
- Fan-in and fan-out views for a task should list ALL linked tasks, not just tasks linked to the highlighted task icon
- User wanted to show parameters as part of a task link

- Scrollbars on a window should show space used, not total space available
- "Good cursor feedback – shape indicates task"
- When a note or task is created, tool should change automatically to selector tool (pointer) so that item can be opened or moved immediately
- Meanings of icons is relatively apparent
- The ability to color-code icons was largely unused
- Abstraction Hierarchy window should be expanded to its maximum size at startup
- Group selection of notes or icons would make their rearrangement easier
- Notebar looks like other test-entry fields – unclear that this is intended for Note icons
- The use of several 'modes' (one tool to create a note, another tool to open it, etc.) and the use of three palettes can be frustrating and confusing. the use of cursor shape to indicate mode helps. the use of a Preferences dialog also helps.
- Buttons on UAN Palette should encode 'macros'
- New notes should be named Textnote1, Sketchnote1, for further feedback on note type - thisaids recognition, and may save user from having to rename them
- Note icons are inconsistent – while Textnote icon shows text (product), Sketchnote icon shows pencil (tool); should use shapes in icon to convey appearance of graphics
- Use of outline in Help windows was good – easy navigation
- Staggered placement of Task Description windows aids in task tracing

evaluator 2

- When a task is created, it should be ready to rename
- The use of straight arrows to represent task links is not optimal – a combination of straight and vertical lines would results in a more pleasing representation
- User wanted placed task icons to 'snap' to a grid, providing automatic formatting of the task hierarchy.
- A Preferences option could allow the user to choose between 'snap-to-grid' functionality or 'free-form' placement
- "I want to iconify a palette."
- Need support for customizing UAN symbology as listed in UAN Reference

- User wants to be able to assign colors to levels in a task hierarchy
- The placement of 'Delete task' on the 'Hierarchy' menu is not optimal. the 'Cut' option on the 'Edit' menu should serve this function
- User spends most of the time arranging task structure
- A choice of line types (straight or diagonal)
- The icon used for the renaming tool (letter 'A') was confusing – user thought it was a text tool that allowed text to be placed anywhere in Hierarchy, or enter UAN in Task Description windows
- User always wanted tool to switch to pointer after placing a new Note or task, or after using UAN palette
- Although tool has design problems, it is still better than manual usage
- User wanted to alias a note
- User referred to a note from within another note

evaluator 3

- UAN Palette should be renamed 'UAN Idioms'?
- Ordering of option in Preference dialog is unclear
- The wording 'raise' and 'lower' is confusing – use 'show' and 'hide'
- Last three buttons in UAN Palette should be read 'untitled' to indicate to the user that they are customizable
- The notion of a task alias was confusing
- The icon used for task links was unclear – not sure where hot spot is
- User expected link arrows to form from the edge of a task icon, and not from center
- Straight-line arrows would clean up link representation
- User wanted to be able to delete task icons with scissors
- User liked the use of 'sticky' modes – it enabled user to create several tasks at once without having to change tools
- Color was not useful

- The one-line message displayed in the Message window should change according to what item is underneath the cursor/pointer – this would allow the user to get quick 'help' on what each tool or item can do, or is used for
- Renaming a note should have same behavior as renaming a task
- The use of a hand in the note placement cursor was confusing
- The user should be able to type text in Notebar as well as place Notes
- Making use of the three buttons available on most mice would allow several different types of selection (perhaps one for renaming, one for placement, one for opening)
- Buttons in UAN list should be scrollable, reducing screen real estate and possibly offering unlimited choices
- User wants to be able to cut out parts of a Task Description for check-in into a library, or for creation of a new task

Questionnaire Results

For each question, the numbers chosen by the evaluators for each spectra are shown in boldface type.

1. Creation and manipulation of tasks within the Abstraction Hierarchy is:

frustrating	-3	-2	-1	**0**	**+1**	**+2**	+3	NA	satisfying
difficult to use	-3	-2	-1	**0**	+1	**+2**	**+3**	NA	easy to use
inadequate in capability	-3	-2	-1	0	**+1**	**+2**	+3	NA	adequate in capability
rigid	-3	-2	-1	**0**	+1	**+2**	+3	NA	flexible

2. Entering and editing UAN task descriptions within Task Description windows is:

frustrating	-3	-2	-1	**0**	+1	**+2**	+3	NA	satisfying
difficult to use	-3	-2	-1	0	**+1**	**+2**	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	+1	**+2**	+3	NA	adequate in capability
rigid	-3	-2	-1	0	+1	**+2**	+3	NA	flexible

3. Storing and retrieving tasks in a Task Library is:

frustrating	-3	-2	-1	**0**	**+1**	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	**+1**	**+2**	+3	NA	adequate in capability
rigid	-3	-2	-1	0	**+1**	**+2**	+3	NA	flexible

4. Creating and attaching a Text Note is:

frustrating	-3	-2	-1	**0**	**+1**	+2	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	+3	NA	easy to use
inadequate in capability	-3	-2	-1	0	**+1**	**+2**	+3	NA	adequate in capability
rigid	-3	-2	**-1**	0	+1	**+2**	**+3**	NA	flexible

5. Creating and attaching a Sketch Note is:

frustrating	-3	-2	-1	**0**	**+1**	+2	+3	**NA**	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	+3	**NA**	easy to use
inadequate in capability	-3	-2	-1	0	**+1**	**+2**	+3	**NA**	adequate in capability
rigid	-3	-2	**-1**	0	+1	+2	**+3**	**NA**	flexible

6. Creating and attaching an Issue Note is:

frustrating	-3	-2	-1	**0**	**+1**	+2	+3	**NA**	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	+3	**NA**	easy to use
inadequate in capability	-3	-2	-1	0	**+1**	**+2**	+3	**NA**	adequate in capability
rigid	-3	-2	**-1**	0	+1	+2	**+3**	**NA**	flexible

7. Use of the tools from the separate palettes (Hierarchy, Note, UAN) is/has:

frustrating	-3	-2	**-1**	**0**	+1	+2	+3	NA	satisfying
difficult	-3	-2	-1	0	**+1**	**+2**	+3	NA	easy
inadequate capability	-3	-2	-1	0	**+1**	**+2**	**+3**	NA	adequate capability
rigid	-3	-2	**-1**	0	+1	**+2**	+3	NA	flexible

8. Using the UAN Palette to insert UAN strings in Task Descriptions is:

frustrating	-3	-2	-1	**0**	**+1**	**+2**	+3	NA	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	**+3**	NA	easy to use
inadequate in capability	-3	**-2**	-1	0	**+1**	**+2**	+3	NA	adequate in capability
rigid	-3	-2	**-1**	0	**+1**	**+2**	+3	NA	flexible

9. Using QUANTUM Help and UAN Help is:

frustrating	-3	-2	-1	0	**+1**	+2	+3	**NA**	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	**+3**	**NA**	easy to use
inadequate in capability	-3	-2	-1	0	+1	**+2**	+3	**NA**	adequate in capability
rigid	-3	-2	-1	**0**	**+1**	+2	+3	**NA**	flexible

10. Interpreting the icons used is:

frustrating	-3	-2	-1	**0**	+1	**+2**	+3	NA	satisfying
difficult to interpret	-3	-2	-1	**0**	+1	+2	**+3**	NA	easy to interpret
inadequate in conveyance	-3	-2	-1	0	**+1**	**+2**	+3	NA	adequate in conveyance
inconsistent	-3	-2	**-1**	0	+1	**+2**	+3	NA	consistent

11. Predicting the results of actions (ex: tools chosen from menus or palettes) is/has:

frustrating	-3	-2	-1	**0**	+1	**+2**	+3	NA	satisfying
difficult to predict	-3	-2	**-1**	0	**+1**	**+2**	+3	NA	easy to predict
inadequate effect in result	-3	-2	-1	0	+1	**+2**	**+3**	NA	adequate effect in result
inconsistent	-3	-2	-1	0	+1	**+2**	+3	NA	consistent

12. Tracing a task execution path using QUANTUM is:

frustrating	-3	-2	-1	0	**+1**	+2	+3	**NA**	satisfying
difficult to use	-3	-2	-1	0	+1	**+2**	+3	**NA**	easy to use
inadequate in capability	-3	-2	-1	0	**+1**	**+2**	+3	**NA**	adequate in capability
rigid	-3	-2	-1	0	**+1**	**+2**	+3	**NA**	flexible

Did you have any problems with navigation among the different windows?

No comments were made.

What is your opinion on the window management in QUANTUM (showing/hiding windows, etc.)?

Control of a small number of windows was not difficult, although control of a larger number may prove problematic.

Was the layout and format of windows consistent? If not, why?

Consistency was not reported as a problem, although the layout and floating behavior of palettes was found difficult by some evaluators.,

Did you find it hard to learn how to operate/use the system? If so, what was difficult?

No problems were reported.

Were there any tasks that you wanted to be able to perform but could not within QUANTUM? Please consider the limitations of the prototype in your answer.

Automatic 'prettying' (layout of tree structure) would have enabled production of cleaner task structures.

Do you have any suggested modifications to the QUANTUM interface design?

Studies of action-object vs. object-action rules of modality might answer some questions as to whether or not to behavior of tools as implemented in QUANTUM is optimal.

Do you have any suggested extensions to QUANTUM (as a tool environment)?

More automation of association between task structure in Abstraction Hierarchy with Task Descriptions.

Do you have any other comments?

QUANTUM is "much nicer than pencil and paper". It affords more organizational power over pieces in a user interface design.

The dynamic treatment of curson icons does a good job of indicating current modes, and where tool as applicable.

Matrix Results

For each of the evaluators' responses to ranking each vehicle, the response for a cell is shown in the corresponding cell in the matrix below. Each evaluator's responses is shown on a separate line in each cell. The first line corresponds to evaluator 3's responses, the second line corresponds to evaluator 2's responses, and the third line corresponds to evaluator 1's responses. In nearly all cases, QUANTUM was the most favored vehicle for most tasks.

Task	pencil + paper	word processor	QUANTUM
Write UAN task description for a single, low-level task (example: click on a button)	2 1 1	1 2 2	3 3 3
Write UAN task description for a single, high-level task (example: create a word processing document)	1 1 1	2 2 2	3 3 3
Associate the UAN task description for a super-task to that of one of its sub-tasks	1 2 2	1 1 2	3 3 2
Navigate the task abstraction structure for an interface design described in UAN	1 2 1	1 1 1	3 3 3
Modify the task abstraction structure for an interface design described in UAN	1 2 1	2 1 1	3 3 3

Attach and edit a textual annotation to a UAN task description	1 3 1	2 1 2	3 2 3
Attach and edit a screen sketch to a UAN task description	2 3 2	N/A 1 2	3 2 1
Attach and edit a state transition diagram to a UAN task description	2 2 1	N/A 1 2	3 3 3
Highlight a portion of an interface design that needs further evaluation or attention	? 1 1	? 2 2	? 3 2
Associate video or audio clips to a particular portion of an interface design	N/A 1 N/A	N/A 1 N/A	3 3 3
Trace an task execution path through an interface design described in UAN	1 2 1	1 1 1	3 3 3
Reproduce an interface design written in UAN for a colleague	1 1 1	2 2 2	3 3 2
Conduct a group walkthrough of an interface design written in UAN	2 1 1	1 2 2	3 3 3
Work with a group of designers on the same interface design using UAN	1 1 1	1 2 2	3 3 3
Ensure the adherence to a set of interface guidelines across several interface designs written in UAN	1 1 1	2 2 1	3 3 2
Update several existing interface designs written in UAN as the result of an interface guideline change	1 1 1	2 2 1	3 3 3
Find the places in an interface design where a particular task is used.	1 1 1	2 2 2	3 3 3
Look up the current UAN notation for a temporal relation	N/A 1 1	N/A 1 2	3 3 1
Build an archive of related UAN task descriptions for later use in other interface designs	N/A 1 1	1 2 1	3 3 3

Vita

Arcel Castillo was born to Armando and Cecille Castillo on January 17, 1968 in Providence, Rhode Island. However, he refers to Virginia Beach, Virginia as his hometown. His first exposure to computers was in elementary school as a part of a gifted and talented student program. He graduated with honors from Floyd E. Kellam High School in Virginia Beach, Virginia in May 1986. He then entered the Computer Science program at Virginia Polytechnic Institute and State University (Virginia Tech) in Blacksburg, Virginia. He graduated in May 1990 with a Bachelor of Science in Computer Science, and Minors in Mathematics and Psychology. He continued on in pursuit of a Master of Science degree in Computer Science, also at Virginia Tech. Upon completion of this degree, he plans to continue research in human-computer interaction. He plans to eventually pursue a Doctor of Philosophy in Computer Science, focusing on human-computer interaction research.