Integration and Iteration of Documentation and Interactive Systems Development via the User Action Notation (UAN)

by

James Barry Towe

Project and Report submitted to the faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF INFORMATION SYSTEMS
in
Computer Science

APPROVED:

Dr. Sallie Henry
Co-chair

Dr. Deborah Hix
Co-chair

Dr. Rex Hartson

December, 1992
Blacksburg Virginia

Abstract.

Deficiencies in user documentation leave users frustrated, annoyed and alienated. Legalistic, system oriented texts are not well suited for active users. The primary goals of documentation are education, motivation and communication. An extensive survey of current literature reveals four interrelated keys to good documentation. Specifically these are:

- style/organization
- audience appropriateness
- usability
- learnability.

Users are active, documentation should not hinder nor inhibit that activity. To accomplish this, documentation needs to be user focused such as in the task oriented or minimalist styles. Documentation, in this vein, is an integral part of the system. To improve usability generally requires an iterative development structure. However, integration and iteration are rather difficult due to the lack of a common frame of reference. The literature suggests a series of guidelines or strategies to facilitate document development. These strategies may be utilized with the User Action Notation to generate task oriented skeletal documentation from UAN descriptions. This provides a common frame of reference, improved user documentation and enhanced integrated development for the whole design effort.

Acknowledgments:

First and foremost I would like to thank my parents, Mary and Franklin Towe. They taught me the value of the past and the importance of the future. The acts of my life can never begin to equal their accomplishments. I can only hope to build upon their example.

I would also like to thank Dr. Sallie Henry for her patience, encouragement, insight and teaching that not only made this paper possible, but greatly improved its readability.

Thanks also go to Dr. Deborah Hix and Dr. Rex Hartson. Their innovation, creativity and instruction are the cornerstone upon which this work rests.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

There was nobody around to help me last night so I had to use the manual [ORAA86].

1.0 The Problem: Deficiencies in User Documentation

The above remark exemplifies all too often the user's dilemma in the real world of deadlines, errors and schedules. In the last decade, great strides have been made in improving the usability of interactive software systems. However, as Human-Computer Interaction evolves from a buzzword to a science, little progress is being made in the field regarding improved user documentation. Advances in this area are primarily relegated to technical writers and cognitive psychologists. This creates a gap in communication among system developers and system documentors. This gap leads to the preservation of comments like the introductory remark. Complicating the problem is the fact that user manuals are complex entities that take from 10 to 18 months to produce [MITD80] and can have disastrous consequences when done poorly. As more systems are built iteratively, the bulky nature of documentation makes constant updating difficult [PARM92] [GUIR87b].

1.1 The Goal: Improving User Documentation

It is painfully obvious that user documentation must be improved. Through discussion of aspects of document design and usability the need for improvement becomes clear. Close to this goal is the need to reduce the difficulty in altering documents to reflect design changes. This research attempts to illustrate this problem as well as suggest close-knit cooperation among interactive systems development and document development as a potential solution.

1.2 The Approach: Building Lines of Communication

The best way to improve documentation is via integration of both interactive software development and documentation development. In examining this aspect, the need for a common language or reference is obvious.

In order to develop an accurate understanding of the problem, an extensive literature review of 162 articles has been conducted, primarily dating from 1980 to 1991. Out of this initial body of material 95 articles were eventually incorporated into this paper. The search was limited to technical communication except in cases where an article specifically referred to a computer science journal or the article dealt specifically with aspects of development related to the documentation process. The search was limited in this way for two reasons; the bulk of material on this subject is found in this area and computer scientists have neglected this domain of knowledge in the past.

The years included were primarily for relevance. User documentation became a central concern during this time due to the increased availability of computers to the general populace. However, when significant or appropriate, older articles were incorporated.

The literature provides a historical background of documentation, from which a series of guidelines may be delineated for the production of good documentation. Within this framework lie the underlying threads for cooperative development. These threads are visible via User Action Notation, a design technique used in the development of user interfaces [HARH90]. UAN provides a common language of discussion that lends itself to cooperative development and facilitates construction of skeletal documentation.

1.3 The Contribution: More Flexible Documentation

Documentation is time consuming and expensive [MITD80]. Herrstrom notes research and editing each require almost as much time/effort as actual writing [HERD87]. UAN, by providing a framework for the development of skeletal documentation within an integrated environment, helps to build developmental cooperation. The skeletal documentation is more flexible and less expensive to change. The cooperation and flexibility produce improved user oriented documentation that is more complete, flexible and accurate.

1.4 Overview

Chapter 2 examines documentation, its costs and benefits, keys to its development and styles it may take. Knowledge essential in establishing the need for developmental cooperation.

Chapter 3, centers on how users, usability and learnability affect development. This is requisite to establishing linkage between systems development and document development along lines of task orientation.

The fourth chapter focuses on integration and iteration of both documentation and systems development.

Chapter 5 coalesces the previous chapters and the bulk of the literature into a family of documentation guidelines

The sixth and concluding chapter devotes itself to the selected technique to foster cooperation between development areas. The technique, User Action Notation (UAN), was created by Hartson, Siochi and Hix [HARH90] [HARH92] for use in user interface design. After a brief history, benefits to design and integration are elaborated as well as application of UAN to the guideline family. This application constitutes a taxonomy to facilitate construction of skeletal documents from design specifications enabling close-knit cooperation among systems and document development.

Chapter 2: Documentation

Page 69/79 ... The first thing you have to do ... [SOHD83]

2.0 Introduction

This chapter focuses on the nature of documentation. The costs and benefits of good documentation as well as keys to its production are elaborated. Also, three primary styles are discussed and compared. Within this context the mission goals of documentation; education, motivation and communication, are identified. Such ideas are crucial in establishing the need for closer integration among interactive systems development and documentation development.

2.1 Documentation Defined

User documentation is an actual an integral part of the system. This fact applies to all forms of documentation regardless of mode of delivery. Guillemette states that documentation is "all transmissible information that facilitates the analysis, planning and control, acquisition or development, operation and use or maintenance and enhancement of computer-based applications" [GUIR87a] [GUIR87b]. Walker provides a somewhat more readable definition; "documentation contains information that a user needs to know in order to operate the system. The general goal of documentation is making information available to users when they need it"[WALJ87].

Andrew Oram supports the idea of the component nature of documentation by defining documentation as the "design, planning and implementation of any interface element of a software system to enhance its usability" [ORAA91]. Rosich and others also add that documentation serves as an initial or primary interface to the system and thus helps users form their first impressions of that system [ROSD86] [GUIR87a]

[GUIR89] [TOMR82]. As such, a primary goal of documentation is to soothe fears and motivate use [SOHD83]. Indeed, documentation must transform users from a state of ignorance to a competence that allows them to achieve their goals with the given system.

From these diverse yet similar definitions, a good working definition of documentation can be developed. For purposes of this research, documentation may be defined as follows: documentation may be considered as a depository of information, both in print and online, that encourages use by education, motivation and communication. Documentation must serve as instructor, guide, reference and companion.

2.2 Discussion/History of Documentation

It is clear, as Oram states, a manual can explain a lot more than imagined. The classic problem is one of balancing the amount and depth of information to what the users need [ORAA86]. One of the primary causes is the "DP historical bind; the situation that arose between 1965 and 1980 when development outstripped the ability of users to keep up" [WILL81]. During this period, the first micro-computers were introduced and the audience of computing technology underwent a great diversification. No longer were computers only in the hands of highly technical, skilled professionals but ordinary people began using them for ordinary tasks. In the early 1980's technical writers began serious attempts to address these changes. Henderson notes that the new audience led to the development of more eye appealing manuals and greater focus on user needs [HENA84]. In the eighties, the field of document design literally ballooned. Schriver points out that Discourse Analysis, Human-Computer Interaction, Cognitive/Social Psychologies and Computing technologies are now contributing factors to the field [SCHK89]. Farkas notes that a possible synergism exists

between product support and document development [FARD86]. Indeed, many technical writers feel that product design and documentation should be a integrated process [GAIB79] [WEIE85] [SODC85] [TOMR82] [GUIR89] [VOGH87] [GRIR88]. Thus, historical evidence supports the integration of document and systems development. However, this integration lacks a common frame of reference that both designers and technical writers can use effectively. This gap in communication still hampers development and leads to disgruntled users.

2.3 The Costs/Problems of Poor Documentation

An average computer manual has 200 pages and takes from 10 to 18 months to produce [MITD80]. This is a complex and time consuming task and, if poorly done, can be disastrous. In one project, "documentation costs accounted for a significant amount of the application systems budget and 30% of all errors were due to poor quality documentation" [GUIR87b]. These problems are summarized in table 2.1.

Table 2.1. Costs/Problems of Poor Documentation

Frozen, not easily changed
Bulky containing extraneous information
Jargon and legalistic phrasing
Difficult to find information
Insufficient detail provided
Difficult to understand
Too brief, no details
Inappropriate audience analysis
Learning confounded
Less trust and credibility

Paper documents are 'frozen' when they are written. Since documents are not easily changed, they must be detailed and complete, this leads to bulky documents that contain more information than anyone wants or needs [GLUR82]. Additionally, in an iterative environment, 'frozen' documents by their nature do not allow documentors to adequately keep up with design [PARM92] [GUIR87b]. Some

manuals are so replete with coding and flowcharts that users need professional help to comprehend them [CLAF83]. Use of such jargon and legalistic phrasing makes information hard to find, incomplete with insufficient detail and requires a great deal of re-reading to understand [TOMR82]. In attempting to correct the problem, documents can become so brief that they become little more than glossaries [CLAF83]. Essentially, audiences are inappropriately analyzed and the resulting analysis yields documents which confound learning. Additionally, hidden costs may arise such as less software and document reuse [GUIR89]. Most costs are due to lost opportunities - such as failing to judge the audience [GUIR89](see chapter three).

Thus, writing styles and poor development practices are key concerns [CLAF83], but lack of managerial planning, policy and support for the documentation effort are also to blame for these difficulties. For many developers there is no perceived link between better documentation and organizational rewards, and since they are already accustomed to the old ways there is resistance to concentrating on the documentation [GUIR87a] [GUIR87b]. The disastrous aspect of this line of thought is that poorly designed material damages credibility and lessens user trust of the product [SODC85]. Thus, the greatest cost concerns loss of future sales and the self defeat of the documentation effort. The fundamental goals of education, motivation and communication are ignored and/or alienated.

2.4 Benefits of Good Documentation

The benefits of good documentation, summarized in Table 2.2, can easily be described as the antithesis of the problems of poor documentation. Possibly the most dramatic benefit is the engineering of overall cost savings in both time and money. Guillemette states that "good documentation

Table 2.2 Benefits of Good Documentation

Time savings
Money savings
Reduced rework
Reveals errors in design
Stimulates feedback
Point of common reference
Enables inclusion of information
Reduces need for user training
Reduces/Prevents errors
Promotes trust and product confidence

results in the facilitation of more efficient and effective software development" [GUIR89]. Quite simply, good document design can point out errors in system design as well as reduce document rework. Good documentation serves as a design tool that forces designers to "explicate design and definition" [GUIR87b]. It augments memories of developers and users alike and helps stimulate user feedback. Good documentation provides a point of reference and a communication tool across the development spectrum [GUIR87a]. Overall cognitive load is reduced and more advanced application features may be incorporated into the material. Also, good documentation reduces the need for user training and helps reduce/prevent user errors [GUIR89]. Last, it promotes commitment and confidence in the software product on the part of users and developers as well [GUIR87a]. In essence, good documentation adequately satisfies the mission goals of education, motivation and communication.

2.5 The Four Keys to Good Documentation

By examining the literature several commonalties may be seen. These common threads may be coalesced into four keys to good documentation. The keys to documentation are anything that allows documentation to complete its mission goals of education, motivation and communication. There are

four keys to good documentation, all of which are interrelated in some way. Specifically, they are:

- style/organization
- usability
- learnability
- audience appropriateness.

The first of these keys, style/organization, is a composite of both the orientation and presentation of the material. The orientation or style component may range from software orientation to minimalism and will be discussed in detail in section 2.6. The second component, presentation, deals with how the document appears. According to Horton, a document may be organized or presented in four ways: in sequence as in traditional text, by grid or in a tabular format, in a tree structure or in a web structure of dynamic links as in hypertext [HORW89].

The second key, usability, often proves elusive to define clearly. A simplistic definition can be borrowed from Maynard who defines usable manuals as those that maximize both their learning and retrieval capabilities [MAYJ82]. Usability is discussed in detail in chapter 3.

The third key is closely woven to usability but learnability is of sufficient importance to rate separate distinction. Learnability can be thought of as anything that promotes the acquisition of new knowledge. Learnability is also closely related to style/organization. Duin points out that document organization is a contributing factor to learning. Additionally, learning involves tasks external to any given system such as prior experience [DUIA89]. Learnability is elaborated more fully in chapter 3.

This brings us to our fourth key to good documentation, that of audience appropriateness. Specifically, this involves writing documentation to meet user needs and meet their personal goals and strategies. This is interrelated

with the other keys in that, as Soderston states, "usability is a function of the interaction between the text and the reader" [SODC85] [HUSK88].

The four distinct yet interrelated keys are bound together by a common thread - the user. Specifically, when all are equally considered, the mission goals of education and motivation of users as well as communication to users become less hazy more attainable goals. Style is discussed in section 2.6, the remaining keys and their relevance are elaborated in full in chapter three.

2.6 Documentation Styles

2.6.1 Introduction

The first key to discuss in detail is style/organization. As discussed in section 2.5, the style of a document refers to its orientation and the organization refers to its presentation. The presentation may take form separately from orientation and serves only as a delivery vehicle of the information. For brevity only style is discussed in detail. There are three main styles currently in use today: software orientation, task orientation and minimalism. These styles are either system focused or user focused resulting in two general classes of style. Software orientation, the eldest, relies on descriptions from the view of the system. Task orientation, in increasing use over the last decade, relies heavily on the user. Minimalism, an offshoot of task orientation, relies on the idea of active learning and letting the user lead.

2.6.2 Software Orientation, The Systems Approach

The eldest of the styles is software orientation, often called the systems approach. Partridge defines software orientation as "writing, structuring and organizing software user manuals based on the design and structure of the

software itself and telling the user the capabilities of the software by discussing its modules, routines or commands" [PARS86].

Systems approach documentation consists of "the hierarchical decomposition of learning objectives. Each objective is incorporated into a systematic lesson structure incorporating specific events of instruction: gaining the learner's attention, informing the learner of the lessons objective, stimulating recall or prior learning, presenting stimuli with distinctive features, guiding learning, eliciting performance, providing information feedback, assessing performance, and enhancing retention and learning transfer" [FARD90].

The biggest tangible benefit of this approach is that it fits nicely with the designer's internal model of the system. Martin acknowledges the internal model of the designer and points out that systems approach documentation for that reason yields marginally usable manuals that tend to alienate and anger users [MARP86]. Many older manuals, which used this approach, look like legal documents, are impersonal, passive and hostile to the novice [HENA84]. Another difficulty as Bethke points out is that readers of such manuals need to be somewhat familiar with the system which can be disastrous for novice users [BETF91a] [BETF91b].

2.6.3 Task Orientation

The next evolutionary step in document style has been the task oriented approach. Brockmann defines task orientation as "based on an analysis of the user's use of the program and is limited to what information is required to do a specific task using the program. Information not needed to do a specific task is not provided" [BROR90]. Task oriented books are no longer passive containers for any

information that happens to come along - they are active tools that apply to specific user tasks [BRAD90].

Task oriented manuals consist of start-up information, quickpath information and models for understanding [HENA84]. Bethke notes that three essential aspects of task oriented texts are that information is: easy to find, easy to understand and sufficient for the task at hand [BETF91b]. Task oriented information identifies the action of the user, the conditions and environment of the action, as well as the criteria for the task [HORR92].

One of the greatest benefits of the task oriented approach is that "since the information in the document explicitly mirrors observable user actions, prototypes are supported and more easily performed" [BROR90]. The benefit of this is that document design and product development may be integrated and both may draw upon the same user action information during development. Also, prototyping and usability testing of the 'working document' early on will allow documentors to catch flaws quickly and therefore reduce overall document costs.

One difficulty with the task oriented approach is that "explaining how the software works" and "explaining how to work the software are opposite documentation goals [ROSS86]." Care should be taken not to use task orientation from the view of the computer because the approach reverts to software orientation [SCHD86]. There is also a short term increase in the time required to produce this style of documentation as compared to the systems approach. Last, task orientation may require more text as it explicitly states information left implicit or ignored in the systems approach [BROR90].

2.6.4 Minimalism

The third style under discussion is minimalism. "Minimalism is the self-described label that a group of current researchers and writers have given to computer documentation's newest style of writing" [BROR90]. Minimalism is often mistaken as a commitment to brevity [FARD90]. Granted, minimalist documents are brief, but only as a side effect of their guiding philosophy. The basic principle behind minimalism is that training material must not impede the natural impulses of learners and it requires "careful attention to the behaviors of the intended users" [FARD90].

There are nine fundamental principles [BROR90][CARJ88a] [CARJ84] that guide minimalist philosophy.

Table 2.3 Minimalist Principles [BROR90][CARJ88a]

Let user lead
Slash verbiage
Guide with hints, questions
Readable in any order
Account for user errors
Provide early error recovery
Utilize prior knowledge
Use the situation for learning
Do not systemize development

First, use real tasks for training and let the user lead. Next slash verbiage, eliminate almost all orientational material in order to get the learner started quickly. Third, use questions and hints to guide the user. Also materials should be designed to read in any order. Fifth, help users by providing roadmaps for normal errors. Next, early focus must be placed on errors and their recovery. Seventh, utilize prior knowledge. Eighth, consider using the learning situation, as opposed to practical examples for learning examples, exercises and explorations. Last, avoid systemizing development and cranking out manuals

and concentrate on testing to determine optimization of learning [BROR90] [CARJ88a] [CARJ84].

Minimal manuals by slashing verbiage are more task oriented and to the point. Since readers are not necessarily passive and want to get work done, this approach facilitates that goal [BROR90] [FARD90] [CARJ88a][CARJ88b] [CARJ84]. Minimalist design is best used iteratively, which facilitates prototyping and testing to allow for early user feedback and design correction. Last, in such systems, users "get started faster, produce better work and spend less time in errors and error recovery. This indicates better learning" [CARJ84].

One difficulty with the minimalist philosophy is that it constantly compares itself to the outdated systems approach rather than contemporary task oriented approaches. This somewhat lessens the impact of minimalism in that it cannot accurately be gauged among its contemporaries [FARD90]. Brockmann [BROR90] adds the following list of possible difficulties to the minimalist approach:

- **learning by self discovery is less predictable and may result in gaps or lack of depth in learning**
- **assumes motivated audience, may or may not be the case**
- **when is concision cryptic? liable?**
- **danger of just recklessly slashing verbiage.**

[BROR90]

Last, Mirel notes that while studies support that minimalism encourages exploration, the sample sizes are too small for generalizations [MIRB91b].

2.6.5 Comparisons/Contrasts of Documentation Styles

As discussed in sections 2.6.2 - 2.6.4, the three styles may be grouped into two classes of orientation with regard to developing documentation. It may be oriented to the user or the software. The task oriented and minimalist approaches attempt to utilize a user focused philosophy. The

systems or software oriented approach structures itself around the design and function of the software.

Though systems and task/minimal orientations are diametrically opposed in their approach, several similarities are nonetheless apparent. Each, in its own way, tries to achieve the mission goals of education, motivation and communication. For example, as previously discussed both classes and all three styles attempt to make use of prior knowledge and examples to facilitate new learning. Also, system's approach documents attempt to guide learning through a systematic lesson structure [FARD90]; and though the user is more in control, the "models of understanding" [HENA84] used in tasked oriented material also serve to steer the learner in the desired direction. Minimalist design shares in this common thread in that the philosophy grew out of experiments in guided exploration [CARJ88a] [CARJ88b] [CARJ85a][CARJ85b][CARJ84].

Most differences in styles are due to the differing nature of their focus. Although as mentioned in 2.6.4, no studies of current task oriented approaches versus minimalism are available for direct comparison, by examining how the approaches differ from the systems approach some additional comparisons are visible. Both task oriented and minimalist approaches show improvements over systems oriented documentation. However, task oriented material takes 42% longer to initially construct compared to systems documents [BROR90]. Minimalist design is seen as an attempt to merge task orientation's user focus with the shorter development time associated with software orientation [BROR90].

Odescalchi shows that user failure rates are 310% higher with software oriented material when compared to task oriented material. Conservatively, with task orientation, gains in user productivity of 41% were also recorded

[ODEE86]. A similar 40% less learning time was recorded for minimal manual users compared to their software oriented counterparts [CARJ88a][CARJ84]. Also given a number of tasks minimal manual learners were able to perform 2.7 times as many subtasks as software oriented subjects. Perhaps the most startling, in another experiment, minimal manual users achieved 93% more per unit of time than software oriented users [CARJ88a] [CARJ84].

From the above evidence, it may be easily concluded, as many researchers already have, that a user focused approach is best for documentation of interactive systems. Specifically a mixture of task oriented and minimalist approaches is required that focus on the user and real tasks [MIRB91a] [BROR90] [FARD90]. The area of import with regard to the goal of close-knit cooperation stated in section 1.1 is that task oriented and minimalist approaches both lend themselves to iterative development. As mentioned in 2.6.3 and 2.6.4, iteration facilitates more rapid development and reduced costs [BROR90][CARJ84]. The specific aspects of iteration and integration are presented in chapter four. This has an impact on the choice of UAN as a technique to facilitate developmental cooperation and is discussed in chapter 6. Finally the specifics of user centeredness or usability are elaborated in chapter three.

2.7 Summary

This chapter has identified the primary mission goals of documentation: education, motivation and communication. Without close-knit cooperation between developers and documentors, these mission goals become unattainable and documents are only marginally useful. The background, costs and benefits of documentation were elaborated. The keys to good documentation, style/organization, appropriateness, usability and learnability, have been introduced. Through these discussions the historical and evidential grounds for systems and document integration have been introduced and the ground laid for further discussion. Chapters three and four will build on these foundations dealing with issues of documentation keys and iteration/integration respectively.

Chapter 3: Users, Usability and Learnability

A writer's capacity to adjust to an audience depends on the ability to internalize that audience [SODC85].

3.0 Introduction

Documentation is integral to an interactive software system. A user focused approach is best for interactive software documentation (chapter two). However, there are considerations beyond style or presentation of material. Specifically, the remaining keys to good documentation: appropriateness (awareness of target users), usability of the material and learnability of the material - all require consideration. This chapter addresses the remaining keys and their implications towards satisfying the documentation goals of education, motivation and communication.

3.1 Users

Systems can and do have an impact on the user; thus, the role of the user cannot be ignored [BANL86]. Too often there is a tendency to get lost in the details and forget the individuals that eventually use the system being developed. Additionally, there is an economic motivation to consider: the extent of use and overall degree of satisfaction are key to the success of development [MOYJ82]. Awareness of the user, also called audience appropriateness, is the focus of this section. This key to good documentation is elaborated by examining how users interact with text/systems, individual differences and diversity that set users apart, as well as the characteristics that they share.

.

3.1.1 The Active User

The active user is synonymous with the active learner [MIRB91a] [MIRB91b] [CARJ84] [CARJ85a]. The connection to

learnability is elaborated in more detail in section 3.3. The Active user "is not in the role of passive receiver, but rather in the role of participative observer" [SODC85]. The active user does not want to read but achieve some objective. "Active users want to be actively involved in learning and to understand why and what they are doing" [REDJ88]. In support of this idea, one study reported that more than 90% of users were unwilling to read systems documentation [GUIR89]. In a related area, experimentation in exploratory learning and active users were contributing factors to the development of minimalism [CARJ88a] [CARJ85a] [CARJ84]. This connection to learning is discussed in section 3.3. Because of this exploratory or active nature, documentation is often not read linearly [HENA84] [FOWS86]. The documents therefore must serve as roadmaps for the system and are, as discussed in chapter 1, integral components of the system.

3.1.2 Individual Differences

Users are active participants within the system, but only to a matter of degree. Each person behaves and reacts differently. Each individual brings to the setting prior experience which shapes their expectations, goals and determines how they react to the system [CARJ88b] [MIRB91b] [MIRB92] (see section 3.1.4). Egan states that our differences "usually play a major role in determining whether humans can use a computer to perform a job effectively. In a group of 30, performance differences of 20:1 for certain computer-based tasks are not uncommon" [EGAD88]. The reason - users differ in their goals, level of expertise, style of usage, learning and preferences [WALJ87]. Egan adds, "differences among people usually account for much more variability in performance than differences in system design or training procedures"

[EGAD88]. It is for this reason that users are integral components of the system. Anything that takes into account their needs and aptitudes naturally leads to better documentation and systems that satisfy the mission goals of education, motivation and communication.

3.1.3 The Spectrum of Knowledge

It is difficult to classify the typical user of a software product [SOHD83]. One general classification is for whom the material is written. A popular method is based on an expert/novice dichotomy [EGAD88] [MAYR88]. A dichotomy that is usually realized in the form of the ready reference/tutorial respectively.

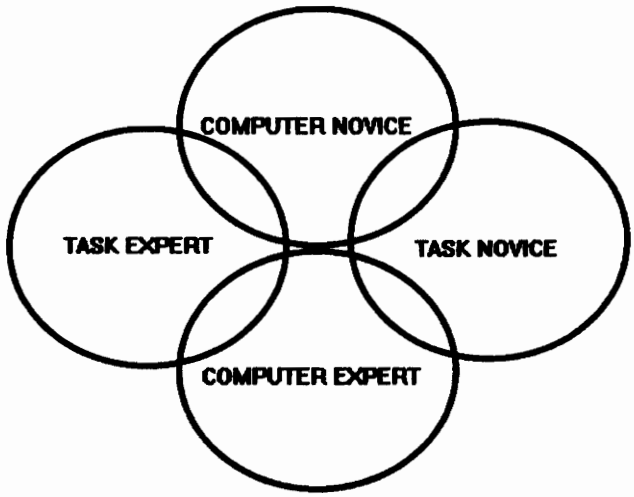

FIGURE 3.1 SPECTRUM OF KNOWLEDGE [ROSS86]

Walters and Rosembaum developed an audience model of a user's guide [ROSS86]. This is more aptly described as a spectrum of knowledge (Figure 3.1). An individual may be a novice to the computer system or to the task at hand or both. At the same time, a user may have expertise in the system area or subject area or both [ROSS86]. The two dimensions are not dependent on each other; as individuals

gain knowledge they move along the spectrum. It is a rarity for 'generic' experts to exist. Users may have 'local expertise' but should the proper sequence of events transpire, they are again reduced to near novice level [EGAD88]. The advantage is that experts have better heuristic reasoning based on experience [ROSS86] [MOOM88] [MAYR88]. Experts, like novices, do not first seek information from the documentation. The expert applies existing heuristics based on prior knowledge and the novice attempts to formulate heuristic strategies to enable usage [ROSS86] [WRIP85]. Thus, audiences (users) may be generally classified as a novice or expert, but there is a blurring distinction between the two. Still, the mission goals of education, motivation and communication are highly relevant.

3.1.4 Holist/Serialist

Research indicates that most users desire to be productive and do not wish to be hampered in their goals [CARJ88a][CARJ85a]. There are two types of learners as identified by Pask, the holist and the serialist [PASG76]. Holist learners are quite similar to the active learners as described by Carroll [HORR92]. Horn adds, "Holists like to get the big picture before getting the details. They like to start reading in the middle of books. They skip around alot" [HORR92]. This is quite similar to minimalism [CARJ88a] [BROR90] and the active learner as discussed in sections 2.6.4 and 3.1.1 respectively. The serialist "likes to proceed through learning step by step. They like to fulfill all prerequisites necessary for one level before going on to the next" [HORR92]. Like the spectrum of knowledge of 3.1.3, this reflects the depth of individual differences and the need for audience awareness.

3.1.5 User Characteristics

Despite the subtle gray areas discussed in sections 3.1.3 and 3.1.4, there are several characteristics that users share, summarized in table 3.1. Users are occupied with external pressures. Often, users are working multiple tasks at once and "they want to utilize the software as painlessly and quickly as possible" [PARS86]. Should an error occur or a command be forgotten, the user will need information.

Table 3.1 User Characteristics

Are occupied with real tasks
May search on incorrect keys
Have difficulty finding information quickly
Have difficulty finding information details
Search interrupts work
Are often annoyed/frustrated
Follow nonlinear strategies
Tap prior knowledge to search
Are active participants

It is at this point that the documentation effort becomes critical. Users may reach incorrect keys from which to search [WALR84] [GLUR82]. If not properly designed, users have difficulty finding information quickly in appropriate detail [OMAC84]. This search interrupts the task at hand, leading to annoyance and frustration [WRIP85]. In attempting to locate this information, users follow a non-linear reading strategy [FOWS86] [HENA84]. To aid in the search, users tap prior knowledge. Prior knowledge exists in our minds as schemata. These schemata serve as a "unified system of background relationships whose visible parts stand for the rest of the relationship" (section 3.3 for more on prior knowledge) [DUIA89]. Schema aid recall, search strategies and comprehension [DUIA89]. Foremost, users are participative observers [SODC85], who search for information, digest and attempt to understand the

information and then apply that information to the task at hand [WRIP85]. These characteristics have an impact on usability and learnability and are elaborated in sections 3.2 and 3.3 respectively.

3.1.6 Implications for Documentation

The implications for documentation can best be seen through analogy. Consider expert systems and expert/novice dialogues. Like the users/learners of computer systems, "people asking advice from experts often do not know what information they need to obtain in order to achieve their goals and consequently fail to ask the most appropriate question" [POLM85].

Table 3.2 Expert advice strategies [POLM85]

Give direct advice based on query
Direct advice to an inferred plan
Suggest alternate action to achieve goal
Suggest action to make goal possible

Users of documentation, may fail to search on an appropriate key (see section 3.1.5). Experts may, depending on the situation, give direct advice based on questions asked, address an inferred plan, suggest an alternate action or suggest an action that enables the possibility of the intended goal of the advice seeker [POLM85]. These possible actions exemplify the mission goals of documentation to educate, motivate and communicate. In fact, "the nature of expert systems demands even more focus on user needs and increased attention to human factors" [McGK86].

Since most forms of documentation seem rigid, choices must made about information detail, what information to include and how to make users aware of that information [EGAD88][ORAA91][ORAA92]. One approach is to consider Documentation as a static removed expert on its respective system and apply the advice strategies just mentioned.

3.2 Usability

Usability is the next key to good documentation. In section 2.5, usability was simply defined with regard to those manuals that maximize their learning and retrieval capabilities [MAYJ82]. Usability embodies the user's need of reference and tutorial as discussed in section 3.1. A user's ability to use a system successfully depends on the ease with which usage can be communicated [GOON87]. Bethke et al suggest that usability consists of three key principles. Specifically these are[BETF91b]:

- easy to find information,
- easy to understand information
- task sufficient information

Documentation usability concerns the ease by which users achieve their objectives in reference to the written material [GOON87].

3.2.1 Connections to other Keys to Documentation

Recall from chapters one and two that documentation is an integral system component and from 3.1 that users are also central to the whole of any interactive system. It is clear that "usability is not only task related but people related" [GOON87]. Soderston supports this notion by stating "usability is a function of the interaction between the text and the reader" [HUSK88][SODC85]. Design of a usable system requires understanding of intended users, their levels of expertise, the expected usage and how their needs change as they gain experience [GOON87]. In fact, as Girill observes, what separates documentation from other forms of technical writing is that its readers are actually users [GIRT91].

3.2.2 Key Usability Characteristics

The three usability characteristics stated in 3.2 are critical for successful documentation. Bethke notes that information may be easy to find, easy to understand but if it is wrong for the given task it is unusable [BETF91a][BETF91b].

3.2.2.1 Ease of Finding Information

The first characteristic is ease of finding information. Girill states that from the eyes of the reader, documentation consists of potential answers. Documentation is inadequate if it fails to provide easy access to those answers [GIRT91]. Thus, information must be easy to find in order to be easy to use. Bethke identifies three factors that contribute to the ease of finding information[BETF91b]:

- **Consistency**
- **Landmarks to Information**
- **Arrangement**

Consistency allows users to become accustomed to the text, know what they'll find and where they'll find it [BETF91b]. Next, pointers act as landmarks or signposts and help the user get the information they want quickly. Pointers range from indices, tables of contents to highlighting [BETF91b] [PRID91]. Lastly, arrangement "anticipates ways in which users may search for information such as alphabetical or chronological listings" [BETF91b].

3.2.2.2 Ease of Understanding

Bethke [BETF91b] decomposes ease of understanding into:

- simplicity,
- concreteness
- naturalness.

Simplicity means the vocabulary suits the audience and data are presented in a comprehensible manner [BETF91b]. This

usually involves, as noted by Miller, the "magic number seven plus or minus two", or a limiting factor to human information processing that limits our cognitive capacity to five through nine items at any one time [MILG56]. When subjects are complex, "they are organized into layers with nine or less items per layer" [BETF91b]. Concreteness refers to specific verbal descriptions rather than generalities. Naturalness refers to the idea that information flows in a manner such as it is used and understood [BETF91b].

3.2.2.3 Task Sufficiency

Task sufficiency has a strong link to the task-oriented approaches discussed in chapter two and, of the three characteristics, perhaps the most elusive. Mirel accurately describes task sufficiency as a "rich and complicated concept" [MIRB91a]. Task sufficiency consists of three contributing factors [BETF91b]:

- completeness,
- accuracy
- exclusivity.

Completeness refers to the inclusion of all necessary information to complete a given set of tasks. Next, information must be accurate and reflect the facts of the system and tasks at hand. Finally, information not needed is not provided or excluded from the document [BETF91b].

3.2.3 Key Usability Design Principles

The characteristics discussed in 3.2.2 make up the concept called "ease of use". In addition to this dimension Guillemette identifies "ease of learning" as another high level dimension of usability. These two dimensions are most easily seen in references and tutorials respectively [GUIR89]. "Ease of learning" for purposes of discussion is essentially learnability. Learnability is discussed in

section 3.3. The "ease of learning" concept specifically refers to the degree of learnability for a given system. These two dimensions are not fully compatible, maximize one and the other suffers [GUIR89][TOMR82]. As discussed in section 3.1 the needs and goals of the users are of prime consideration. The final choice of which is more requisite, ease of use or ease or learning, depends upon the intended audience, their prior knowledge and information needs.

3.2.4 A Framework for Usability.

From discussion, it can be seen, as stated by Gould, that "any system should be easy to learn (and remember), contain functions people really need and be easy and pleasant to use" [GOUJ85]. Soderston adds that conventional methodology has made it difficult to access individuals acting in a special time and place" [SODC85].

Table 3.3 A Framework for Usability [GOUJ85]

specification of the user interface
development of behavioral goals
analyze/describe target users
analyze/describe user tasks
define usability measures/specifications
collect critical data
analyze critical data
revise/repeat until criteria met

"Usability is not something that the technical writer adds when writing the documentation after the hardware and software have been developed" [VOGH87]. Indeed, as with the task oriented approaches from chapter two, a prototyping environment is more conducive to usability. Gould states that empirical testing, early focus on user tasks and iterative design are key considerations to promote usability. In the design process, usability consists of development of behavioral goals, description of intended users and their tasks, specification of the user interface,

definition of measures for collecting critical user data and, upon analysis, revision and repetition [GOUJ85]. In this way usability testing enables developers to assure quality not just measure it after the fact [SODC85].

3.3 Learnability

Chapter 2 defined learnability as the acquisition of new knowledge. The idea of learning has been alluded to throughout this chapter. In section 3.1.5 the ideas of prior knowledge and schemata were introduced. This section elaborates on these concepts, the types of knowledge and how readers learn from text.

3.3.1 Types of Knowledge

There is a general consensus that two types of knowledge exist:

- declarative and
- procedural.

"Declarative knowledge is our knowledge about things that we can express on demand. Procedural knowledge is our knowledge of how to perform skills" - knowledge not always consciously available [LIEJ91]. Anderson relates these two types of knowledge/learning through a theory of skill acquisition. He divides learning into declarative, knowledge compilation and procedural stages. Over time, new declarative knowledge is compiled into procedural form through practice [MOOM88] [ANDJ86]. Procedural knowledge becomes relatively automatic; an example would be driving a car. The goal of documentation then is to educate and communicate information to a user to the point of at least declarative knowledge and hopefully procedural knowledge and a thus a degree of expertise.

3.3.2 Factors that Affect Learning

How do we learn? Prior knowledge and schemata were introduced in section 3.1.5. These factors are elaborated and metaphor and pragmatic implication are introduced.

table 3.4 learning factors

Schema Theory
Prior Knowledge
Metaphor
Pragmatic Implication

3.3.2.1 Schema

A schema "can be described as 'the abstract prototype of a class or objects, events or situations'. For example, a schema for a face would contain places for eyes, ears, nose etc." When grouped together these objects are recognized as a face [LIEJ91]. Schemata empower comprehension and recall [DUIA89]. This is done by organizing information into "instantiations of schema" [LIEJ91].

3.3.2.2 Prior Knowledge

Prior knowledge is the vast collection of these schemata in our mind. Duin adds, numerous schemata function as "a unified system of background relationships where visible parts stand for the rest of the schema" [DUIA89].

3.3.2.3 Metaphor

The next factor to learning, metaphor, has been one of some debate. The basic idea is that the act of learning is composed of analogous comparisons [CARJ85b]. "Functional metaphors can make the function of a machine more compatible with the user's view of the task" [CLAC83]. Metaphors can be operational, and establish expectations about what is available and how to use it [CARJ85b][CLAC83]. Organizational metaphors use location and distribution to classify information. Finally integrating metaphors show

commonalty and share conceptual components (i.e., paper to a typewriter and a file cabinet) [CLAC83]. "The role of metaphor may vary, but it serves to draw attention to certain critical elements or relations of the target domain" [CARJ84][CARJ85b]. However, care must be taken in selection of metaphors and attention paid to the selected audience to assure that proper analogies are drawn to enable the development of proper mental models [CARJ85b].

3.3.2.4 Pragmatic implication

Pragmatic implication is based on the human information processing system. Stored information is actually verbalized or logically inferred from dialogue. However, under certain circumstances, inferences are made pragmatically as to what the speaker means because the content of the message may strongly suggest some information but not the information expressly implied nor logically derived from the original content [HARR78]. Research supports that pragmatically inferred information is recalled and recognized as directly asserted fact [HARR78]. For instance: "the 20 ton semi struck the Ford Fiesta" implies that a truck destroyed a small compact car. This inference is plausible despite the fact that the Ford family could have been having a party and was hit by a truck. Even if the car was the object struck, was the truck traveling fast enough to destroy the car? The consequences of this learning factor are a reinforced need for audience awareness and knowledge of the system. Additionally it supplies a further argument for careful selection of terminology and choice of metaphor.

3.3.3 Learning from Text

Readers (users) are active participants. There is a need to make documents user-centered or as Huston suggests "reader-based not writer-based or content-based" [HUSK88].

In section 3.3.2, the factors that affect learning were discussed; but how do readers learn from texts? Basically there are three approaches[REDJ88]:

- **reading to do,**
- **reading to learn**
- **reading to learn to do**

These concepts were originated by Sticht and extended by Redish and others [REDJ88][GUIR89][CARJ88b].

"Most occupational tasks are 'reading to do'"; they require interpretation of existing knowledge in the external world to accomplish a task [GUIR89]. While 'reading to do', the goal of the reader is to produce immediate action - not learn [REDJ88]. This is analogous to looking up a number in the phone book.

The next style of reading is 'reading to learn'; best exemplified by the traditional tutorial or textbook [GUIR89]. The primary goal when 'reading to learn' is to acquire and later recall information for use [REDJ88].

As a general rule, 'reading to do' is far more common in the work environment than 'reading to learn' [GUIR89]. However, there are situations that fit neither of these categories. A third is necessary, Redish refers to this third method as 'reading to learn to do'. This is needed under circumstances such as a typical computer tutorial [REDJ88]. Users are active and do not want to be bogged down reading [CARJ88b] [CARJ85a] [REDJ88] [MIRB91b]. In this setting, the 'reading to learn to do' style is essential, but Redish adds that for references and user guides 'reading to do' is more appropriate [REDJ88].

3.4 Summary

The remaining interrelated keys to good documentation have been elaborated. Users are active and integral to the system. Usability consists of ease of finding information, ease of understanding information and task sufficiency of the information. Ease of use runs in parallel to ease or learning when considering usability. Even though related to usability, learnability is of sufficient importance to merit separate consideration. When learning one draws on prior knowledge based on schemata to make analogous comparisons to create new knowledge. The choice of analogy or metaphor is important as learners may draw incorrect pragmatic inferences based on the metaphor. These three keys all reiterate the need for user focused, integrated, iterative development. By considering users, usability and learnability when producing documentation the goals of education, motivation and communication not only become more feasible, they become more relevant as well.

Chapter 4: Integration and Iteration

Don't procrastinate, Iterate! [MINA83]

4.0 Introduction

Documentation is an integral system component. The impact of usability and learnability further stress the need for a user focused approach towards document development. Clearly, to maximize the usefulness of the system; the document and systems development processes need to be integrated. This chapter concentrates on the issues of integration and iteration and their implications for the documentation effort.

4.1 Integration

The user interface is a complex entity; the software, the workstation, training and manuals all work together to create the frame-work within which the user works [GOUJ85]. Thus, "we should apply the same ergonomic principles to our written communication that we apply to a new user-oriented on-line computer system"[MARP84]. Integration, then, is the simultaneous, concurrent development of both systems and documentation [DANM87][FISL88][ZEIK88][HOUD91].

4.1.1 Examples of Integration

This section gives three case studies of integrated development. The first two come from the literature and the third is from a data point of current research at Virginia Tech.

One researcher [FISL88] reports that during a recent project, user descriptions were little more than job titles. This was not a sufficient level of detail to be of help. Such information must be collected early to be useful. The documentors decided to act early to ensure the task

sufficiency of the information and to accurately detail the intended audience. The developers really saw no relevance at this early a stage in the product life cycle. Business planners initially refused such early assistance due to the unorthodox nature of the idea. Documentors wanted to catalog and glossarize terms being used in design in case they were later going to be used in user materials. Developers protested that such was not the case. However, after analysis, a few terms were of importance to the user and the developers themselves were inconsistently applying them in the design. Although there was resistance at the interference, trust and appreciation soon arose among the respective groups. The resulting functional specifications were not all they could have been, but they contained a great deal of user and task-oriented information. The specifications were beneficial to designers as they gave their functions a context. Lastly, the documentors were credited with ontime delivery of a 'usable' functional specification [FISL88].

A similar experience is reported by another researcher [DANM87]. Once involved in the process, she had "no reservations about asking questions". Once she began an active role at the design meetings, initial engineer reservations about documentors not making meaningful contributions "quickly disintegrated." In another experience, initial reservations were a bit more pronounced and vocalized. This led to a breakdown in communication for all concerned. Ultimately the team did come together in order to get the job done. Overall, satisfactory working relationships were produced and the experiences showed her strengths and weaknesses in a new light. Beyond the internal cohesion and mutual respect was an increased level of customer satisfaction and confidence [DANM87].

The next case involves current research at Virginia Tech. A technical writer [PARM92] was appointed the task of developing user documentation for a large interactive system. The system consisted of around 300,000 lines of C code. The documentor began somewhat late (than the other cases) in the development process and was given an initial design document of approximately 400 pages from which to produce the documentation. The documentor utilized User Action Notation (UAN) to analyze the design [CHAJ92][PARM92]. The analysis revealed a task hierarchy which then served as a foundation for a table of contents. User Action Notation is discussed in depth in chapter six. Every two weeks, the documentor would meet with the project managers and discuss issues of the design, the system or the documentation. In the course of developing the documentation; flaws and omissions in the design were identified, leading to subsequent revisions in design. A sense of camaraderie was soon established between herself and the rest of the project team as a result of the close interaction [PARM92].

4.1.2 Benefits of Integration

Table 4.1 Integration Benefits

Shared User analysis
Consistency of information
Window to development flaws
Documentors serve liaison role to users
New perspectives to development
More Comprehensive design
Document, system completed simultaneously

An obvious benefit, shownin table 4.1, is that both documentation and the interactive system evolve from a shared user analysis. This assures a degree of consistency between the manual and the product. Because of this common foundation, Guillemette notes, documentation problems may

reflect design problems which have an impact on usability [GUIR89]. Soderston adds that whenever a user encounters a problem performing a task, documentors should see a deficiency in design [SODC85]. Reciprocally, the system can also serve as a tool for checking deficiencies in document design. Houghton-Alico points out that "documentation can be used as a tool for facilitating system tests (and) the accuracy of documentation can be ensured during systems tests" [HOUD91]. Developers may not have a clear grasp of user needs due to immersion in design. Documentors act as a liaison between these two differing groups. The documentation acts as a buffer of common ground [FISL88].

Many qualities that make person-to-person dialogues possible are also essential to human-computer interaction [BLAT84]. Technical writers and their background in communication can bring these ideas into perspective during design. This primarily involves a focus on users and their tasks (see chapter 3). Similarly, some ideas that lead to readable programs help in writing readable English [GUIR89].

Thus, both specialties may benefit from cooperation. Another benefit is that integrating both processes requires thought and planning. This planning yields a more thorough, complete design specification [HOUD91]. Fisher adds that early verification of user requirements by documentors discovers needed functions that designers may have considered unimportant or afterthoughts [FISL88]. Last, "the documentation will be available for release with the system and will provide a tool for the immediate use of the system"[HOUD91].

4.1.3 Barriers to Integration

Developers are often annoyed at the intrusion of writers into the design process [DANM87] [FISL88]. Zeidenstein suggests that one cause may be that "technical

writers are not seen as professionals". Engineers consider writers "impediments" to their progress [ZEIK88]. These barriers are summarized in table 4.2.

Table 4.2 Barriers to Integration

Developers annoyed at intrusion
Documentors previously 'reported' passively
Organizations fear change
Some fear user analysis is skipped

One additional problem is the historical role technical communicators have taken in the development process. Documentors have assumed the role of journalist, reporting on a system after the fact and not being active in its creation. Writers 'need to make the news' not write the news [FISL88] [ZEIK88]. This historical role has typecast both documentor and developer. When Fisher's group wanted to become actively involved early in development they met resistance because "we'd never done this sort of thing before" [FISL88]. This suggests an overall fear of change and organizational resistance. Another significant barrier is skipping of the user analysis. Houghton-Alico states that integration difficulties "might be used to force approval of specifications that have not been well conceptualized or verified for usefulness just to get specifications completed so work can start" [HOUD91]. These barriers must be considered in development planning.

4.1.4 Implications for Documentors

In the course of getting work done, the user and/or the documentor may utilize analogous comparisons to stimulate learning. If care is not taken, these metaphors may yield incorrect pragmatic inferences (section 3.3.2.4) [HARR78]. This complexity requires close, early attention on the users and their needs. This requires that documentors must participate in design [TOMR82].

Houghton-Alico [HOUD88] describes a program for simultaneous development called TIA. The model stresses:

- **Transactional - communicative action among at least two parties (developers + users).**
- **Interactive - reciprocal action between parties**
- **Analysis - detailed examination of needs to detect crucial requirements**

While this model claims to work regardless of the form of the development effort (traditional or iterative), the evidence in the literature [GUIR87b] [GOUJ85] [CARJ88a] [CARJ88b] [BROR90] suggests the iterative approach as most suited to interactive systems development.

4.2 Iteration

This section elaborates on iteration, how it further shows the need for integration and the connectivity to the keys to good documentation. The iterative method as well as its benefits and costs are described.

4.2.1 Iteration defined

A distinguishing characteristic of computer documentation is the need for repeated updating [GUIR87b]. New releases of software are frequent and errors must be corrected. Iteration through prototyping is the construction of a "'working model' of the product you are trying to design. Prototyping is not a new idea at all; engineers, architects and the like have been building models of their designs for decades" [GERV87]. The main purpose of iteration is not to fine-tune but adjust and confront each harsh user reality [GOUJ85]. Iteration, with regard to documentation, consists of "collecting information on user needs, designing an integrated set of tools to support those needs and evaluating the effectiveness of those tools" [OMAC84].

4.2.2 Need for Iteration

Farkas suggests a possible synergism between documentors and technical support personnel [FARD86], because support technicians are the first ones in an organization to find out a problem exists. Farkas adds that documentors need this information in order to improve the next manual release or prepare a revision package, if needed [FARD86]. Houghton-Alico observes that a great deal of maintenance costs are attributed to misconceptions and misunderstandings. Other studies reveal that nearly half of the serious problems with design are not detected until after user testing. Finally, the relative cost for correction during the final testing is 50 to 100 times greater than correction early in design [HOUD91]. After the product reaches the market, the costs are astronomical, not only in monetary terms but in time and reputation as well. Prototyping or iteration lessens these worries by "accommodating uncertainties in the requirements (specifications) by the presentation of a small scale working entity which serves as a baseline for critical analysis and communication among users and developers" [GUIR87b]. Thus, iteration is necessary for documentation and development of interactive software systems.

4.2.3 Application of the Iterative Method

Renaldo defines the iterative method as being composed of research, design, development, testing and production [RENL87]. Research consists of collecting information on user needs, analyzing the product and defining the audience [RENL87] [GUIR87b] [OMAC84]. The prototype is next designed and baseline documentation developed. Next, from design, working documentation is produced. Usability testing is then conducted by presenting the prototype to prospective users for evaluation. Feedback is recorded and later analyzed as

Table 4.3 Iterative method applied to documents

Research user needs
Research product
Define Audience
Produce baseline documentation.
Flesh out Working Document
Conduct usability tests
Encourage/Analyze feedback
Upon meeting criteria, document completed

input to a new research stage. Based upon this analysis, the design is further refined and tested until all the specifications defined in research are adequately met. At this stage the final product is released [GERV87] [OMAC84] [RENL87] [GUIR87b]. The iterative method, shown in table 4.3, is principally the same for text and software [GERV87]; yet another ground for the integration of documentation and system development. There is a ready comparison with table 4.3 and the framework for usability discussed in chapter three. A careful user analysis and accompanying usability specifications must be observed and these data used to gauge successive iterations. When all specifications are met or exceeded then the iterative process ends. If not properly observed, the iterative process could repeat needlessly.

4.2.4 Benefits of Iteration

Table 4.4 Benefits to iteration

Serves as a testing tool
Less cost/commitment to single avenue
Errors easily found/corrected
Encourages Feedback
Allows foresight

There are a variety of benefits, listed in table 4.4, most concern less costs in time, resources and money. With iteration there is a lower risk of failure; all organizational resources are not engaged in full-scale implementation. At the end of each iteration, cost/benefit

analyses are performed to determine the next course of action [GUIR87a]. Prototyping can thus provide the capacity of a testing tool, mistakes and errors are found earlier and are less expensive to correct [RENL87]. Prototyping further allows early user feedback and greater user involvement [GUIR87b]. This is essential for developing user centered documentation that satisfies the goals of education, motivation and communication. This yields an overall smaller relative effort [GUIR87b]. One study shows prototyped systems require 45% less effort to develop and were 40% smaller than 'pre specified' systems [BOEB84]. Prototyping also enables developers to 'visualize the end product'. This allows setting realistic objectives and enables a certain degree of foresight by the developer [RENL87]. All of these benefits enable the encapsulation of variety, that is designers are able to try more ideas more affordably in order to produce the best possible system.

4.2.5 Concerns/Problems of Iteration

Table 4.5 Problems of Iteration

Danger of only seeing the obvious
Indeterminate ending
Difficult to budget
Audience will 'change' during process
Require cooperative environment
Possibly limited applicability of process

One concern in table 4.5 is that prototypes only reveal the obvious. If done properly, iteration allows developers to approximate user needs and 'zero in' through further iteration [GERV87] [BOAB84]. Another difficulty is when does the process end? A possible answer to this occurrence is that, when iteration does not reach closure, a fundamental design assumption is wrong or the design is not practical [GERV87] [BOAB84]. Guillemette adds that this difficulty may cross over into budgeting and accounting, specifically with

regard to management of the iterative process [GUIR87b]. There is also a need to maintain a high level of user participation and manage expectations for the product [GUIR87b]. In other words, keep users involved and informed that the prototype is not the system. Users and developers must both be aware that the first time is never the last time. Diminishing returns and the prospects for tangible results drive the process. During this time the audience will undoubtedly change [MINA83]. As discussed in chapter one and two, traditional documentation is too 'rigid' to adapt iteratively. This is especially true in the case of approximated audiences. Like integration, there is a need for a cooperative environment to facilitate iteration. The lack of a cooperative environment is often due to participants prior experience or bias toward other paradigms [GUIR87b] [GOUJ85]. There is a limited applicability of the iterative method, such as well defined areas where users and processes are well known and there is little perceived benefit from a prototype [GUIR87a].

4.2.6 Implications.

Since active users are reluctant readers [MINA83], documentation must be comprehensive, detailed and kept to a minimum. Guillemette notes one study that found "on average over three times as many pages of documentation were generated by standard development teams" to convey the same information [GUIR87b] [BOEB84]. Since usability testing is key to the iterative process, iteration is conducive with the ideas of usability and learnability. In 4.1 it was discussed that document and systems development can point out deficiencies in each other. Renaldo observes that problems with document prototypes similarly reveal difficulties in system design and vice versa [RENL87]. As seen in 4.2.3 and 4.2.4, iteration enables a user centered

view. Finally, as Gervickas observes, nothing can replace good design. "However, any tool that can make the process of integrating the user's needs into the product more efficient should be seriously considered" [GERV87].

4.3 Summary

This chapter has discussed the aspects of integrating document and system development efforts and why they are important. Additionally, iteration was defined and discussed in detail. What remains is how to bridge the gap between documentors and developers. One of the difficulties in 4.1 was the differing views held by both documentors and developers, let alone how the user and developer differ. A technique is still needed to provide a common frame of reference, smoothly integrate these two processes and foster developmental communication across channels.

Chapter 5: Guidelines for Successful Documentation

The fear that people will do wrong blocks the opportunity to help them do right [BLAT84]

5.0 Introduction

To facilitate integration, some founding principles need to be established. This chapter concentrates on these principles which are in the form of a family of guidelines. The need for these guidelines, their nature and implications on the underlying documentation goals of education, motivation and communication are discussed and detailed.

5.1 The Need for Guidelines

Perhaps the greatest need is that the interactive software development effort brings together a diverse selection of individuals and professions. Daniel notes "problems can occur that decrease the effectiveness of the team's work at hand" [DANM87]. Oram adds that an understanding of the system is necessary before one can begin to understand how to produce a manual for that system [ORAA91]. Because of these problems and the need for their understanding and resolution, a framework is necessary. Daniel adds "[development] teams should ascribe to principles that will lead to their greatest chance of success" [DANM87].

5.2 Fundamental Principles

A survey of the literature reveals five fundamental principles or guidelines to documentation that supersede all other considerations. These are:

- **know yourself (how you write)**
- **know the system**
- **evaluate with the eyes of the user**
- **acknowledge the web/wheel nature of the guideline family**
- **consistent application of the guidelines**

The first of these concepts seems overly simple, but it is crucial to developing quality documentation. Documentors should be aware of preconceived ideas that may bias their viewpoint. Specifically they must also be aware of their writing style, their strengths and weaknesses. This is necessary in order to best apply their skills to the given situation. Such awareness is the product of experience. Daniel agrees with the idea noting that her experience as a technical writer in an integrated effort "enhanced her professional growth" and revealed new perspectives as to her abilities [DANM87].

There is also a general consensus that an awareness of the system is crucial for good documentation [BEHL80] [VOGH87] [GRIS81] [ORAA86] [ORAA91]. This principle is both a consequence and a reason for early integration and iteration of both the system and document development efforts. Such familiarity enables documents to accurately address user needs, tasks and how they may be achieved with the given system.

Complementing an acquaintance with system design is the ability and necessity for documentors to, as Sohr states, "evaluate with the eyes of the user" [SOHD83]. A wealth of research supports the need for user centeredness [MARP84] [PARS86] [HUSK88] [FOWS86] [DUIA89] [SHNB82] [MOYJ82] [VOGH87]. The driving force during development should be that the user, not the software, should dictate document organization [ROSS86]. In other words, the goals and needs of the user, not the functionality of the system should drive documentation efforts.

The next principle applies to the whole of the guideline family. Horton described the web structure of dynamic links as a possible form of document presentation [HORW89]. This hypertext organization is also the best way to represent the guideline family. The guidelines are a

synthesis of a vast bulk of material from a variety of disciplines. Each is interrelated and dependent on the others in some way. It is the delicate balance of these criteria that leads to quality documentation and constitutes the artistic quality of technical communication. This interrelated nature is evident in the problem of maximizing retrieval and learning characteristics discussed in chapter three.

Finally, the last principle deals with consistency. There is a danger that different terms and/or interpretations may arise for the same task or concept within development. This practice is evidenced by Fisher during an instance of integrated development. She discovered that developers were inconsistently applying terms during design [FISL88]. Dorazio adds that ideally helpful information should be, among other things, consistent [DORP88]. Dorsey adds that "it is vital to correctness of the final product that objectives are given consistent meanings and values throughout the documentation" [DORE92]. Otherwise users could be trapped in information loops within the documentation [MARP84].

5.3 The Guideline Family

Beyond the five fundamental principles, a larger family of guidelines is apparent from a review of the relevant literature. Where the fundamental principles apply regardless of the situation the guidelines are flexible - depending on the documentation goal one branch may be more relevant than another. The guideline family is broken into five main branches four dealing with the keys to good documentation (table 5.1) discussed in chapters two and three. The fifth branch concerns aspects of integration and iteration of the document and systems development processes

Table 5.1 The Guideline Family
* Indicates a key to good documentation

Style/Organization*
Users/Audience appropriateness*
Usability*
Learnability*
Integration/Iteration

5.4 Guidelines for Style/Organization

A user focused approach such as task orientation or minimalism was deemed appropriate for documentation of interactive systems (see 2.6). The choice of such a style of writing suggests several guides for organization and presentation of documentation. Walker suggests that user documentation should be concise and avoid redundancy [WALJ87]. Oram adds that legalism should be avoided [ORAA86]. Basically, information should be presented in terms the user can readily understand. Users should not have to resort to "services of scarce system gurus" [ORAA92]. An effort must be taken to provide an adequate level of detail but simple enough to facilitate usability and learnability.

Table 5.2 Style/Organizational Guidelines

Be concise
Avoid redundancy
Avoid legalism
Keep it simple
Utilize visual cues
Use headings, subheadings and indices
Alphabetize lists rather than list by frequency
Use tables rather than expository text
Balance retrieval with learnability
Maintain flexibility
Do not organize based on software
Orient towards user
Organize with short orienting paragraphs

The remaining guidelines for style relate to physical appearance of the information. For instance, the above table summarizes the guidelines for this section. By reading the passage and the table it is obvious that the table is a more

effective means of conveying information. Research has shown that tables are more compact than equivalent text, without loss of information [WALJ87]. This suggests the broader use of visual cues to aid users [WALR87] [STED86]. Examples of visual cues are the position of text, the use of headings, subheadings, high-lighting and indices to aid users in locating information and orienting themselves with the documentation [STED86] [WALR87] [DUIA89] [WALJ87] To further aid the user, short orientational paragraphs are also recommended [MIRB92]. Walker adds that lists of information should be alphabetized rather than listed by frequency of use [WALJ87]. The reason for this is that novices will not recognize the mapping and thus the visual cue will be lost on them. This has an analog in the guidelines for usability and audience appropriateness which illustrates the interrelated nature of the guideline family.

These guidelines reflect the idea of a user focused design with documentation oriented towards the user. As mentioned before, the software should not dictate document organization. Text should reflect this style by being more congenial, but not of a personal or chatty tone [RUBB86]. Partridge notes that occasions may arise where it is easier for the user to reference a single command rather than a whole task [PARS86]. For example, the user may partially know what is needed to complete a task, but other details elude them. Being able to reference a single command or subtask may help them. Also, a user may want to know the function of a command which may or may not be related to their current task. This reflects the limits of short term memory discussed in chapter three. This further illustrates the guidelines flexibility and must be used in the context of the situation.

5.5 Guidelines for Users/Audience Appropriateness

Walker states that "people differ in their goals, level of expertise, style of use and learning, and their preferences" [WALJ87] (see section 3.1). In a user centered development effort, knowledge of the product audience (the users) is crucial. Audience awareness, like product awareness (knowing the system) is facilitated by early integration of the documentation effort with interactive system development.

Table 5.3 Guidelines for Audience Appropriateness

Account for individual differences
Account for learning/mastery of system
Develop user profile first
Do not resort to stereotypes
Interact w/user before/during design
Know your audience

Rosenbaum and Walters [ROSS86] suggest the following provisions to help account for audience diversity -

- **start by determining what each audience wants to answer with regard to the software**
- **different groups have differing goals and priorities**
- **orient writing towards one target audience, but acknowledge hidden audiences**
- **pleasing some users alienates others.**
- **documentation should allow for learning**
- **share audience assumptions with user**

These provisions require a thorough user analysis that should be done early in design. Oram argues for greater attention to the different levels of system use [ORAA86]. Such attention is necessary because misconceptions of the user may complicate and propagate through the design, possibly until testing or release.

From personal experience, stereotypes, predictions and generalizations do not equate with actual user needs. In designing a VCR interface for a group project in a graduate course in Human-Computer Interaction, articles, past

analysis, personal preference and technician field experience formed the basis of the initial user analysis. A more comprehensive analysis was not possible due to time and monetary constraints. During usability testing, some of the materials and interface features the analysis predicted as profound did not map as favorably with users. This supports the idea that a user audience must be identified early, and document and systems developers must closely interact and observe that audience before, during and after design.

Huston and Southard concur with this idea by establishing that user needs and purposes, as well as determination of user tasks, should be a crucial part of document planning [HUSK88]. Gould and Lewis similarly agree that stereotyping should be avoided [GOUJ85]. These analyses are ultimately assumptions about the users and use of the system. Rosenbaum and Walters suggest that these assumption should be revealed to the user to aid in learning [ROSS86].

Similar to individual differences is the idea of mastery and learning. Specific guides to aid in learnability are presented in section 5.7. However the rate at which people learn and the width and depth of that learning do constitute a measurable individual difference. Along these lines, Rosenbaum and Walters suggest that documentors should allow for a learning curve [ROSS86]. As revealed in chapter three, users are active and learn at different rates. Because of this aspect of individuality, hidden user audiences may develop. Regardless of the level of expertise and experience, Rosenbaum and Walters suggest that these users still need the information presented [WALR87]. This suggests the need for visual cues and alludes to the delicate balance of retrieval and learnability (see 5.7).

5.6 Guidelines for Usability

Document usability is defined as the ease that users may achieve their objectives in reference to the written material [GOON87]. Usable documentation should provide easy to find information, easy to understand information and task sufficient information [BETF91b]. Inherent in these ideas is again the interconnected nature of the guideline family. Goodwin supports this idea by stating that usable design requires a knowledge of intended users [GOON87]. From the literature several guidelines to the production of usable texts become apparent, shown in table 5.4.

One of the keys essential for producing usable documentation that meets the goals of education, motivation and communication is the use of concrete details/examples. McDonald notes that documentors take great pains in the use of general statements but readers often have specific statements that need execution [McDM86]. Often, as detailed in chapters two and three, examples are incomplete, too few in number and frustrate novice users.

Table 5.4 Guidelines for Usability

Give concrete details/examples
Provide example closure
Provide essential cross referencing
Eliminate program internals
Stress essential features
Provide a task-oriented index
Communicate all assumptions to user
Account for incorrect user assumptions
List startup information soon/linearly
Separate instruction/exposition
Account for user/system errors

A good example of poor documentation is the syntax of PROC PRINT; a procedure in the SAS package as shown in Dilorio's *SAS Programming Language: A Gentle introduction* as seen in figure 5.1.

```
PROC PRINT [DATA = dataset] [N] [U] [D] [LABEL]
[SPLIT = 'char'] [NOOBS];
[ID idvars;]
[SUM sumvars;]
[SUMBY sumvar;]
[PAGEBY pagebyvar;]
```

Figure 5.1 The PROC PRINT procedure [DILF91]

The example is followed by 2 sample usage sans results and over a page of detailed expository text. For active users, such an example is like a brick wall in front of a speeding car - the car stops or crashes. Fortunately, Dilorio also provides four rich examples showing a sample PROC PRINT statement, the output of that statement and a detailed explanation of that output. Rich example sets reveal not the structure and form of the system, but practical application of real user tasks. McDonald adds that such examples help users as they provide a platform to meet their own needs by noting the similarities and differences between their task and the most relevant example [McDM86].

Similarly, examples should provide a sense of closure or completion. This partially violates the minimalist principle of using the situation for learning [CARJ88a] [CARJ85a]. However, creation of 'open ended' examples can be difficult [HORR92] and can lead to gaps in knowledge [BROR90]. The bulk of examples should provide a complete view of a task and its outcome. Open-ended exercises, if used at all, should be included (and stated) only as optional exercises.

Part of the user's nature, as O'Malley observes, is the possibility of a huge gap in the user's conception of information they need and what information is actually necessary to meet their goals [OMAC88]. Paramount to these goals is cross-referencing [WALJ87]. One way of achieving this goal is via the task-oriented index. Martin describes

the task-oriented index as "the best attempt to describe the uses of the system in terms familiar to real world people" [MARP86]. Such an index should account for synonyms for specific user tasks. For example, one user describes deleting a file as erasing it but another may describe the operation as destroying the file. In this instance, then all three terms should be placed in the task-oriented index [MARP86]. However, for the sake of consistency, only one term should be used within the text of the documentation.

Documentation should contain only that information which is necessary for the user to achieve their goals. Oram observes that product internals should be removed from documentation and greater emphasis placed on essential product features [ORAA86]. Inclusion of internals makes documents appear more legalistic and garbled. One difficulty with systems orientation was the minutiae that plague such documents (see chapter 2). An example of this type of information detail is a person can learn to drive a car without knowing how to replace a carburetor. Huston and Southard state that instruction and exposition should clearly be separated to avoid confusion [HUSK88]. Minimalist philosophy eliminates most, if not all, expository and orientational material [CARJ88a] [CARJ88b] [BROJ90]. However, Duin suggests that start-up and task critical information should be listed at the beginning of documents [DUIA89]. Huston and Southard feel that such information should also be linearly presented [HUSK88]. This parallels idea of short orientational paragraphs. The detail of expository information depends on the style selected, the individual system and the audience. This conforms to the artistic quality of documentation and flexible nature of the guidelines.

Documentation is an integral part of the user interface. Hooper alludes that the interface is a facade

which presents a front between the inner world of the machine and the outer world of the user [HOOP88]. As such, "the overall structure should be apparent to the user" [MARP84]. The textual content should be congenial without being too personified [MARP84] and audience assumptions should be relayed to that audience. Users may also have incorrect assumptions about the system; such assumptions must be cleared away (chapter three). This is partially accomplished by adhering to the guideline for style and audience but it must be consciously and vigilantly applied.

System Messages		
Should Not Be	~	Should Be
-Wordy	~	-Brief
-Negative tone	~	-Positive
-Critical of errors	~	-Constructive
-General	~	-Specific
-Cryptic	~	-Comprehensible
Suggest System Control of User	~	Emphasize User Control

Figure 5.2 Shniederman Message Strategies [SHNB82].

Designing for error is a particularly challenging prospect. If improperly managed, Lewis and Norman describe error messages and correction as offensive and frightening [LEWC88]. Some applicable interface research [SHNB82] may be applied to the problem. Figure 5.2 shows possible strategies for dealing with error. These strategies produce much more usable and learnable documentation than otherwise possible.

5.7 Guidelines for Learnability

Chapters two and three define learnability as the acquisition of new knowledge. Learnability is essential in the production of quality documentation. When systems are designed in such a way that learning is confounded then there is no motivation for continued use. Learnability guidelines range from organizational to content and

psychological. The guidelines are listed in table 5.5.

Table 5.5 Guidelines for Learnability

Apply 'chunking' theory
Teach how to do it
Teach why to do it
Organize similar information similarly
Use prior knowledge to advantage
Classify tasks/write accordingly
Stress key information with lists/diagrams /keywords/hi-lights
Key learning with appropriate questions
Present deductive frameworks - general to specific
Avoid/minimize repetition

First, there is a need to utilize 'chunking' in the organization of information. Miller defines what is now called 'chunking' as a limit to human short term memory of five to nine items at any given instance [MILG56]. Norman reports that short term memory in some instances consists of as few as five slots and each item in the slots decay with a half-life of fifteen seconds [NORD88]. The central concern involves limiting section content to within the 'chunking' limit and to limit page content to no more than nine paragraphs and figures to avoid user information overload.

Carroll and others have shown that due to the active nature of the user, it is not practical to dictate by rote specific actions. A better approach is to teach why to do it and how to do it [CARJ88a] [CARJ85a] [CARJ84]. Because information is presented in a understanding context it is more aptly recalled and lasting. Such background information will provide users knowledge of what the system is capable of accomplishing. Should an error occur, the information will give the user an edge in recovery.

To aid in understanding and learning, several strategies present themselves. Lists and diagrams serve to outline key points and encourage learning [DUIA89]. To avoid confusion and help with 'chunking' of information, Huston

and Southard suggest that similar information be organized similarly [HUSK88]. Martin asserts that nomenclature and syntax should be consistently applied and that users are not trapped in information loops [MARP84]. Houghton has stated "solid blocks of texts are neither helpful nor reassuring to the user" [HOUR84]. Information should be recognizable at a glance and at a minimal level of complexity (reading level) [HOUR84]. Further, keywords and questions may be used to stimulate learning and motivate use [DUIA89].

The above strategies aid in 'chunking' of new knowledge and act as a visual cue to stimulate usage of prior knowledge. When used in a user focused approach that classifies tasks and writes the text accordingly, learning is encouraged. However, use of repetition which, as DUIN notes, can help learning, should be carefully gauged and monitored [DUIA89]. Carroll observed that such repetition is frustrating and can counter learning after a certain level [CARJ88b] [CARJ84]. Finally, Huston and Southard suggest the use of deductive frameworks that proceed from the general to the specific [HUSK88]. This corresponds to the idea of specific examples discussed in 5.6 and adds credence to the integrated nature of the guideline family.

5.8 Guidelines for Integration/Iteration

The underlying concepts behind integration and iteration of both document and interactive systems development were detailed in chapter four. The guidelines for iteration are found in section 4.2.3. Integration is possible due to the close cohesion that is apparent in document and interactive system design and the consistent application of the rest of the guideline family.

Table 5.6 Guidelines for Integration

Have same user analysis for both
Acknowledge user component of documentation and interface
Coordinate schedules
Establish communication
Establish common frames of reference
Begin document development early

First, user documentation development must begin early in design. User centered methodologies demand a high degree of task sufficiency that is only possible with early and consistent involvement [VOGH87] [FISL88]. This naturally requires that one comprehensive user analysis be done for both document and systems design efforts. This is necessary due to the need to assure consistency among the development. Guillemette notes that in this manner documentation can reveal design flaws [GUIR89]. Reciprocally, design can also show faults in documentation [SODC85]. Since diverse groups are working together there is a need to coordinate schedules of not only meetings but deliverables as well. Lines of communication and a common frame of reference must be established, by user analyses and application of the UAN technique to the guideline family (see in chapter six).

When working in an integrated environment the document is integral to the user interface. This is crucial because, avoidance and resistance are key barriers to integration (see 4.1.3). The historical roles of documentor and designer must be avoided to best allow the usage of the unique and valuable contributions both fields offer.

5.9 Implications of the Guideline Family

The guideline family attempts to coalesce diverse information and research into a firm, cohesive unit. This family of guidelines are interrelated with aspects of each branch affecting and affected by the remaining branches.

Table 5.7 Implications of the Guidelines

Provides a framework for skeletal documentation
Allows innovation via flexible nature
Encourages the artistic quality of document design
Provides additional quality control
Allows new perspectives to development

An Implication for documentation is an initial framework from which to base skeletal documentation. The guides are not rigid and do not lessen the artistic quality of documentation development. Carroll suggests that it isn't possible to 'crank out' manuals [CARJ88a] [CARJ84]. Also, the guidelines, by integrating development, allow additional checks for document quality not possible under segregated development.

The project team is enriched by application of the guidelines. System designers are exposed to a wealth of material and knowledge from a largely ignored perspective. Technical writers are placed one step closer to the user and the machine at the same time. By getting involved early and working from a common framework insights and details not possible in any other manner are realized. Three sample cases in chapter four barely begin to reveal the benefits of the guidelines. Efforts must be made to foster cooperation and communication to maximize these new opportunities.

5.10 Summary

Systems have been produced under a semi-rigid framework for at least three decades. Most technical writing shops also have standard operating plans. The guidelines act as a Rosetta stone of development that unites design of documentation and interactive systems along lines of audience, usability, learnability and iteration. This commonalty opens up new worlds for cooperative development.

Chapter 6: User Action Notation: A Technique, a Solution

Communication requires Analysis of Thought - [HORR89]

6.0 Introduction

This chapter describes User Action Notation and applies the guideline family to that notation. This design technique is used to illustrate and facilitate the integration of both forms of development and provide the means for construction of skeletal documentation from user/task specifications.

6.1 The User Action Notation defined

Hartson, Siochi and Hix note that "software engineering methods still do not necessarily produce user interfaces with high usability" [HARH90]. Furthermore, "developers know better how to construct a system than how to specify what it is to accomplish and how it is to interact with the user" [HARH90]. This view corresponds to the difficulties of system oriented documentation discussed in chapter two. Hix and Chase add that "almost all existing user interface representation techniques are constructional, focused on interface implementation and therefore do not adequately support a user centered focus" [HIXD92a]. The User Action Notation (UAN) fulfills that role of a behavioral interface representation technique. "The UAN is a user- and task-oriented notation that describes physical (and other) behavior of the user and the interface as they perform a task together. The primary abstraction of the UAN is a User task" [HARH92]. This user centered technique provides a bridge between user analysis and interface design. Thus UAN is the technique of choice to visualize the integration of development.

6.2 The Behavioral and Constructional Domains.

Section 6.1 alludes to behavioral and constructional domains. In terms of documentation, these domains impact mental models. The constructional domain is analogous to the system designer's internal model of the system. Martin notes the reliance of that internal model in systems oriented material results in only marginally usable manuals [MARP86]. In contrast, the behavioral view is more analogous to user centered documentation styles such as task orientation and minimalism which pattern after the user's internal model of the system.

Constructional techniques describe from the view of the system. "Constructional representation techniques support the designer and the implementer of the interface software but do not support design of the interaction part of the interface itself" [HARH92]. Chapters two and three show that documentation is key to the interface, especially what the user sees and interprets. Thus, systems and documents that ascribe solely to constructional techniques have difficulties in style, appropriateness, usability and learnability. Hartson and Gray note that the use of constructional techniques can yield descriptions that are "unusable - overwhelmingly large and complex" [HARH92]. This corresponds with the deficiencies of user documentation discussed in chapter one.

The behavioral domain allows consideration of the user's viewpoint. It is in this area that "developers of the interaction part of an interface (e.g. interaction designers and evaluators) do their work" [HARH92]. The documentor works within this domain as well. The behavioral domain concerns processes that influence, precede and input software design, including task analysis, functional analysis, task allocation and user modeling [HARH90]. These processes also greatly affect documentation development.

6.3 Justification of UAN

"The people who are in various development roles must have a representation of the interface design to do their work" [HARH90]. The quality of design depends on the capacity of the diverse design group to understand, communicate and evaluate design during development [HARH90] [HARH92]. Documentors must also have an acute awareness of audience and design. UAN, by its nature, is an excellent choice for capturing the representation of behavioral design as created and communicating that design to implementers [HARH90]. Additionally, the UAN is an open, extendible notation which is readable with little instruction. Thus, UAN can be used by a number of diverse developers and researchers [PARM92] [HARH92][HIXD92a].

6.4 Sample UAN

For more complete discussion of UAN, Hartson et al [HARH92] [HARH90] [HIXD92a] should be consulted. However, some simple examples are necessary. The following samples are adapted and derived from Hartson, Siochi and Hix [HARH90].

Example 6.1: A simple example involving the Macintosh interface and showing only the user action.

Task Description:

(1) Move the cursor to the file_icon
(2) Depress and immediately release the mouse button

UAN Description:

TASK: select_icon
USER ACTION
(1) ~[icon]
(2) M$\vee\wedge$

Analysis:

(1) move the cursor (~) to the context of [icon]
(2) depress ($\vee$) release ($\wedge$) the mouse button (M)

Notice that the same information content is maintained in both the task description and the UAN notation. However, feedback, an important aspect of Human-Computer interaction, is missing.

Example 6.2: example 6.1 incorporating feedback.

Task Description:

(1) Move the cursor to the icon

(2) Click the mouse button and the file will be highlighted

UAN Description:

TASK: select_icon	
USER ACTION	Interface Feedback
(1) ~[icon]M$^{\vee}$	icon!
(2) M$^{\wedge}$	

Analysis:

(1) move the cursor (~) to the context of [icon] depress ($\vee$) the mouse button (M). The icon is now highlighted to indicate selection

(2) release ($\wedge$) the mouse button (M)

To be still more precise, the mutually exclusive nature of highlighting an icon may also be represented.

Example 6.3: showing conditional information in feedback

Interface Feedback
(1) icon -!: icon!,
(2) $\forall$ icon'!: icon'-!

Analysis:

(1) highlighting (!) is applied to an icon iff it is not already highlighted

(2) For all ($\forall$) icons that are highlighted (!) those icons are consequently un-highlighted (-!).

Highlighting here is visual feedback for selection.

The degree of detail is not necessary, but is available should developers wish to utilize it.

Additionally, UAN provides a mechanism to illustrate how individual user actions relate to the interface state and specific connections to computation.

EXAMPLE 6.4 UAN showing interface state and connections to computation.

TASK: select_icon			
USER ACTION	**Interface Feedback**	**Interface State**	**Connections to Computation**
~[icon]M^	icon -!: icon!, ∀ icon'!: icon'-!	1)selected = icon	
2) ~(x,y)*~(x',y')	3)outline(icon)>~		
M^	4)@x',y' display(icon)		5)location(icon) = x',y'

Analysis:

1) interface state: icon selected
2) move cursor from (x,y) via an undetermined path (*) to (x',y')
3) an outline of the icon follows (>) the cursor
4) at x',y' display the icon image
5) the system records the new location of icon at x',y'

Conditions of viability, such as files that must be de selected to be selected, may also be placed on user actions. UAN tasks may be grouped together to form larger task descriptions. An example is the various subtasks involving file manipulation could be grouped under the greater task of file manipulation. File manipulation may again be grouped in another collective. This 'quasi hierarchical' stacking [HARH92] [CHAJ92] [HIXD92a] [HIXD92b] [PARM92] is highly important to the development of skeletal documentation and is discussed in detail in section 6.6. Additionally, UAN may represent temporal aspects, such as interleaving of tasks [HARH92], or even be extended to address user cognitive processes and memory [HIXD92a] [CHAJ92]. The notation is flexible enough to meet a variety of developmental needs, yet rigid enough to provide a basis of communication between diverse groups.

6.5 UAN and Documentation Development

This section addresses how UAN fits into an integrated iterative framework for interactive systems development to facilitate skeletal documentation production. The framework is shown in figure 6.1. There are two dimensions to the framework. The framework is divided horizontally by degree of fluidity in design. The framework is divided vertically into bands that represent the iterative software engineering process (ISE), integrating processes and the documenting process. This section concerns the later two processes.

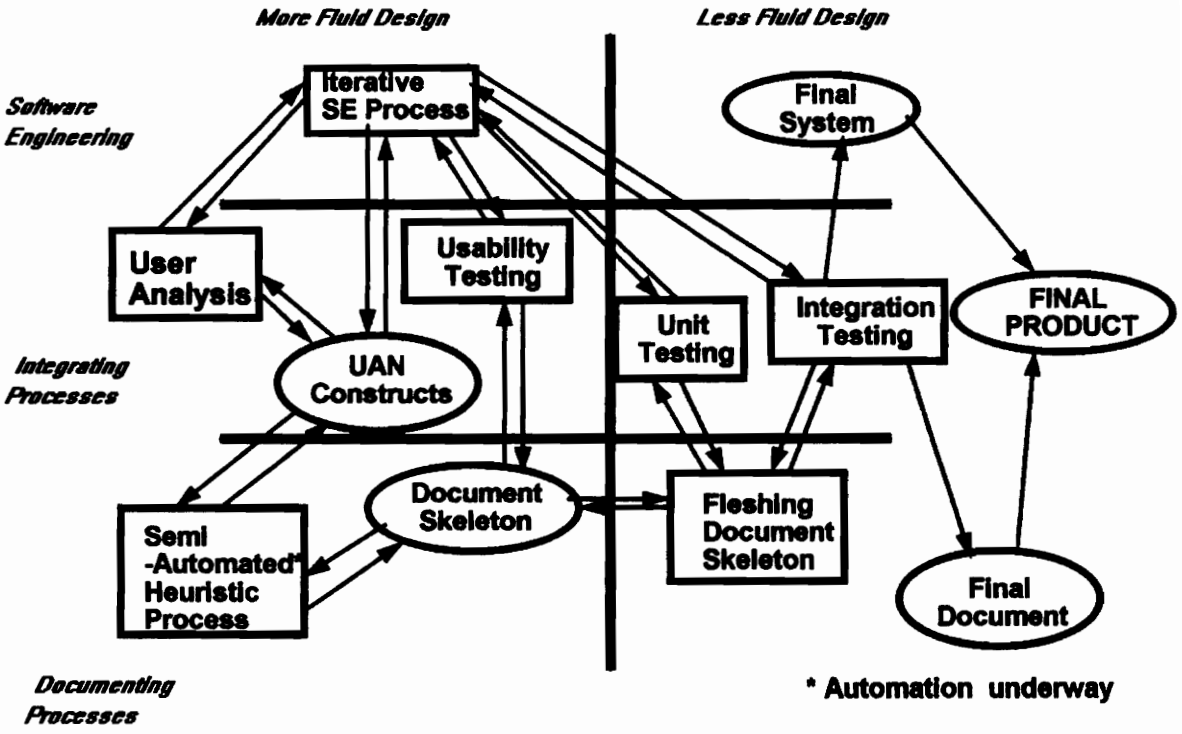

Figure 6.1 UAN and Integrated Documentation Development

6.5.1 Organization of the Developmental Framework

The framework in Figure 6.1 produces skeletal documentation that is integrated and reconciled with interactive systems development. To the left of the vertical is the more rapid, dynamic iterative processes that are

subject to frequent change early in design. The right side represents slower more predictable change, such as the 'fleshing' of the skeletal document into final form.

For simplicity, Figure 6.1 does not show all possible arcs, only the more general flow of design. Usability testing, for example, early in the design process, may lead to major alterations in the user analysis, UAN, interactive software and/or the document skeleton. The fluidity is due to rapid changes that occur during iterative design. In this manner, iteration confronts and meets the needs of a changing audience [GOUJ85].

6.5.2 The Integrating Processes

Integrating processes enable consistent, simultaneous production of interactive systems and their documentation. Integration begins with construction of a shared user analysis that serves as input to ISE and UAN task construction. The UAN then acts as a common database for producing interface prototypes and skeletal documentation. Usability tests are conducted with the ISE prototype and document skeleton as inputs. This enables prototypes and skeletons to act as testbeds for each other.

As the process slows, or units of the system become complete, unit testing determines whether the portion of the interactive system and the respective documentation are consistent with each other. Unit testing differs from usability testing in that a 'fleshed' document and 'evolved' prototype are inputs. Impetus is on how well the entities complement each other on a task level.

Upon completion of at least two units, integration testing begins. Similar to unit testing, integration tests assure that systems and documents form a cohesive unit. Completion of this testing yields a comprehensive package or final product. Impetus is on how the parts form the whole.

6.5.3 Roles in the Integrating Processes

Constructing user analyses involves; prospective users, interface designers and documentors. Users are an obvious choice, interface designers are needed to "make sure that design of both input and output take users' needs into consideration [SANC91]. Technical writers are included because they also work in this domain (see chapter four).

UAN task descriptions require an initial agreement to be reached on any enhancements to standard UAN. Interface designers then build the UAN descriptions. If UAN is not stored in a hypertext format, then a documentor needs to organize the UAN into a form that will enable construction of a task oriented outline [PARM92] (see section 6.6).

Usability tests, Unit tests and Integration tests require participation of interface designers, system designers (those developing the prototype) and documentors. Potential users are also needed for testing purposes. The remainder are included to bring those involved together, provide cross channel communication and feedback. This assures consistency in the development effort (see chapter five).

6.5.4 Heuristic Documenting Strategies with UAN

UAN task descriptions act as an input to a set of heuristic processes which can and should be automated, appendix B lists some suggestions regarding automation. The Heuristics produce skeletal documentation and accompanying filler (examples) from UAN descriptions. Sample documentation created following these heuristics is located in section 6.6. The rules are:

- UAN symbols and tasks must consistently map to text.
- A UAN row with a distinct user action is one complete step w/feedback in documentation. i.e. | (or) indicates possible path not a distinct action.
- User action columns translate to descriptions of user's roles in a given task.
- Interface feedback columns translate into the interface's role (i.e., what the users sees as a result of her action).
- Interface state columns translate to supplemental information regarding the consequence of the user action.
- Connections to Computation columns suggest system context information that maybe useful in clarifying task objectives (i.e., "Does this do what I think it does?").
- Conditional actions, aborts, error recovery and 'undo' actions should be listed after the task or separately, in a manner consistent with these strategies.
- UAN task names = section headings
- Frequent subtasks should be named as separate UAN tasks and constitute startup information.
- User action columns, interface state columns and connections to computation columns yield minimal texts.
- Examples are created by substituting 'real' objects for object names (i.e. [file_icon] = "My Life's Work").
- Example sets are produced by creating separate examples of the various means and consequences of performing the same task (i.e. various ways to delete files).
- UAN based examples and skeletal documentation serve as inputs to prototype usability testing and vice versa.
- UAN task names should be determined by user analysis.
- Names used in task oriented indices are determined by usability testing.

6.5.5 'Fleshing' Document Skeletons

The document skeleton serves as an input to usability testing. As the process reaches conclusion or sections of the document skeleton become firm, the skeleton serves as an input to the final 'fleshing' of the documentation. At this point, traditional guides and strategies for technical writing supersede the UAN Heuristics. This constitutes an artistic quality of documentation that cannot be systemized (see chapters three, four, and five).

6.5.6 Roles in the Documenting Processes

Documentors hold the key responsibility during the construction of skeletal documentation once UAN task descriptions have been prepared. Depending upon the degree of automation and information available, interface designers and possibly system designers may be required/helpful. Interface designers may be needed to explain the UAN symbology or help organize the task hierarchy. System designers may be needed to clarify the meaning of some connections to computation (see section 6.6). Traditional 'frozen' documents have been unable to keep up with the fluidity in iterative design [PARM92] [GUIR89], but these heuristic strategies minimize that difficulty.

'Fleshing' the document requires the expertise of the documentor. During unit and integration testing, interface designers, system designers and prospective users provide additional feedback to assure consistency, sections 6.6 and 6.7 further illustrate these ideas.

6.6 UAN as Applied to the Guideline Family

Chapter five described a family of five groups of guidelines and five superseding principles, summarized in table 6.1.

Table 6.1 The guides to quality documentation

Superseding Principles	Guideline Family
Know yourself (how you write)	Style/Organization
Know the system	Users/Audience Appropriateness
Evaluate with the eyes of the user	Usability
Acknowledge related web nature of guidelines	Learnability
Consistency	Integration/Iteration

The UAN, due to its behavioral nature is ideally applicable to the fruition of these guidelines. The UAN unites both documentation and interactive systems development with a common language, frame of reference and purpose.

6.6.1 UAN & The Fundamental Superseding Principles

KNOW YOURSELF (HOW YOU WRITE). Here the UAN does not directly impact documentation development. However the detailed design inherent in the UAN does free the documentor to concentrate more on aesthetic details and writing style. This is the wild card of documentation development, the artistic quality that prohibits the systematic construction of manuals. Paretti observed that the UAN helped in organization but the content was still her own [PARM92]. "Communication requires analysis of thought; it requires understanding and leadership" [HORW89]. Thus, UAN should be a technique at the writer's disposal, not a technique that disposes of the writer.

KNOW THE SYSTEM. One of the greatest problems in documentation has been incomplete, inaccurate representation of information [TOMR82][GLUR82][CLAF83]. UAN is prepared early in interface design, the user analysis from which it is based forms the cornerstone of interface design. By getting involved early with the production of UAN, the documentor becomes intimate with the task structure of the system and thus there is less chance of error or

inconsistency within document/interactive system design [PARM92].

EVALUATE WITH THE EYES OF THE USER [SOHD83]. Impetus for development should stem from the user, not the system. Because UAN is a behavioral representation technique, it is aptly suited to fill this need. "Behavioral descriptions can be thought of as procedures executed by the user" [HARH90]. UAN captures this information as well as how the user and interface react to each other. This forces design to be seen from the eyes of the user, which is the first step to evaluating the system from a user orientation.

ACKNOWLEDGE THE RELATED WEB NATURE OF THE GUIDELINES. The Guidelines are interrelated, maximize one principle and another principle suffers. This forms a web of interconnected information and constitutes an artistic aspect of documentation. Since the notation is 'quasi-hierarchical', UAN exhibits some of these characteristics [CHAJ92] [HIXD92a]. A subtask such as SELECT_ICON can be part of many larger tasks (i.e. deleting a file, launching a program) or a task unto itself. Similarly the subtask is composed of individual user actions such as depress mouse button (M^{V}). In this sense, the web structure of the UAN can serve as a metaphor and reminder of the web structure of the guides.

CONSISTENCY. The web structure is a primary reason for the need for consistency of information within all aspects of development. The complexity of the web causes a great deal of difficulty. "Maintaining integrity between the user manual and the implemented system is a common problem" [HARH90]. Dorsey identifies consistency as "the invariant application of definitions, values and concepts within the documentation" [DORE92]. Because the UAN captures the nuances of interaction [HARH90][HARH92], it is ideally suited to foster a certain degree of consistency. Since both

documentation and interactive software both evolve from the UAN and a common user analysis both draw upon the same core material. This helps to produce consistent application of information during development. However, any modifications or enhancements to the UAN must be agreed upon by the entire team to ensure understanding and consistency of application. Sections 6.6.2, 6.6.4 and 6.6.5 further elaborate on this crucial principle.

6.6.2 UAN and Documentation Style.

Consider the example of the task of selecting a choice from a menu, adapted and extended from [HIXD92b].

EXAMPLE 6.5: MENU SELECTION

TASK: Select (choice) from (baritem)menu		
USER ACTION	Interface feedback	Interface State
~[baritem]M$^{\vee}$	baritem! display(baritem_menu)	
(~[choice]	choice!	
([choice]~	choice-!	
~[choice'])	choice'!	
[choice']~	choice'-!	
~[choice'']	choice''!	
M^)	choice''!! erase(baritem_menu)	result = choice''
|		
(~ OFF (baritem_menu)	∃[choice]!:[choice]-!	
M^)	erase(baritem_menu)	result = null

Example 6.5 shows the promise UAN has towards fulfilling the role of a documentation aid. In accordance with the guidelines for style the UAN task description is concise, there is no redundancy. Only information specific to the task is presented. Thus, the description is kept simple and avoids legalism. These aspects carry forward to the

documentation. The skeletal documentation from Example 6.5 as seen in Figure 6.2 reflects these principles.

```
To select a choice from a bar menu do the following:
      1.  Using the mouse, move the cursor to the desired bar item,
      depress and hold the mouse button.
           The bar item selected will highlight and the bar item menu
      will be displayed listing several choices.
      2.  To Select a choice, using the mouse, move the cursor to
      your choice.
           Notice as the cursor moves up and down the bar  item menu,
      only the current choice under the cursor is  highlighted.
      3.  When your choice is highlighted, release the  mouse button
           Your choice will now 'depress' on the screen and the bar item
      menu will erase. This indicates that your choice has been recorded.
      4.  If the incorrect bar item was selected or you wish to make
      no choice, simply move the cursor off the bar item menu and
      release the mouse button.
           If a choice was highlighted it will return to normal when
      the cursor leaves the bar item menu. When the mouse button is
      released the bar menu is erased. No choice will be recorded
```

Figure 6.2 Skeletal Documentation for a bar menu

Notice that the text is oriented even at this stage to the user. The text is also broken into short orienting paragraphs that deal with specific user actions and their resulting feedback. The skeleton is flexible and can be modified to represent any bar menu or selection from that menu. Additionally the skeleton may be enhanced with visual cues such as screen pictures and sample interactions.

Where possible, tables and lists should be used rather than expository text. As in Figure 6.2, the task structure of UAN easily facilitates construction of such tables.

Though not in the example, judicious use of headings, subheadings and indices must also be used. These are derived from main UAN task names. Paretti observed that 80% - 90% of the UAN task hierarchy translated directly to her document outline [PARM92].

The guidelines also impact the UAN. Documentation should use alphabetized lists wherever possible as opposed

to listings by frequency. For consistency this must be applied to the design and by extension the UAN descriptions.

The need for flexibility in documentation is mirrored by the UAN. The notation is easily extensible. The previous example illustrates two extensions to the UAN. First the boundary condition of OFF is used to clarify the user action of leaving the bar menu. The concept of boundary condition are not built into the UAN [HIXD92b] Such actions could consist of:

- OFF - leaving a screen object
- ON - entering a screen object
- NEAR- approaching a screen object
  etc.

Next, to clarify feedback conditions, the existence symbol (∃) is used. The 'for all' symbol (∀) could also represent this concept, but in this case existence is more appropriate. UAN's flexible nature thus adapts and aids in the clarification of tasks and document design. However, to be useful, any extensions to the UAN must be agreed upon and consistently used by the design team.

6.6.3 UAN and Appropriateness

The UAN, by being a behavioral representation technique, satisfies the bulk of the appropriateness requirements. Prior to any work on the UAN, the target audience must be identified. The goals, priorities and needs of that audience dictate how the interface and consequently the documentation are designed. The UAN requires a user profile as an input to its development [HARH90]. This requires close interaction with users, analysis and avoidance of stereotyping. UAN can be developed using an audience stereotype but it will not accurately reflect the scope of individual differences nor the extent of user needs. By creating UAN based on detailed user analysis,

developers come to know the audience and the system. This awareness helps to point out enhancement and error in design of the interactive software and the documentation [PARM92]. Documentation should also account for learning and mastery.

6.6.4 UAN and Usability

Essential to usable documentation is the use of concrete details/examples. To maximize usability and enhance learning, examples should be rich, appropriate and incorporate a variety of realistic scenarios. Users then examine the samples, select the most appropriate for the current task and extrapolate proper usage from the scenario. Recall "the primary abstraction of UAN is the user task" [HARH92]. At the lowest level, UAN provides a script from which accurate examples may be written. Such examples can be used as scenario scripts to portray sample interactions for usability testing. These scripts are naturally complete and provide closure. Consider example 6.6, deleting a file on the Macintosh (adapted from [HARH90]).

Example 6.6: Deleting a file

TASK:Delete file			
USER ACTION	**Interface Feedback**	**Interface State**	**Connections to Computation**
~[file_icon]M^	file_icon-!:file_icon!, ∀ file_icon'!:file_ icon'-!	selected = file	
~(x,y)*	outline(file_icon)>~		
~[trash_icon]	outline(file_icon)>~, trash_icon!		
M^	erase(file_icon), trash_icon!!	selected = null	mark file for deletion

The UAN in Example 6.6 and skeletal documentation of Figure 6.3 may be expanded to include various techniques to select a file for deletion or the selection of multiple files for deletion. The exact meaning of "marked for

deletion" is not evident within the context of the UAN nor the skeletal script. However the step by step actions needed to delete a file, the relevant feedback and consequences are presented. The richness of the example may be improved by adding sample interaction illustrating the expanded methods of deletion as previously discussed.

Deleting the file - "My Life's work"

1 Using the mouse, select the file "My life's work". Depress and hold the mouse button.

The file "My life's work" is now highlighted, indicating selection. Notice that any previously highlighted files are no longer highlighted.

2 Using the mouse, move the cursor the to the 'trash can' on the lower right hand corner of the screen.

Notice, an outline of "My life's work" follows the cursor. When the cursor reaches the 'trash can', the 'trash can' is then highlighted.

3 Release the mouse button.

'My life's work is now removed from the desktop. Notice that the 'trash can' now 'burgeons' to indicate that "My life's work" is now in the trash. "My life's work" is now marked for deletion.

Figure 6.3 Resulting skeletal documentation from UAN example 6.6 useful as a Sample Script Scenario in Usability Testing

The "marked for deletion" is not identified by the UAN. This system oriented context must be defined in terms the user can understand. The internal components must be eliminated. The user must be told why deletion should occur and how to delete. The essential features and components that constitute this act must be stressed. Short orienting paragraphs as discussed in 6.8.2 are very useful in the delivery of such information. Such passages of crucial or startup information should be presented early and act as a road map to guide the user to desired information. Figure 6.4 illustrates an example of such an informative paragraph.

Notice that the passage in figure 6.4 is user centered, brief and that the information cannot be found completely within the UAN. This is another reason for the need for close integration. Designers have an internal model of the system that causes difficulty in documentation development

(see chapter two, three). Similarly, the external, user oriented model of the documentor can create difficulties in document design. The UAN, by creating a common frame of reference, presents a background for the development team to discover their information needs and design inadequacies.

```
DELETION:
Deletion removes files from your disk. You may want to delete a file if
any of the following occur:
                    your disk is full,
                    you need more disk room,
                    you have another copy of the file,
                    file is no longer needed, wanted or used,
                    etc.
Deletion frees up disk space to store additional information. Deletion
is accomplished by moving a file(s) to the trash. The files are now
"marked for deletion".When you shutdown the system or remove the disk
containing the file(s) - the file(s) is deleted. Examples follow.
```

Figure 6.4 Short orienting passage

To accommodate individual differences and documentation needs, there is room for example variation. The skeletal documentation for the deletion example could have been written as in Figure 6.5. In this way, UAN allows a variety of documentation to be derived from the same UAN description. The level of detail and content depend on the audience and the type of documentation being prepared. The UAN could produce skeletal scripts suitable for usability testing, reference guides, tutorials or user's guides.

```
1 Select and move the file "My life's work" to the trash
  "My life's work" is now marked for deletion
```

Figure 6.5 Alternate Skeletal Deletion Documentation

Such UAN scripts are a natural consequence of the 'quasi-hierarchical' structure of the UAN. The selection of a file is a separate subtask of, among other things, deletion. Deletion of a file is in turn a subtask of deletion of multiple files. The deletion of multiple files

is, in turn, a subtask of the more generic task of file manipulation. This aspect may be used to aid in the creation of referencing and outlines crucial to the usability of the document.

Riley notes that users "constantly set goals and must plan how to achieve those goals with available commands" [RILM88]. These form the basis of a hierarchical goal structure or planning net [RILM88], seen in Figure 6.6.

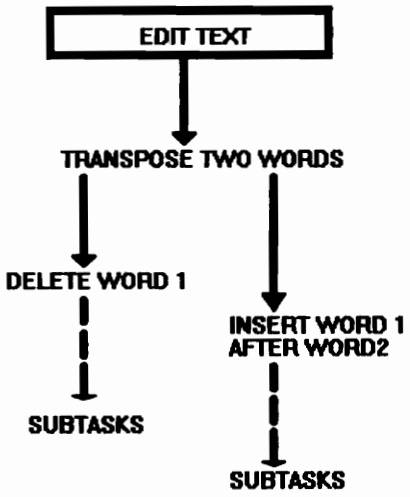

Figure 6.6 A planning net [RILM88]

Oram sees this structure as a 'connect the dots puzzle'[ORAA86]. The writer's job is to describe the multitude of task variations and how each are accomplished. The complex structure is transcribed into the system as a massive tree of static choices and interaction paths. In this sense, "design is a dynamic user process flattened into a static set of paths and nodes" [ORAA86].

The UAN exists as a complex web of task descriptions. Within most modern interactive systems, the positioning of a cursor is the same regardless of context. In the course of an interaction, a single file or a group of files may be deleted, created, modified, viewed or ignored in an indeterminate order. A partial view of this web structure is shown in figure 6.7. and corresponds to the task structure

described by Chase [CHAJ92].

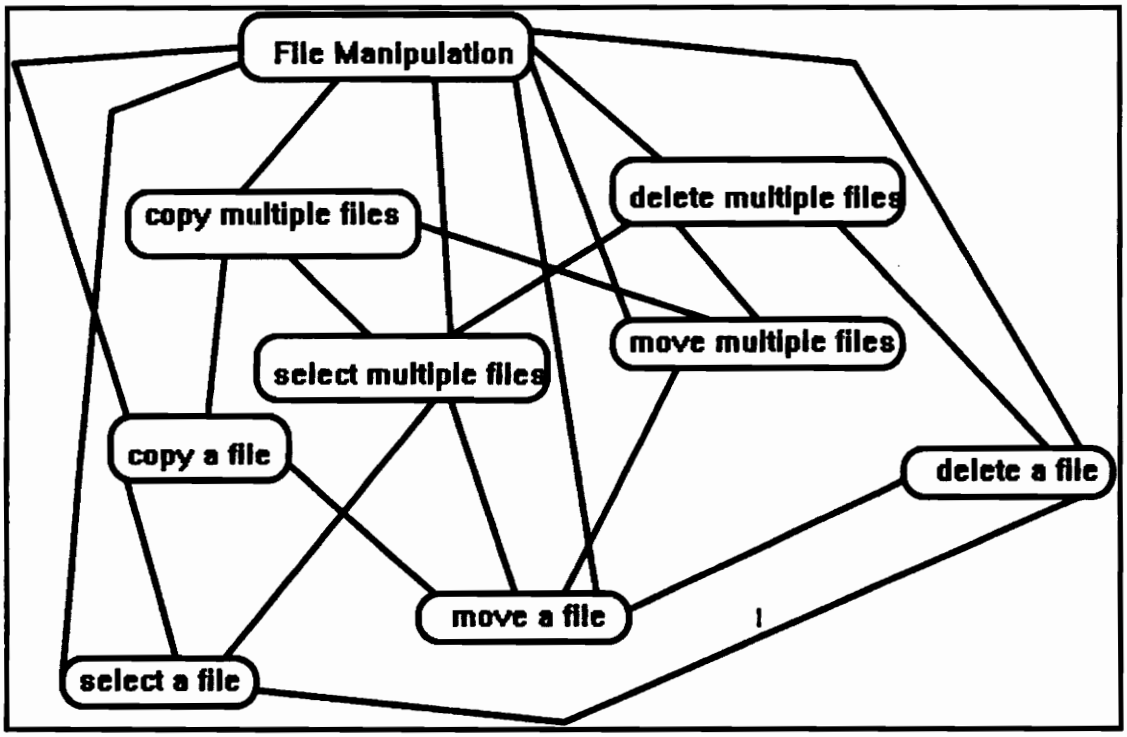

Figure 6.7 UAN partial task web for file manipulation inspired and adapted from [CHAJ92][ORAA86]

At the lower levels of the web are distinct user actions. The arcs of the web are bi-directional. A task could begin anywhere in the web and lead to any other task in that web. The UAN captures a great deal of this within the notation itself. The task hierarchy corresponds to nodes in the web. This information is a source for outline and cross-referencing information. A flattened form of this structure is visible in figure 6.8 with accompanying skeletal outline.

In Figure 6.8.a, a high level task (manipulate files) is decomposed down into distinct user actions. Detail is omitted to conserve space, but a task structure is clearly evident. From this task structure, a user oriented outline

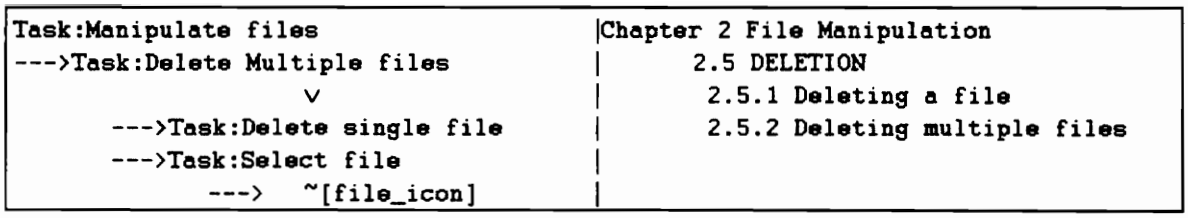

Figure 6.8.a UAN TASKS **Figure 6.8.b Skeletal Outline inspired by [PARM92][CHAJ92]**

is easily produced (figure 6.8.b). The lower level subtasks and discrete user actions then form a skeletal filler for the outline, that can now evolve into a table of contents and document skeleton. Paretti found this quite beneficial in the development of a task oriented outline [PARM92].

In addition to providing the framework for an initial outline, the foundation is also laid for a task oriented index. This index involves close interaction with both the system design and the potential users of the system. Martin notes the index should be the best attempt to describe usage in user terminology. This involves the use of synonyms to supplement indexing [MARP84]. However for consistency, within the UAN and consequently the system and the documentation, a single task name should be used. The most appropriate name is be determined by user analysis and usability testing. In keeping with the example of deletion, the user could think of deleting as trashing, killing, destroying, purging, rubbing or erasing. The most frequently referred term becomes the proper UAN task name and possibly all or part of the synonyms should be included in the task oriented index. The extent of inclusion again is determined by user testing. Within the index, the synonym would direct users by the proper task name such as 'deletion'.

Any assumptions made about the user audience should be revealed to those users (chapters 3, 5). The UAN does not directly help in this endeavor. However, the user analysis and close user involvement in interface design required by

the UAN is crucial to determining such assumptions. Thus the UAN allows a more complete understanding of the behavior of the user which aids in articulating interface design assumptions.

Similarly, incorrect user preconceptions must be dealt with during document development (chapters 3, 5). This is accomplished partly by using of short orientational passages and the task oriented index (chapter 3). The index and passages serve to guide the user to an initial point of reference. An additional strategy is use of questions to key understanding. However, occasions WILL arise when user misunderstanding is not 'intercepted' and erroneous system states will develop. These errors must be accounted for and recovered. This applies to the whole of development: documentation, UAN and interactive software. Care must be taken to ascertain that error recovery information, from UAN onward, addresses the suggestions for system messages in figure 5.2. This need for consistency in error handling is yet another reason for integration of development via UAN.

6.6.5 UAN and Learnability

Learnability relates heavily to usability and style. However there are aspects such as short term memory, and the information content that distinguish this key to good documentation. The UAN affects and is affected by this principle.

The classic aspect of learnability is "the magic number seven, plus or minus two" [MILG56]. From this line of reasoning, UAN tasks should not exceed a total of nine distinct, linear subtasks or discrete user actions. Documentation and the interface software then inherit a degree of built-in learnability. This is a logical extension of Miller [MILG56] but experimental study is needed to validate this idea.

The reason for this restriction is simple, anything above this limit will constitute an overload of information. If there are more than nine subtasks, then the task may not be properly decomposed. Notice that nonlinear tasks are not included in this restriction. These tasks are those that mutually exclude each other. For instance a single copy of a file cannot be loaded and saved simultaneously. A single user of a text editor cannot simultaneously type and delete. The MS-WORD 'save' and 'save as' actions cannot be done together, though they are variations of the generic save task. These are all potential paths that the user may take, each with distinct consequences and results. Since they cannot be done together, there is no danger of information overload. However the possible paths at any given point should be constrained to these human memory limits.

To circumvent this limit, and produce more organized design, group user tasks into larger subtasks in a logical fashion. For example, as discussed earlier the specific user action of selecting an on-screen object is the same regardless of context in most modern systems. Thus, when a task requires the selection of an object, the UAN subtask select_object may be substituted, provided that the select_object task has been defined in UAN. Within the documentation the UAN subtask select_object will become essential start-up information, presented early in the documentation. Learnability thus reflects a common denominator in both document and interactive system design and is yet another reason for integration with UAN.

Next, users should be told how to do a task and why to do it. This information is partially built into the UAN. As evident in the UAN examples, 'how to do it' information is readily available. This is due to the fact that UAN is built around the user task [HARH90] [HARH92]. 'Why to do it'

information is found in the context of user analysis and their goals with the system. As seen in section 6.7.3, the UAN indirectly supports such information by facilitating the creation of short orienting paragraphs.

For the sake of consistency, similar information should be presented similarly. This is achieved easily with the UAN. Since the basis for frequent subtasks like selection, movement or insertion are based upon common UAN task descriptions regardless of context, consistency of example and format is derived naturally.

The remainder of the guidelines concerning learnability have parallels in appropriateness and style. Learnability, like style, requires the use of lists, diagrams, keywords and highlights that minimize repetition. These document features as discussed under style are all available in one form or another via the UAN. Parallel to appropriateness, individual differences and prior knowledge are monitored and incorporated as a result of the increased user centeredness necessary in the production of the UAN.

Similar to usability, overall learnability benefits from the UAN classification of tasks and the resulting task oriented documentation. In this vein, questions can be useful to key learning. Proper questions, like the creation of the task oriented index, are determined during user analysis and usability testing. In keeping with the deletion example figure 6.9 reveals some sample questions.

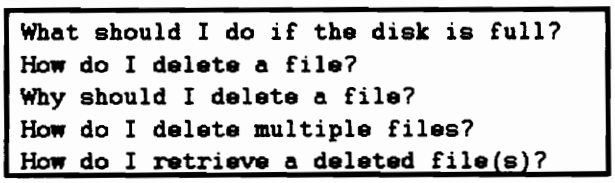

Figure 6.9 Questions to key learning

Next, deductive frameworks which proceed from the general to the specific are necessary. These frameworks are available two ways via the UAN. First, the UAN task web

presents a break down of general to specific tasks. Next, deductive frameworks can be generated much the same way as examples are generated in section 6.6.3. UAN is the user task in its most abstract form, simply stating user acts and consequent feedback and interface states. By filling in missing details, i.e., delete "My life's work" instead of delete file, a deductive framework based upon example is easily created. However, frameworks by example are not as beneficial as rich example sets and should be used sparingly. However, the UAN task web frameworks, are beneficial to document creation and overall learnability.

6.6.6 UAN and Integration/Iteration.

When used as a technique for integration, the UAN establishes a common frame of reference based on a mutually shared user analysis and common means of discussion. UAN forces a behavioral approach to design that acknowledges and stresses the importance of the user. The UAN enables early development of documentation by producing a platform for skeletal documentation. UAN enables closer developmental communication through a common means of cooperation and language. This cohesion leads to a coordination of efforts and schedules that lead to simultaneous development and release of more accurate documentation and interactive systems.

UAN easily fits into the framework of iterative development as presented in sections 4.2.3. and 6.5. The UAN is most applicable to the collection of user information, audience definition, user needs analysis and task identification. UAN is useful in the creation of skeletal documentation and usability test scenarios which facilitate both document and system design. UAN is because it captures design as it is created [HARH90]. UAN works within the behavioral domain incorporating task analysis, functional

analysis, task allocation and user modeling [HARH90] [HARH92]. UAN is more flexible than documentation and can easily be changed to reflect feedback from user testing.

However, the complex nature of the UAN web may make application of the UAN difficult at first. Paretti noted that she did not keep the UAN up-to-date with design changes [PARM92]. This is probably due to the following:

- **UAN was constructed midway through design which lost precious time**
- **There was no way to store and adequately access the resulting UAN web, making modification difficult.**

Thus, to adequately utilize UAN iteratively, the UAN must be constructed early in development and have a supporting information system to adequately store, access and maintain the UAN task web.

6.7 UAN and Education, Motivation and Communication

EDUCATION. UAN, by incorporating development, promotes not only a greater awareness of users and their needs but of the task structure as well. UAN engenders an increased familiarity that provides a solid basis for selection of metaphor (see chapter 3). Properly selected metaphors lead to information in a form that users can understand and thus learn. By making information learnable users are motivated and more apt to use the product.

MOTIVATION. Since there is an increased integrity and consistency of information with the UAN [HARH90] and skeletal documents produced with the UAN inherit these attributes; the potential exists for increased user satisfaction. When errors occur, users turn to documentation in desperation. By presenting accurate information that reflects the context and situation, the user can more easily recover from the error. Thus, user confidence in the system increases. It is not a great leap to conclude that confident

users will be motivated to use the system (see chapter3).

COMMUNICATION. In order to educate or motivate, information must be conveyed language the user can understand. Selecting a behavioral technique such as UAN as the basis for integration and production of skeletal documentation and committing to a task oriented writing style, information will be presented from the user's point of view. This enables communication with that user that opens the door to all other objectives.

6.8 Benefits of UAN

UAN is a beginning for close-knit cooperation between documentors and developers. The benefits of UAN vary from documentation, improved design to more team cooperation.

Table 6.2 UAN benefits

Fosters communication
Captures all possibilities
Simple to learn and use
Provides generic scripts for users tasks
Establishes common frame of reference
Facilitates new design insight
Enables identification of error
Increases user awareness
Promotes consistency in design

First, UAN is an excellent technique to foster design team communication. Hartson and Gray note that "experience has been that the UAN design representation technique is typically more accurate, more complete and more precise than the prototype as a source for answers to the questions that arise in a design walkthrough" [HARH92]. The reason is that UAN captures all the possibilities for interaction [HARH90] [HARH92]. Oram observes that dealing with the multitude of possibilities is the crux task of both developer and documentor [ORAA86]. UAN is also simple to learn and use by a diverse class of professions [PARM92] [HARH90]. This fact

enables the UAN to be utilized as a means for integration of document and interactive systems development. The task orientation of the UAN is what makes it so useful to document production. The UAN easily provides general scripts for user tasks which can easily be transformed into rich examples and document skeletons. The common frame of reference provided to both documentors and developers facilitates new insights into design. Errors and discrepancies in design of both documentation and the system are more easily identified due to the common user analysis built into the UAN. The early construction of documentation leads to a better awareness of users and the system necessary for quality documentation. This early involvement creates and maintains increased developmental communication which leads to more consistent design. The use of UAN as a technique for integration enables the realization of the goals of education, motivation and communication.

6.9 Costs of UAN

There are difficulties to the use of UAN. First, user analysis is critical to the creation of the UAN. There is a danger that stereotyping or even system orientation could be used in the development of the user analysis [GOUJ85] (see section 5.5).

Table 6.3 Costs of UAN

Danger of stereotyping
Danger of middle or reverse engineering UAN
Resistance to change
Slight learning curve
Complexity of UAN task web
Commitment to consistency and Cooperation

Next, there is a temptation to engineer the UAN in the middle of design or even after design. Paretti began UAN somewhat towards the middle of the design effort. [PARM92]. This leads to unnecessary costs in time and money because

the UAN is not there to provide insight into design. The lack of foresight causes design error and less design enhancement. The UAN rectifies this over time but resources are wasted in the short term. As described in chapter four, there is organizational resistance to change. This applies to inclusion of documentors and by extension the use of UAN. Supporting this resistance is a slight learning curve and the complexity of the UAN web. The UAN task web is difficult to visualize and requires a hypertext organization to map its complexity. The development of this tool could aid in the increased acceptance and use of the UAN. Finally, the UAN requires a commitment to consistency and cooperation that may be difficult in some design efforts.

6.10 Summary

User Action Notation is a behavioral design technique for use in interface development. UAN is task oriented and can serve as a means of integration between document and interactive systems development. Its open and flexible nature, make it ideal for this purpose. When used with the guideline family for quality documentation, accurate, early development of skeletal documentation becomes feasible. UAN promotes communication, iteration and a focus on user tasks. While easy to learn an use, a small learning curve may cause resistance. Stereotyping must be avoided in order for accurate UAN to be produced. Last, the same resistance to documentors in the design effort could apply to UAN. The difficulties can be avoided by the development of a hypertext tool to accurately store, maintain and retrieve UAN. By using UAN and the guideline family, the goals of education, motivation and communication become attainable and integrated within the development effort.

Chapter 7: Conclusions

It ain't over 'til it's over - Yogi Berra

7.0 Summary

This project has centered around a massive literature review to determine the qualities and means of producing good documentation. The literature reveals the primary goals of documentation are education, motivation and communication. Chapters one and two discuss the nature of documentation in detail. Chapter two introduces the four keys to the production of documentation and elaborates on the first key of style. Chapter three details the specifics of the remaining keys of appropriateness, usability and learnability. To facilitate the production of quality documentation a user focused approach that integrates document and interactive systems development is needed. This is accomplished best via an iterative process. Issues of integration and iteration are discussed in chapter four.

However, a common frame of reference is needed to accomplish these goals. Chapter five illustrates this need and establishes five superseding principles and a family of five codependent guideline groups based on the previous chapters. Chapter 6 introduces the User Action Notation, the chosen technique to foster cooperation and promote the development of documentation. UAN background and examples are provided as well as how the UAN when applied in conjunction with the guideline family bring about integration of document and software development. Examples are given that illustrate how the UAN, by its nature, yields the foundations of skeletal task-oriented documentation.

This chapter reflects on these concepts, and suggests what is and is not possible with UAN regarding integrated documentation development. Additionally areas of future work and concluding remarks are presented.

7.1 What UAN CAN Do

With regard to the early, integrated development of user documentation, the UAN is capable of fulfilling a wide variety of needs. When used in conjunction with the guideline family, integration and construction of accurate skeletal documentation are a natural consequence. Table 7.1 reflects some of the uses of UAN towards integrated document and interactive systems development

Table 7.1 What UAN CAN Do

Construct task oriented outlines
Construct task oriented indices
Construct generic scripts
Construct rich example sets
Construct skeletal filler
Consistent task descriptions
Produce early document skeleton
Assure accurate information content
Prepare scripts for usability testing
Provide common frame of reference
Facilitate better communication
Detect & correct design error
Accuracy of information content

Most of these benefits are possible due to the shared user analysis created during the development of the UAN(see chapter six). This common core leads to more complete, flexible, and accurate user centered design.

7.2 What UAN CANNOT Do

Although a powerful descriptive technique, there are some functions UAN, as it now stands, cannot perform. First, the technique is an aid to the developer of documentation; it cannot replace the documentor. Carroll states that manuals cannot be 'cranked out' [CARJ88a] [CARJ88b] [BROR90]. Any guideline or technique must acknowledge this fact. Similarly, the UAN cannot crank out a system. To be used accurately, the UAN can serve as a 'user prototype' during usability testing.

Table 7.2 What UAN CAN'T Do

Crank out manuals
Crank out systems
Convey system context information
Validify underlying user analysis
Replace the documentor

The UAN is behavioral in nature and must be translated into the constructional realm [HARH90] [HARH92]. This deficiency leads to some other weak areas. As noted in the discussion of example 6.5, the UAN cannot adequately convey system oriented information. The meaning and context of a system oriented term such as 'marked for deletion' must be translated. Thus, while a vehicle for communication, the UAN cannot substitute for cooperative communication. This need to communicate is related to another weakness in the UAN. UAN does not reveal the foundation of its underlying user analysis. Short of actual testing or knowledge of the authorship, there is no way to determine the soundness of the UAN. This marks the need for both honesty and a technique for verification concerning development of UAN.

7.3 Future Considerations

The benefits and problems of UAN reveal a wealth of for future work. Also, current advances in 'conference computing' where many users interact on a single application simultaneously provide some intriguing possibilities. The following are some areas that either need consideration or will benefit from future work with the UAN.

Dorsey [DORE92] has identified a set of characteristics of documentation that mirror to a great extent the guideline family. Reconciliation of the two would be beneficial to validation of the guideline family.

Dorsey [DORE92] also developed a Document Analysis Tool (DAT) which could be modified to access the quality of UAN

and resulting UAN based documentation.

This research has consisted of a review of existing literature. The next logical step is empirical verification of the guideline family and usefulness of UAN as a integrated technique to foster document development.

UAN does not adequately express system internals, because of its behavioral nature. However, certain system context information is crucial to document development, i.e., what is meant by 'marked for deletion'. Perhaps some method to not only name connections to computation but briefly clarify them should be added.

Hix and Chase have suggested modification of UAN to incorporate inclusion of user cognitive processes in the form of memory cognition action tables [HIXD92a] [CHAJ92]. This is crucially important to both usability and learnability of documentation. Also, when used in conjunction with the guideline family, it is of great use in determining choice of metaphor, creation of task oriented outlines and formation of questions to key learning.

Another area of interest is online documentation, which IS a part of the online interface. Aspects of good documentation are invariant regardless of media (see chapter two). However, retrieval mechanisms used to access online information as well as the information itself will be designed in UAN. Thus, the UAN is even more appropriate. The extent of applicability of the guideline family, the UAN and possible modification for online merit consideration.

The area of work that is perhaps the most crucial is the development of a hypertext UAN information system. This would aid maintaining UAN throughout design. Also, the development of an accompanying document development tool would also be useful (see Appendix B). Such a tool would take UAN and automatically produce task oriented outlines and skeletal scripts for documentation development.

7.4 A Disclaimer Towards System Orientation

This work has been unduly hard on System oriented information. Constructional information is crucial to development of usable documentation. The difficulty is the heavy reliance so often placed on that information. System information such as what is meant by 'marked for deletion' is crucial to proper usage. However, how and why to do a task is equally or more important than internal consequences. By utilizing a behavioral technique to create documentation, such constructional information is put in its proper place. It is supplemental to user tasks and used only to clarify those tasks not detail them.

7.5 Conclusions

Since a user oriented approach is most appropriate for documentation anything that helps in the creation of a user analysis is beneficial to that documentation. The User Action Notation, due to it nature as a technique to capture and communicate interface design is aptly suited to aid in this endeavor. The guideline family evident in the literature, when applied to the UAN satisfies or helps to satisfy directly or indirectly the integration and iteration of user documentation with interactive systems development. UAN reflects aspects of style, appropriateness, usability and learnability essential to document production. UAN is most beneficial in the construction of skeletal documentation in the form of task oriented outlines, indices and examples. The skeletal documents are flexible and can reduce costs by adapting easily to changing design. While there are difficulties to the use of UAN to achieve these goals, they are not insurmountable and are outweighed by the potential benefits. Through use of UAN and the Guideline family the primary documentation goals of education, motivation and communication become attainable realities.

Bibliography

[ANDJ86] Andersen J.R. "Acquisition of Cognitive Skill. Psychological Review 1986:pp. 369-406.

[BANL86] Bannon, Liam J. "Issues in Design: Some Notes." in User Centered System Design. Donald A. Norman & Stephen W. Draper (eds.) (Hillsdale, New Jersey: Lawrence Erlbaum Associates Pub., 1988).

[BEHL80] Behnke, Lynn. "Planning Software Manuals." Technical Communication. (Fourth Quarter, 1980) pp. 6-10.

[BETF91a] Bethke, F.J. "Usability Revisited." SIGDOC Asterisk Vol.15 No.2, September 1991: pp. 41-43.

[BETF91b] Bethke, F.J., et al. "Improving the usability of programming publications." SIGDOC Asterisk Vol.15 No.2, September: 1991, pp. 3-18.(reprinted from IBM Systems Journal, Vol.20, No.3, 1981)

[BLAT84] Blanding-Clark, Theresa and Cross, Thomas B. "Designing Effective User Interface Documentation." Journal of Information and Image Management. (November,1984):pp. 45-48.

[BOAB84] Boar, Bernard K. "Alleviating Common Concerns of Application Prototyping -- The Experience Difference." Computer World, Vol.18, No.22 (1984). pp. 66-67.

[BOEB84] Boehm, Barry W., Gray, Terence E., and Seevaldt, Thomas. "Prototyping versus Specifying: A multi-project experiment." IEEE Transactons on Software Engineering. Vol.10 No. 3 (1984).

[BRAD90] Bradford, David. "The Persona in Microcomputer Documentation." IEEE Transactions on Professional Communication. Vol.PC-27,No.2. June, 1984: pp. 65-68.

[BROR90] Brockmann, R. John "The Why, Where and How of Minimalism." in Proceedings SIGDOC'90 in special issue of SIGDOC Asterisk Vol. 14, no. 4, pp.111-121.

[CARJ88a] Carroll, John M., et al. "The Minimal Manual." in Human Computer Interaction. Vol.3 (1988):pp123-153.

[CARJ88b] Carroll, John M., and Aaronson, Amy P. "Learning By Doing with Simulated Intelligent Help." Communications of the ACM. Vol.31, No.9. September, 1988. pp. 1064-1079.

[CARJ85a] Carroll, John M. "Minimalist Design for the Active User." in Human-Computer Interaction - INTERACT '84, B.Shackel (ed.) (North-Holland:Elseveir Science Publishers, B.V. 1985) pp. 39-49.

[CARJ85b] Carroll, John M. "Metaphor, Computing Systems and Active Learning." International Journal of Man-Machine Studies, (1985) Vol.22, pp. 39-57.

[CARJ84] Carroll, John M. "Minimalist Training." Datamation. November 1, 1984. pp.125-135.

[CHAJ92] Chase, Joseph D., Personal Interview, July 1992.

[CLAC83] Clanton, Chuck "The Future of Metaphor in Man-Computer Systems." Byte. December, 1983: pp. 263-280.

[CLAF83] Clark, Frank. "Special Report: Software Documentation a Major Problem." Computer World. February 28,1983: pp.13-14.

[DILF91] Dilorio, Frank C. SAS Application Programming - A Gentle Introduction. (Boston, Mass.: PWS-Kent Pub., 1991), pp.77-85.

[DORE92] Dorsey, Edward Vernon IV., "The Automated Assessment of Computer Software Documentation Quality using the Objectives/Principles/Attributes Framework." Thesis, Virginia Polytechnic Institute and State University, October,1992.

[DORP88] Dorazio, Patricia. "Help Facilities a Survey of the Literature." Technical Communication. (Second Quarter,1988) pp118-121.

[DUIA89] Duin, A.H. "Factors That Influence How Reader's Learn from Text." Technical Communication. 36:2 (April,1989): pp.97-101.

[EGAD86] Egan, Dennis E., "Individual Differences In Human-Computer Interaction." in Handbook of Human-Computer Interaction. M. Helender (ed.). (North Holland:Elseveir Science Publishers B.V., 1986).

[FARD90] Farkas, David K. and Williams, Thomas R. "John Carroll's *The Nurnberg Funnel* and Minimalist Documentation." IEEE Transactions on Professional Communication. Vol.33, No.4. December, 1990: pp.182-187.

[FARD86] Farkas, David. "Product Support and Documentation Writing: Potential Synergism." in Proceedings of 33rd International Technical Communications Conference. Washington, D.C.: Society for Technical Communication, 1986: pp. 105-107.

[FISL88] Fisher, Lori H. "Getting Involved Early in the Software Development Process." in IEEE 1988 Professional Communications Conference Proceedings. 1988: pp.109-112.

[FOWS86] Fowler, S.L. and Roeger, D. "Programmer and Writer Collaboration: Making User Manuals that Work." IEEE Transactions on Professional Communication. Vol. PC-29, No 4, December 1986: pp.21-25.

[GAIB79] Gaines, B.R. "Man/Computer Communication An Overview." InfoTech State of the Art Report. Vol.2 pp.141-153.

[GERV87] Gervickas, Vicki. "Prototyping: A Model Approach to Development." in Proceedings of the 34th International Technical Communications Conference, Washington, D.C.: Society for Technical Communication, 1987:pp.WE161-WE162.

[GIRT91] Girill, T.R. "Ease of Use and the Richness of Documentation Adequacy." ACM SIGDOC Asterisk. Vol.15,No.2 (September 1991).

[GLUR82] Glushko, R.J. and Bianchi, M. H. "On-line Documentation: Mechanizing, Development, Delivery and Use." The Bell System Technical Journal. July-August 1982: pp.1313-1323.

[GOON87] Goodwin, N.C. "Functionality & Usability" Communications of the ACM 30,3(1987): pp.229-233.

[GOUJ85] Gould, John D. & Lewis, Clayton. "Designing for Usability: Key Principles and What Designers Think." Communications of the ACM. Vol.28,No.3, March 1985: pp. 300-311.

[GRIR88] Grice, Roger. "Information Development Is Part of Product Development- Not an Afterthought." in Text, ConText and HyperText: Writing with and for the Computer. Edward Barrett (ed.), (Cambridge, Massachusetts: MIT Press 1988): pp. 133-148.

[GRIS81] Grimm, Susan J. "EDP Documentation: the missing link." IEEE Transactions on Professional Communication. Vol. PC-24, No.2, June, 1981: pp. 79-83.

[GUIR89] Guilemette, Ronald A. "Usability in Computer Documentation Design: Conceptual and Methodological Considerations." IEEE Transactions on Professional Communication. Vol.32, No.4, December, 1989:pp.217-229.

[GUIR87a] Guillemette, R. A. "Application Software Documentation." Journal of Systems Management. 38,5(1987): pp.36-39.

[GUIR87b] Guilemette, R. "Prototyping: An Alternate Method for Developing Documentation." Technical Communication. 34:3 (August 1987):pp. 135-141.

[HARH92] Hartson, H. Rex and Gray, Philip D. "Temporal Aspects of Tasks in the User Action Notation." forthcoming in Human Computer Interaction Vol. 7, No. 1, 1992.

[HARH90] Hartson, H. Rex, Siochi, Antonio C., and Hix, Deborah. "The UAN: A User-Oriented Representation for Direct Manipulation Interface Designs." ACM Transactions on Information Systems. Vol. 8. No.3, July, 1990, pp.181-203.

[HARR78] Harris, Richard J. and Monaco, Gregory E., "Psychology of Pragmatic Implication: Information Processing Between The Lines." Journal of Experimental Psychology: General, Vol.107,No.1, March,1978: pp. 1-22.

[HENA84] Henderson, Allen, "The Care and Feeding of The Non-Captive Reader." Technical Communication. Third Quarter, 1984:pp.5-8

[HERR87] Herrstrom, David S. "An Approach to estimating the cost of Product Documentation, with Some Hypotheses." in Proceedings of the 34th International Technical Communications Conference, Washington, D.C.: Society for Technical Communication,(1987): pp. MPD-24 - MPD-27.

[HIXD92a] Hix, Deborah, and Chase, Joseph D. "The UAN: A User-Oriented Notation for Direct Manipulation Interface Designs - A Tutorial." DRAFT. Virginia Polytechnic Institute & State University, 1992.

[HIXD92b] Hix, Deborah. HCI Group UAN Case Research Notes. August, 1992.

[HOOK88] Hooper, Kristine. "Architectural Design: An Analogy." in User Centered System Design. Donald A. Norman and Stephen W. Draper (eds) (Hillsdale, New Jersey: Lawrence Erlbaum Associates Pub., 1988).

[HORR92] Horn, Robert E. "Commentary on the Nurnberg Funnel." SIGDOC Asterisk Vol.16, no.1, January,1992:pp.3-11.

[HORW89] Horton, William. "HyperText Manifesto: Reader's Rights, Writer's Responsibilities." Technical Communication, Vol.36, 1(1989):73-74.

[HOUD91] Houghton-Alico, Doann. "Side by Side, A Model for Simultaneous Documentation and System Development." in Perspectives on Software Documentation: Inquiries and Innovations. Thomas T. Barker (ed.). (Amityville, New York: Baywood Press, 1991):pp. 154-174.

[HOUR84] Houghton, Raymond "On-Line Help Systems A Conspectus." Communications of the ACM. Vol. 27, No.2 (Feb. 1984):pp.126-133.

[HUSK88] Huston, K. and Southard, S. "Organization:The Essential Element in Producing Usable Software Manuals." Technical Communication 35:3 (August 1988):pp.179-189.

[LEWC88] Lewis, Clayton. "Understanding What's Happening in System Interactions." in User Centered System Design. Donald A. Norman and Stephen W. Draper (eds) (Hillsdale, New Jersey: Lawrence Erlbaum Associates, Pub. 1988): pp.171-186.

[LEIJ91] Lieberman, Jay. "A Schematic Approach to User Knowledge and Software Documentation Production." in Perspectives in Software Documentation: Inquiries and Innovations. Thomas T. Barker (ed.). (Amityville, New York:Baywood Press 1991): pp. 61-72.

[MACR83] Mack,Robert L., Lewis, Clayton and Carroll, John M.."Learning to Use Word Processors: Problems and Prospects." ACM Transactions on Office Information Systems. Vol.1,No.3, July, 1983: pp. 254-271.

[MARP86] Martin, Peter. "Computer User Documentation Problems: Their Causes and Solutions." IEEE Transactions on Professional Communication. Vol. PC-29,No.4. December, 1986.

[MARP84]Martin,Peter. "Ergonomics in Technical Communication ." IEEE Transactions on Professional Communication. Vol.PC-27,No.2 (June, 1984): pp. 62-64.

[MAYJ82] Maynard, John. "A User Driven Approach to Better Documentation."IEEE Transactions on Professional Communication. Vol.PC-25,No.1, March 1982: pp.16-19.

[MAYR88] Mayer, Richard E. "From Novice to Expert." in Handbook of Human-Computer Interaction. M. Helender (ed.). (North Holland:Elseveir Science Publishers B.V., 1986).

[McDM86] McDonald,Marl. "Using Examples in Software Documentation." in Proceedings of 33rd International Technical Communications Conference. Washington, D.C.: Society for Technical Communication, 1986: pp.430-435.

[McGK86] McGraw, Karen L. "Guidelines for Producing Documentation for Expert Systems." IEEE Transactions on Professional Communication. Vol.PC-29, No.4. December, 1986.

[MILG56] Miller, George A. "The Magic Number Seven, Plus or Minus Two:some limits on our capacity for processing information." Psychological Review Vol.63,No.2. March, 1956: pp. 81-97.

[MINA83] Mingione, Al. "Iteration, Key to Useful Documentation." Journal of Systems Management, January, 1983: pp.23-25.

[MIRB92] Mirel,Barbera. "Analyzing Audiences for Software Manuals: A Survey of Instructional Needs for Real World Tasks." Technical Communication Quarterly. Vol.1,No.1, Winter, 1992:pp.13-38.

[MIRB91a] Mirel,Barbera. "Toward a More Precise Definition of Task Sufficient Information." ACM SIGDOC Asterisk, Vol.15, No.2. (September, 1991): pp. 22-32.

[MIRB91b] Mirel, Barbara, Fineberg, Susan and Allmendinger, Lief "Designing Manuals for Active Learning Styles." Technical Communication. 38,1 (1st Quarter, 1991):pp. 85-87.

[MITD80] Mitchell,D.M. and Ransom,M.J. "Writing Computer Manuals." Technical Communication. (4th Quarter,1980): pp.10-12

[MOOM88] Moore, Mary Ann Dohn and Vanderlindern, Gay. "Developing the User Learning Model:How Novices Become Experts." in IEEE 1988 Professional Communications Conference Proceedings. 1988: pp.291-299.

[MOYJ82] Moynihan, John A. "What User's Want". Datamation 1982.

[NORD88] Norman, Donald. "Cognitive Engineering." in User Centered System Design. Donald A. Norman and Stephen W. Draper (eds) (Hillsdale, New Jersey: Lawrence Erlbaum Associates Pub, 1988):pp.31-62.

[ODEE86] Odescalchi, E.K. "Documentation is the Key to User Success." IEEE Transactions on Professional Communication. Vol. PC-29,NO.4. December, 1986: pp.16-18.

[OMAC88] O'Malley, Claire E. "Helping Users Help Themselves." in User Centered System Design. Donald A. Norman and Stephen W. Draper (eds.) (Hillsdale, New Jersey: Lawrence Erlbaum Associates, Pub. 1988): pp. 377-398.

[OMAC84] O'Malley,C, et al. "A proposal for User Centered System Documentation." in Proceedings CHI'84: Conference on Human-Computer Interaction. 1984.

[ORAA92] Oram, Andrew. "Do We Dare Free Our Computer Users?" Technical Communication. 39, 1 (1st quarter 1992):pp. 60-68.

[ORAA91] Oram, Andrew. "The Hidden Effects of Computer Engineering on User Documentation." in Perspectives on Software Documentation: Inquiries and Innovations. Thomas T. Barker ed. (Amityville, New York: Baywood Press, 1991).

[ORAA86] Oram, Andrew. "Opening Passage: A New Look at the Software Documentation Problem." IEEE Transactions on Professional Communication. Vol PC-29, No.4. December, 1986.

[PARM92] Paretti,Marie, Personal Interview, August 18,1992.

[PARS86] Partridge, Stephen Kurt. "So What is Task Orientation anyway?" IEEE Transaction on Professional Communication. Vol. PC-29, no.4. December, 1986: pp. 27-32.

[PASG76] Pask,G. Conversation Theory Amsterdam:Elsevier 1976.

[POLM85] Pollack, Martha E. "Information Sought and Information Provided: An Empirical Study of User/Expert Dialogues." in Proceedings CHI'85 Conference:Human Factors in Computing Systems. New York: ACM(1985): pp. 155-159.

[PRID91] Prigge,David. "A Response to Bethke et.alii" ACM SIGDOC Asterisk. Vol.15,No.2. September,1991: pp.33-36.

[REDJ88] Redish, Janice. "Reading to Learn to Do." The Technical Writing Teacher. Vol.15, No.3 (1988): pp.223-233.

[RENL87] Renaldo,Lisa. "Prototypes:A Means to an End." in Proceedings of the 34th International Technical Communications Conference. Washington, DC: Society for Technical Communication,(1987): pp.WE163-WE165.

[RILM88] Riley,Mary S. "User Understanding." in User Centered System Design. Donald A. Norman and Stephen W. Draper (eds) (Hillsdale, New Jersey: Lawrence Erlbaum Associates, Pub. 1988): pp. 157-169.

[ROSS86] Rosenbaum, Stephanie and Walters, R. Dennis. "Audience Diversity: A Major Challenge in Computer Documentation." IEEE Transactions on Professional Communication. Vol. PC-29, No.4., December, 1986: pp.48-56.

[ROSD86] Rosich,D. and Grice, Roger A. "From the Guest Editors: The Growing Importance of Computer Documentation." IEEE Transaction on Professional Communication. Vol. PC-29, no.4. December, 1986.

[RUBB86] Rubens, Brenda. "Personality in Computer Documentation: A preference study." IEEE Transactions on Professional Communication. Vol.PC-29, No.4. December, 1986.

[SANC91] Sankar, Chetan S. "The Role of User Interface Professionals in Large Software Projects." IEEE Transactions on Professional Communication. Vol.PC-34, No.2, June, 1991: pp. 94-99.

[SCHD86] Schell, David A. "Testing On-line and Print Documentation." IEEE Transactions on Professional Communications. Vol.PC-29,No.4. December, 1986: pp.87-92.

[SCHK89] Schriver, Karen A. "Document Design from 1980 to 1989: Challenges that remain." Technical Communication. Fourth Quarter 1989: pp.316-329.

[SHNB82] Shneiderman, Ben. "System Message: Guidelines and Experimental Results." in Directions in Human-Computer Interaction. Albert Badre & Ben Shneiderman (eds). (Norwood, New Jersey: Ablex Pub. Company,1982).

[SODC85] Soderston, Candace, "The Usability Edit: A New Level." Technical Communication First Quarter,1985:pp.16-18.

[SOHD83] Sohr,Dana "Better Software Manuals." Byte, May,1983 :pp.286-294.

[STED86] Steinhauer, Diane L. "User Friendly Pages in Computer Manuals." in Proceedings of the 33rd annual Technical Communications Conference. 1986: pp.214-217.

[TOMR82] Toms, R.K. "Human Factors in User Documentation." Information and Management. 5(1982): pp. 149-155.

[VOGH87] Voght, Herbert E. "Usability Must Be Designed In Not Added On At The End." in Proceedings of the 34th annual Technical Communications Conference. Washington, D.C.: Society for Technical Communication.(1987): pp. WE103-WE106.

[WALR84] Walinsky, Robert., Chens, L., and Chin, David. "Talking to UNIX in English: An Overview of UC" Communications of the ACM. Vol.27,No.6, (June, 1984): pp. 574-593.

[WALJ87] Walker, Janet. "Issues and Strategies for On-line Documentation." IEEE Transactions on Professional Communication. PC-30 (1987).

[WEIE85] Weiss, Edmund "InDepth: The Next Wave of User Documentation." Computer World. September 9, 1985. ID15-ID24.

[WILL81] Wilson, Lindsay. "Death, taxes and DP Documentation." Datamation. February,1981: pp.73-76.

[WRIP85] Wright, Patricia. "Manual Dexterity: A user oriented approach to writing computer documentation." in Proceedings CHI '85 Conference: Human Factors in Computing Systems. New York:ACM (1985). PP 11-18.

[ZEIK88] Zeidenstein, Kathryn. "Collaboration as Innovation: Why Technical Communicators Should be Members of the Software Development Team." in IEEE 1988 Professional Communications Conference Proceedings. 1988, pp. 79-83.

Appendix A: Useful UAN Symbols [HARR90]

ACTION	MEANING
~	move to cursor
[X]	the context of object 'X', the handle by which 'X' is manipulated
~[X]	move the cursor into the context of 'X'
~[x,y]	move the cursor to (arbitrary) point x,y outside any object
~[x,y in A]	move the cursor to an (arbitrary) point x,y within object A
~[X in Y]	move to object X within object Y
[X]~	move the cursor out of context of object X
∨	depress
∧	release
X∨	depress button, key or switch called X
X∧	release button,key or switch X
X∨∧	idiom for clicking button, key or switch
X"abc"	enter literal string abc via device X
X(xyz)	enter value for variable xyz via device X
()	grouping mechanism
*	iterative closure, task is performed zero or more times
+	task is performed one or more times
{}	enclosed task is optional (performed zero or more times)
A B	sequence;perform A then B (same if A and B are on separate, but adjacent lines
OR	disjunction, choice of tasks (used to show alternative ways to perform a task)
&	order independence: connected tasks must all be performed, but relative order is immaterial
⇔	interleavability; performance of connected tasks can be interleaved in time
‖	concurrenency; connected tasks can be performed simultaneously
;	task interrupt symbol; used to indicate that user may interrupt task at this point (the effect of this interrupt is specified as well, otherwise it is undefined, i.e., as though the user never performed the previous actions at all
∀	for all
:	separator between condition and action or feedback
Feedback	**Meaning**
|	highlight object
-|	de highlight object
||	same as |, but use an alternative highlight
|-|	blink highlight
$(\|-\|)^n$	blink highlight n times
@x,y	at point x,y
@X	at object X
display(X)	display object X
erase(X)	erase object X
X>~	object X follows (is dragged by) cursor
X>>~	object X is rubber banded as it follows cursor
outline(X)	outline of object X

Appendix B: Some Suggestions for Automating Skeletal Document Production via UAN

- The system should be hypertext in nature due to the complexity of UAN task web.

- The system must provide a flexible store of UAN symbols, i.e., the system should allow users to enhance the UAN. This store is analogous to a data dictionary.

- The system should allow the creation of special purpose UAN symbol stores that intermix with the main store to accommodate of variety of situations. This aids in developing special UAN extensions for specific domains and devices. For example, pen computing or word processing extensions.

- The UAN symbols should have corresponding mappings to normal language, i.e., ~ ⇒ move cursor,
  M ⇒ mouse.

- The system should allow editing of these mappings. This is necessary because of the addition of new symbols, special symbol stores and changes in meaning that occur over time. In cases where the symbols are used in a suite of applications, the mappings should be dynamically linked to all relevant applications.

- For each development effort, a data dictionary of screen-objects and input/output devices must be created that contain schemata for the objects and devices that need to be represented. For suites of applications (i.e. Macintosh or MS-Windows), the dictionary must be flexible so changes that affect the suite propagate to the specific applications.

• A parsing mechanism should exist to enable different actions to have different feedback, interface states or connections to computation depending upon the context. For example:

~[file-icon]M^{V} could have different feedback than ~[file-folder-icon]M^{V}

• The parsing mechanism should allow skeletal documentation to be produced in a manner consistent to that discussed in chapter six.

• The system should allow specific information to be assigned to interface objects and then generate text containing the specific information rather that the abstract interface object. This information should be stored and linked to the relevant UAN interface objects.

• The system should allow a manipulative view of the UAN task web to facilitate creation and maintenance of a task oriented hierarchy (outline).

• Retrieval of specific actions, feedback, states or computational connections should be available and modifiable by query.

i.e., FIND ALL [file-icon]M^{V}

would locate and link instances where in the UAN the cursor is moved to the file icon and the mouse button is depressed.

• Changes in any part of a UAN task description, should propagate to all linked areas within the UAN and skeletal documentation. For example, if the subtask select_object changes, all super tasks that use that general subtask are also affected. These super tasks, in turn alter their respective superior tasks.

• The system should allow multiple views of information:
  - User Actions and/or corresponding skeletal text,
  - Feedback and/or corresponding skeletal text'
  - States and/or corresponding skeletal text'
  - Computational Connections and related information,
  - UAN alone or some combination of the above,
  - Document skeleton and/or a combination of the above,
  - Design annotations(i.e.,change imminent) and some, combination of the above.
  - Task outline at various levels of detail: Main task, super task and/or subtask.

• The detail of generated skeletal documentation should be readily selectable and allow simultaneous construction of detail differing text (i.e., rich task descriptions to minimal scripts for usability testing).

Vita:

Barry Towe was born March 4,1968. He has had the good fortune to have lived the past 24 years at the foot of the Blue Ridge Mountains of Virginia. His early years were plagued with chronic health problems stemming from acute asthma and severe allergies. At age twelve, his health was so poor that it endangered his performance at school. An instructor arranged a meeting with his parents and retorted "Your son is incapable of making an 'A'". However, the ensuing years led to three governor's magnet schools.

In 1986, he enrolled in Clinch Valley College of UVA, and pursued a double major in Business and Computer Science Information Systems. In May 1990, he graduated cum laude with a 3.56 grade point average. In 1990, he applied to one graduate school -- Virginia Polytechnic Institute and State University. This work is the culmination of his studies at Virginia Tech.

If a moral exists to this life, it is that no one individual dictates your future. Despite ridicule, if you have faith and believe in your abilities, you can achieve and overcome.

"The best angle to approach any problem is the TRY angle."
-- Anonymous

James Barry Towe, December 1992.