Evaluating a Multimedia Authoring Tool with Cognitive Walkthrough and Think-Aloud User Studies

Bonnie E. John & Matthew M. Mashyna

28 August 1995
CMU-CS-95-189

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Also appears as Human-Computer Interaction Institute Technical Report
CMU-HCII-95-105

This work was supported by the Advanced Research Projects Agency, DoD, and monitored by the Office of Naval Research under contract N00014-93-1-0934. The think-aloud usability data was collected and analyzed at CMU's User Studies Lab with equipment acquired through the National Science Foundation Equipment Grant #9022511. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of ARPA, ONR, NSF, or the U. S. Government.

Bonnie John is associated with the Departments of Computer Science and Psychology and the Human-Computer Interaction Institute at Carnegie Mellon University. Matthew Mashyna is associated with the Masters of Software Engineering Program and the Department of Psychology.

Keywords: Cognitive Walkthrough, usability engineering, user studies, multi-media

Abstract

We present a detailed case study, drawn from many information sources, of a computer scientist learning and using Cognitive Walkthrough to assess a multi-media authoring tool. We then compare the predictions produced by the analysis to the usability problems actually found though empirical usability tests. This study results in several clear messages to both system designers and to developers of evaluation techniques: (1) the Cognitive Walkthrough technique is currently learnable and usable, but (2) it may not predict most usability errors found empirically, and (3) several elaborations of the technique might improve its effectiveness. In addition, the emergent picture of the process this evaluator went through to produce his analysis sets realistic expectations for other novice analysts who contemplate learning and using Cognitive Walkthroughs.

1. Introduction

Information systems in our networked and multimedia age must be usable by the people they are intended to serve. From the professional librarian expert in database searches to the school-child with a multimedia encyclopedia on a CD-ROM to new surfers of the World Wide Web, people must be able to find the information they want. From the professional cataloger to the student using a hypertext system to link class-notes to homework assignments, people must be able to create categories and links that allow themselves and others to find that information again. From CD-ROM textbook publishers to the PTA news-letter volunteers, people must be able to create the documents that give expression to their ideas.

But how do you know if an information system is usable? All too often, this question is only answered after a system is built and put into use, and praise or complaints trickle back from users. All too often, unusable systems are put into place at extreme capital cost to the organization, extensive training costs for its staff, and continuing costs for end-users in terms of their time and frustration-level.

To address these problems, researchers in the field of Human-Computer Interaction have developed several methods to evaluate computer system usability. These techniques are intended to evaluate a system before it is built, even before a prototype exists. Some are based on modeling the user's cognitive processing to evaluate the procedures people need to learn and use to accomplish tasks with the computer system (e.g., GOMS, Card, Moran & Newell, 1983; John & Kieras, 1994). Others are primarily based on heuristics gleaned from years of user interface design and development (e.g., Heuristic Evaluation, Nielsen, 1994). Others are primarily based on software engineering design procedures applied to the specification and evaluation of the user interface (e.g., User Action Notation, Hartson, Siochi, & Hix, 1990). And some are a combination of these influences (e.g., Cognitive Walkthrough is a combination of psychological theory and software engineering practices, Wharton, Rieman, Lewis, & Polson, 1994). All of these techniques are intended to be useful for system designers with many types of background: professional programmers, graphic designers, domain experts, and individuals creating their own information landscape.

Although promising, the development of such methods is relatively new and few of these techniques have been subjected to rigorous validation (with the notable exception of GOMS, whose predictions have been extensively tested both in the lab and in the field, e.g., Gong & Kieras, 1994; Gray, John & Atwood, 1993). Open questions remain about which techniques actually predict usability problems that will cause users to fail at their real-world tasks or be inefficient, annoyed, or even misinformed. We do not yet know which techniques work for which types of systems or user problems, or when in the design process to apply them, or what educational background is necessary to successfully use each technique, or how to combine techniques to get the most benefit with the least effort.

Recently, some work has been done that compares different techniques (e.g., Cuomo & Bowen, 1994; Desurvire, 1994; , Jeffries, Miller, Wharton & Uyeda, 1991; Karat, 1994). This work has taken the form of experiments, formally comparing performance outcomes of different techniques. The dependent variables are typically quantitative: the number (and type) of usability problems identified, how closely a given technique predicts a user's behavior, the time it takes to perform the evaluation, the labor costs involved. Perhaps the biggest difficulty with these studies is that they provide no data about what people actually *do* when they are using these techniques. Without process data, it is difficult to understand how the technique itself leads the analyst to identify usability problems, as opposed to the analyst simply being clever. Without process data, a developer or analyst does not know what to expect when setting out to use a new technique. Finally, without process data, it is difficult to provide meaningful feedback to the method-developers so they can improve the techniques. Thus, at this formative stage in the development of evaluation techniques, richer, more insightful data are required than summative comparisons can provide.

In his book *Case study research: Design and methods*, Robert Yin (1994) states that a case-study approach has an advantage over surveys, experiments, and other research strategies "...when a 'how' or 'why' question is being asked about a contemporary set of events over which the investigator has little or no control." (p. 9, Yin, 1994) This seems to describe the situation facing the field of HCI when evaluating evaluation techniques. We are asking *how* a given technique can be used to predict usability problems, *why* it works in some situations and not in others, and we have virtually no control over how an analyst learns or uses a technique.

The essence of the case-study approach is to collect many different types of data and use them "in a triangulating fashion" (p. 13, Yin, 1994) to converge on an explanation of what happened. When multiple sources of information converge, it boosts our confidence that we have understood the series of decisions occurring in a case; how these decisions were made, how they were implemented, and what result was achieved. This deep understanding should allow us to know whether these processes and results are likely to reoccur with other developers or in the next design project.

This paper presents a case study of a novice analyst learning and applying the Cognitive Walkthrough (CW) usability assessment technique to a multimedia authoring tool.[1] The study analyzes several sources of evidence: problem description forms filled out as the analyst worked, a final report by the analyst, and, to obtain detailed process data, a diary kept by analyst while he learned and applied this technique. This case-study approach allows us to look at what the analyst did, the confusion and insights he had about the technique, as well as more traditional performance measures like the number and type of usability problems identified.

The next section of this paper will present a brief summary of the CW technique that the analyst learned and used in the case detailed in Section 3. Section 4 will present the empirical usability study performed to provide data with which to evaluate the predictive power of the technique. Section 5 will discuss the differences between the CW and the usability study. Finally, the concluding section will summarize the lessons learned from these studies.

---

[1] In all, data were collected with analysts using Cognitive Walkthrough, Claims Analysis, User Action Notation, GOMS, and Heuristic Evaluation, and simply reading the specification. However, all of these data have not yet been analyzed and this paper will report the details of only a Cognitive Walkthrough case.

2. The Case Study Situation

2.1. The analysis situation

It is not uncommon for a development team to ask one of its members (or an outside person) with an interest in HCI to make recommendations about a user interface design (in fact, it was just such a request that led the first author to switch fields from mechanical engineering to HCI 15 years ago). When this happens the analyst must figure out what assessment technique he or she wants to use, find books or papers describing how to use it, learn the technique from these materials, and apply it to the system design. In this study, we set up a similar situation in the following way.

2.1.1. Figuring out what technique to use

Five volunteer analysts were given a 30 minute lecture on HCI assessment techniques. This lecture was the methods part of the *Introduction and Overview of Human-Computer Interaction* tutorial given at the CHI conference for the last three years (Butler, Jacob & John, 1995). The analysts heard this lecture from the same lecturer (the first author) and received the same tutorial notes and bibliography as tutorial participants, which is the way scores of professionals get their first introduction to usability assessment methods each year. One week later, each analyst chose the method they wished to use.

2.1.2. The system and specifications

The analysts were given two documents with which to do their analyses: the user interface specification of the ACSE multimedia authoring system and a target multimedia document (described below). The analysts read these documents at home for one week and then were given two 1-hour sessions with the head developer of the system to clarify any points of misunderstanding. The results of these sessions were written up and sent to each analyst via e-mail. The analysts had access to the developer via e-mail if any other difficulties arose understanding the documents, and all such e-mail conversations were sent to all analysts.

The ACSE Multi-media Authoring System The Advanced Computing for Science Education (ACSE) project has built a multi-media science learning environment to teach the skills of scientific reasoning (Pane & Miller, 1993). This software system provides an author with a structured document framework and a set of tools for constructing science lessons containing text, still graphics, movies, and simulations. The system provides the science student with tools for navigating through the lesson, viewing the movies, and manipulating and running the simulations.

Based on several years experience with this system, the ACSE project recently designed a second version of it's authoring tool, called the VolumeViewBuilder (henceforth, *Builder*). The redesign was a group effort recorded in a user interface specification written by a software developer and a technical writer, both of whom had recently sat in on an HCI design class (Gallagher & Meter, 1993). The implementation of the Builder was begun at the same time as our case studies and continues at the time of this writing.

The ACSE project gave us their user interface specification for analysis. The specification included an introduction to the system and the goals of the document (4 pages), the interface for science students (the VolumeViewExplorer, 9 pages) and the interface for the Builder (22 pages). The specification included 37 figures of the screens (1 in the introduction, 11 in the Explorer, 24 in the Builder) which ranged from small pictures of specialized cursor icons to full-page figures of the entire screen (e.g., Figure 1).

The Example Multi-Media Document: The target multi-media document was a biology lesson about Drosophila development adapted from an advanced undergraduate biology course using the ACSE system for laboratory sessions. This volume is 55 pages long, and in addition to text contains 23 high resolution images and figures, 3 movies, 7 simulations, 10 fragments of simulation code, and 10 review questions. The original volume was produced with the first version of the Builder and did not included a table of contents, a glossary, or hyper-links, because those features were not included in that version of the Builder. The first author modified this lesson to include these features and produced a hard copy target document. This document did not run on a computer, but the links were explicitly indicated in supplementary lists, i.e., all the table of contents entries and their page numbers were in a list, all the glossary terms were in a list, all the hyperlink "hot phrases" and where they would point were in a list. This document and the lists were given to the analysts as a typical document created with the new Builder.

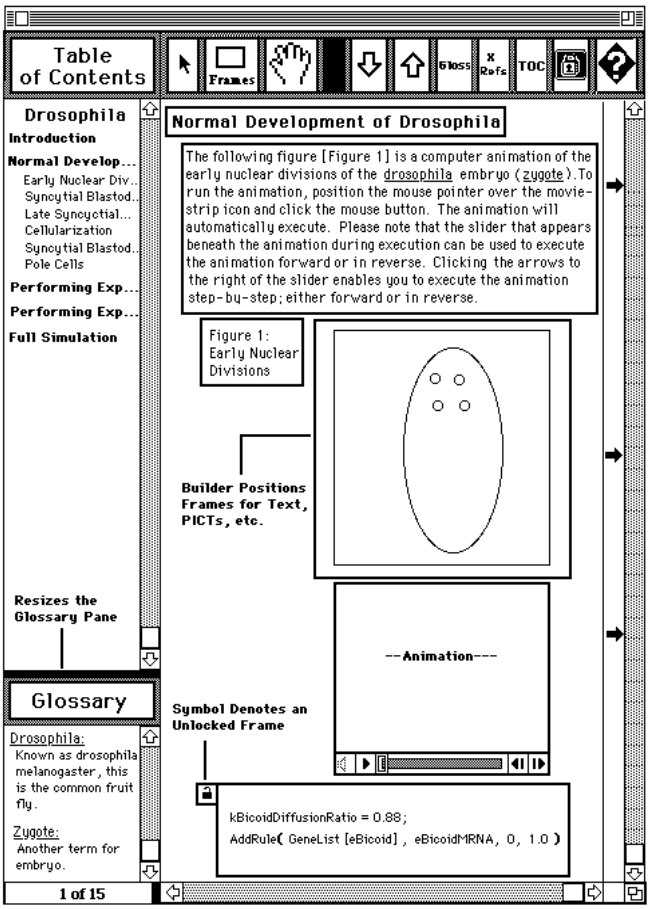

Figure 1. Example of an illustration in the *Volume View Interface Design* [5, p. 16]. In that document, this is a full-page illustration of the screen of the Builder application

2.2. The Analysis Process

The analysts worked primarily on their own for an elapsed time of ten weeks. They used two forms for recording their work on an ongoing basis: a structured diary and a problem description form (described below). The group met once or twice a week to discuss the analysis process. Each analyst produced both a verbal and written final report of their analyses. A questionnaire assessed the analysts educational and professional background. All of these data were analyzed to produce the case reported in this paper.

2.2.1. The Diary Form

The diary form was adapted from (Rieman, 1993). The analyst used it to record his or her activity each half hour while working on the analysis. The analyst wrote a short description of the activity and then placed it in one of six different categories: literature search, reading for "what-it-is", reading for "how-to-do-it", reading/analysis (when reading and analysis are so intertwined as to be inseparable), analysis (when the analyst knows the technique well enough to analyze without reference to the literature), and unrelated (e.g., copying papers). The form also had columns for recording a difficulty with the technique, an insight into the technique, a problem with the design, a solution to a design problem, and "other." At any time, the analyst could make a note on the form and write an extended explanation of what he or she was thinking in a separate, free-form diary.

2.2.2. The Problem Description Form

The problem description report (PDR) was adapted from (Jeffries, et. al., 1991). The PDR provided an area for describing the problem, an estimate of the severity and frequency of the problem, and an assessment of whether these judgments came from the technique itself, as a side effect of the technique, or from some form of personal judgment. Each PDR had a reference number that also appeared in a diary's column for recording a design problem.

2.2.3. The Analysts' Discussions

The discussions with the analysts were structured around several open-ended questions: What did you do in your analysis in the last couple of days? Did you have any difficulty with the technique? Did you have any insights into the technique? Did you discover anything else notable about the technique? Thus, these discussions centered around the process, not the content of the analyses. That is, they discussed things like problems getting or understanding papers, problems making the techniques applicable to the Builder, types of information their techniques needed or provided, but not specific usability problems with the Builder (e.g., that the method of creating hyperlinks was awkward or that the menu items were in the wrong place). The first author took notes during the discussions, which contributed to the case in this paper. These discussions were also audio taped, but the tapes have not yet been analyzed and do not contribute to this case.

2.2.4. The Final Report

Each analyst produced a written report that included a brief summary of the technique, an annotated bibliography of the papers used to learn and apply the technique, any modifications made to the technique to allow it to be used for the Builder specification, any areas of exceptional doubts or confidence about using the technique, suggestions for improving the technique, and the three most important problems that the designers should fix in the Builder.

3. The case of analyst, A1

A1 was a researcher in the School of Computer Science. He had taken over a dozen courses in CS, considered himself fluent in two programming languages, and had worked professionally as a programmer before taking part in this evaluation. He had taken one cognitive psychology course, but none in HCI. A1 received graduate-course credit for participating in this analysis.

3.1. The Choice of a Technique

A1 spent 12 hours finding and reading papers about several HCI techniques and reading the Builder specification before deciding which analysis technique to choose. He considered PUMs (Howes & Young, 1991) but rejected it because it required getting the PUM simulation code. He considered heuristic evaluation as defined by Jeffries and colleagues (Jeffries, et. al., 1991) but decided he did not have sufficient UI design experience. He considered using standards or guidelines but rejected it because it meant "a lot of reading though thick books." He read Chapters 5 (Usability Heuristics) and 8 (Interface Standards) in Nielsen's *Usability Engineering* (1993). Despite some doubts about whether he had the prerequisite training, he chose Cognitive Walkthrough (CW) based on the initial HCI methods overview lecture (above) and the summary of the CHI'92 workshop on usability inspection methods (Mack & Nielsen, 1992).

CW is an usability analysis technique similar in rationale and execution to requirements or code walkthroughs. However, CW is based on decades of research in cognitive psychology that help focus the walkthrough issues surrounding the ease of learning a new interface. This method has been evolving since its introduction in 1990 (Lewis, Polson, Wharton, & Rieman, 1990) and it's current form (Wharton, et. al., 1994) was eventually used by A1. Inputs to a CW are a description of the interface, a task scenario, assumptions about the knowledge a user will bring to the task, and the specific actions a user must perform to accomplish the task with the interface. The group, or individual, performing the CW then examines each step in the correct action sequence asking the following four questions. "(1) Will the user try to achieve the right effect? (2) Will the user notice that the correct action is available? (3) Will the user associate the correct action with the effect that the user is trying to achieve? (4) If the correct action is performed, will the user see that progress is being made toward solution of the task?" (Wharton, et. al., 1994, p. 106) If a credible success story cannot be told for each question at each step, then a usability problem has been identified and CW suggests ways of fixing the problem.

3.2. The time course of the analysis

A timeline of the activities of A1 appears in Figure 2. After choosing CW, A1 read several other papers (Bell, Rieman & Lewis, 1990; Lewis, Polson, Wharton & Rieman, 1990; Polson & Lewis, 1990; Rowley & Rhoades, 1992; Wharton, Bradford, Jeffries & Franzke, 1992) before finding Wharton, et. al. (1994) (which at the time was only a University of Colorado technical report, but is now publicly available). In his final report, A1 stated that Wharton, et. al., (1994) is the only reference really needed to learn and use the technique and strongly recommends it to other analysts because of its clear, step-by-step description and concise examples. When he found this paper he read it with special emphasis on learning how to do the technique (total 3 hrs).

A1 then spent a total of 4.5 hours setting up task scenarios with correct action sequences in preparation for doing the walkthrough. To do this, A1 examined the functionality of the Builder and the frequency of elements (e.g., PICT frames, movies, glossary terms) occurring in the Drosophila target document. In doing so, A1 discovered that the Drosophila document did not include several of the features included in the Builder interface document, e.g., it did not include graphical representations of simulation code like

radio buttons for a fixed set of mutually exclusive parameters or slide-bars for changing numerical parameters in a simulation. Noting this deficit in the target document in the diary for later discussion, A1 chose two task scenarios, the first being a short one (creating pps. 1-3 of the target document) to test his understanding of the walkthrough technique, and the second being much longer (creating pps. 24-35; 104 user actions).

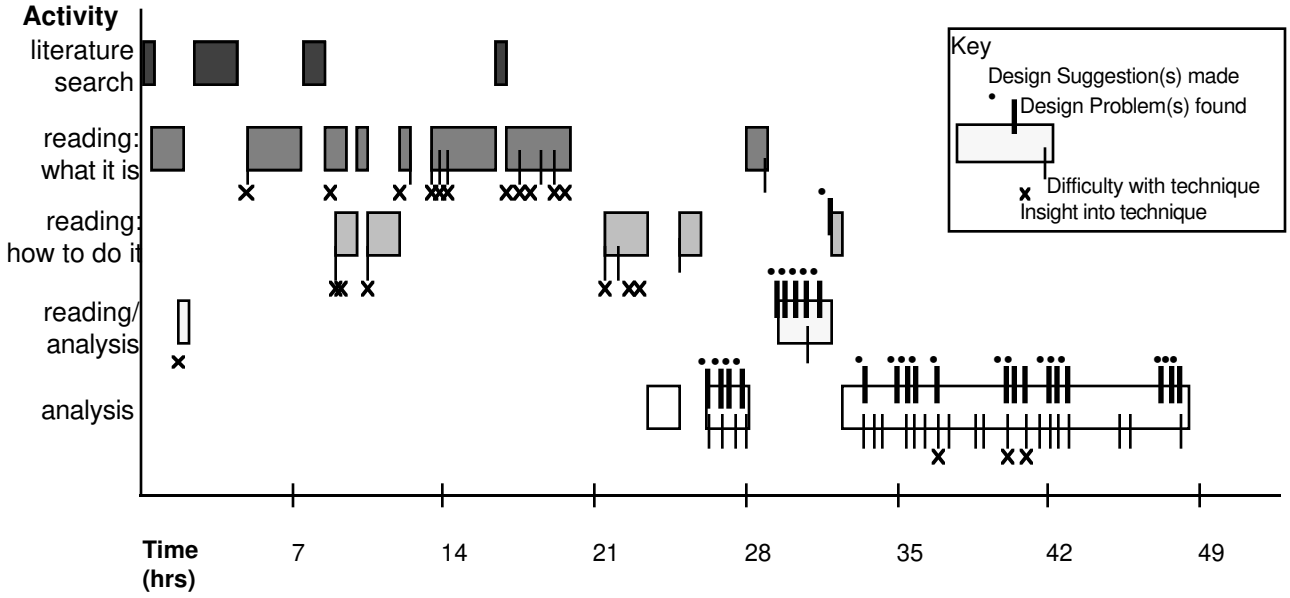

Figure 2. Timeline of the activities of analyst A1 as he learned and applied the Cognitive Walkthrough inspection method to the Builder multimedia authoring tool, as recorded in his diary forms.

A1 then determined the correct sequence of 36 user actions for the short task (1.5 hours) and performed the walkthrough (2 hours). He found six usability problems and six associated design suggestions during this self-imposed practice walkthrough. In addition to usability problems, this short analysis pointed out many gaps in the Builder interface specification. After this, A1 read Wharton and Lewis (1994), which he found interesting, but not useful for applying CW.

A1 then began to apply the CW method in earnest. Since he did not have ready access to designers of the system, he performed the walkthrough himself, rather than in a group. He first prepared the correct action sequence for this longer task (104 actions). He began the walkthrough with frequent reference to Wharton, et. al. (1994), but after 2.5 hours of tightly intertwined reading/analysis and a second reading of the Builder specification, he did not need to refer to the papers anymore. He then spent 16 hrs doing the walkthrough, that is, for each user action he asked the four questions recommended by CW and either wrote a success story or filled out a problem description report for each question. During this phase he reported 36 usability problems and 35 design suggestions. His final report cites seven more general psychology and HCI papers, which his diary does not record as being read during this time period (presumably read during his cognitive psychology course).

3.3. Tasks chosen for the walkthrough

3.4. An example of the content of A1's walkthrough

Figure 3 shows a portion of the action sequence A1 prepared for himself for the second task scenario. He listed both the action a user would have to take to accomplish the task and the system's response for each step.

Figure 4 shows A1's walkthrough of the actions specified in Figure 3. For each action, he enumerated the CW questions and either checked off a question if he considered it a success and recorded the reason for his judgment, or justified why he thought the question indicated a failure of the interface to support learnability. If there was a failure at a action, A1 prepared a PDR (Figure 5) and put the number of the PDR in his CW. In addition, A1 often recorded a suggestion for redesigning the system to prevent the failure.

Action Sequence
Task B: Create 24-35 in Drosophila document

USER ACTION	**SYSTEM FEEDBACK**	
Reopen Drosophila volume and go to p. 23		
3.1.4.1.	Click on Drosophila icon	brings up simulation
3.1.4.2.	Choose "Show Volume" from the "Windows" menu	brings up Drosophila volume with page 1 on top

Figure 3. Sequence of correct actions for a portion of the long task scenario used by A1.

Cognitive Walkthrough
Task B: Create 24-35 in Drosophila document

1 .1 ¸ part of task, Mac experience
.2 ¸ icon visible
.3 ¸ Mac experience
.4 ¸ simulation comes up
success
2 .1 problem here: Mac experience tells you that you immediately get into the application by clicking on an icon representing an instance of this application
.2 ¸ accessible [through] menu
.3 again problem here
.4 ¸ loads Drosophila volume → visible in window
failure → 26

Figure 4. A1's answers to the four CW questions for the actions shown in Figure 3. The arrow pointing to "26" indicates that this failure is reported in PDR #26 shown in Figure 5.

Problem Description Form

Reference number in diary: 26
Brief description of the problem: Starting a volume in builder mode: Necessity to choose "Windows - Show Volume" to get into builder mode with the current volume, contradicts Mac experience that you get into the correct application just by clicking on a file created by this application
How did you find this problem? Using my technique
How frequently will users encounter this problem? Often
How did you estimate frequency? "Other" - [this will occur] each time the builder wants to edit an existing volume
How serious is this problem? Serious
How did you estimate severity? Personal Judgment
Other comments? Design suggestion: Redesign procedure to open an existing volume. 1) click on icon representing the volume, 2) a dialog box comes up asking whether this volume shall be opened in builder or explorer mode, 3) application loads volume and displays first page of it *without any further action.*

Figure 5. PDR#26 reporting the learnability problem identified in the CW of step 2 (Figure 4).

3.5. The usability problems identified

A1 submitted 42 PDRs, all of which were written while doing tightly-coupled reading/analysis or analysis. These PDRs included 46 unique usability problems and 6 duplicate problems (there were 9 PDRs with more than one problem on them) as judged by consensus between the authors[2]. These problems covered all aspects of the Builder application: bookmarks, frames, cross-references, glossary, table of contents, cursor, help system, pages, undo, volume, and the entire application as a whole; but not problems with the Drosophila target document. Of these PDRs, A1 said he found 61% of them directly from using the technique, 12% as a side effect of using the technique, 15% simply from reading the Builder design specification, and 10% from other sources (2% were left blank). Of the "other" reports, the source of these problems were the group discussions.

The PDRs asked for a judgment of how frequently the problem would occur to a user of the Builder. A1 judged that none of the problems he reported would occur only once to a user, 22% to occur rarely, 37% to occur occasionally, 29% to occur often, 10% to occur constantly, (2% were left blank). None of these judgments came directly from using the technique, 39% came from personal judgment, 56% came from "other" sources (5% were left blank). The other source of frequency information was usually the frequency analysis of the target document that A1 had performed when selecting the task scenarios, whereas a few were attributed simply to "common sense."

The PDRs also asked for a judgment of the severity of the usability problem on a scale of 1 to 5, where 1=trivial, 3=moderately serious and 5=must be changed for software to be usable! On this scale, A1 judged none of the problems he found to be trivial problems, 12% to be 2, 29% to be moderately serious (3), 49% to be 4, and 5% must be changed for the software to be usable at all (5% were left blank). Again, none of these judgments were attributed to the CW technique, whereas 95% were personal judgment (5% were left blank).

---

[2] A previous report of these data found 48 usability problems and 4 duplicates (John & Packer, 1995). Subsequent examination of the data resulted in two more PDRs being judged as duplicates. A full list of the usability problems found by this analysis can be found in appendix 1.

In his final report, A1 said that the three most important problems to fix were the icons in the tool palette, the items in the menus, and the relationship of the glossary and table-of-contents panes to the main-media pane of the Builder window. The first two of these problems had many PDRs associated with them (6 and 5, respectively, about 13% of the total number of PDRs for each one). This is not surprising, as two of CW's four questions focus on the availability and meaningfulness of cues in the system, both of which point to issues with icons and menus. The last problem appears in only one PDR. However, A1 justifies the importance of this problem with his frequency analysis of the Drosophila document: glossary terms were the most frequently occurring feature of that document.

3.6. Difficulties with CW and insights into its use

In all, A1 recorded 87 notes in his diary which he labeled either a difficulty with the CW technique or an insight into its use. Figure 2 shows that the difficulties and insights occur not only when initially reading about the technique or when learning how to use it, but throughout the analysis. These notes fall into several major content categories. Also, some concerns disappear as A1 learns more or becomes more familiar with CW, whereas others persist for the duration of the analysis.

3.6.1. Background and training required of the analyst (3 notes).

When reading about the CW technique, A1 was initially concerned about the amount of training necessary for the analyst because of statements made in Lewis, et. al., (1990) and Mack & Nielsen (1992). However, this concern disappeared from his diary notes after the 16th hour. By the final report, A1 was "very comfortable with learning and applying the technique" and asserted that "little or no experience in either user interface evaluation or cognitive psychology is required of the user [of the CW technique]."

3.6.2. Applicability of CW to the Builder application (5 notes).

A1 wondered whether CW would scale up to the complex Builder interface. However, reading Wharton, Bradford, Jeffries, & Franzke (1992) seemed to allay these concerns. In the final report, A1 says he "...wondered whether the assumptions of the walkthrough...would apply to the evaluation of the Volume View. But the assumptions about the user population (novice users with Mac experience) turned out to be consistent with these assumptions and the more recent versions of the CW are not restricted to walk-up-and-use interfaces."

3.6.3. Underlying theory (13 notes).

A1 expressed many initial concerns about the underlying theory of CW. He was concerned about the validity of CE+ (Polson & Lewis, 1990) and what part the CE+ theory played in the actual walkthrough. He was concerned about his lack of knowledge about the underlying theory, and how that would impact his ability to conduct an effective walkthrough. However, these concerns disappear while reading Wharton, et. al. (1994) because A1 reasoned that CW's assumption that the user's actions would largely be guided by the interface was valid for the Builder. In his final report, A1 feels confident in his ability to conduct a walkthrough, and states that "Little or no experience in...cognitive psychology is required... The main strength of the CW is that the technique is very simple and easy to learn."

3.6.4. Task scenarios (13 notes).

A1 had two types of concerns about the task scenarios CW requires. The first concern is with his ability to develop "good" task scenarios. He felt that in order to come up with realistic task scenarios, he would

have to talk to the users of the Builder, but this application was too new to have many users or a history of use, so he had to settle for performing a frequency analysis on the Drosophila example document. He continued to have insights about the choice of task scenarios throughout the analysis process, right up until the very end when he decided that task scenarios for a design tool like the Builder should include modification tasks and recovery from error as well as the create-from-scratch tasks he actually analyzed (this last insight came from the group discussions, brought to the table by the analyst using GOMS).

A1's second concern was about not having the designers accessible to dictate the action sequences for the tasks as the CW papers suggest. This remained a concern throughout the analysis because he believed he needed to know more about the Builder than the specification contained, as evidenced by the fact that his diary contains 23 notes about gaps in the specification. However, A1 turned this problem into a virtue by the final report; he stated, "determining the sequence of correct actions for longer task scenarios can take quite a lot of time; and this can be quite costly if being done by a designer. Therefore, I suggest that the sequence of actions is determined by the evaluator himself. Another reason for this modification is that it puts the evaluator naturally in a situation of learning by exploration. After all, this is the main focus of the Cognitive Walkthrough." A1 recognizes, however, that if the evaluator determines the sequence of actions, he or she must be given an opportunity to clarify questions with the designer.

3.6.5. The process of doing a CW (9 notes).

A1 had several concerns about the process of performing a CW that arose from reading the early research papers about the technique (Bell, Rieman, & Lewis, 1990; Lewis, et. al., 1990; Polson & Lewis, 1990; Rowley & Rhoades, 1992; Wharton, et. al., 1992). He was concerned about filling out structured forms, how to handle design suggestions, and how to record usability problems that were not directly connected to the task scenarios. However, all of these concerns disappeared when A1 read Wharton, et. al. (1994).

During the practice walkthrough, A1 found the process of stepping through re-occurring sub-procedures to be very tedious. To relieve this burden, A1 introduced macros for tasks that occur frequently in the same context in the task scenarios. That is, he would perform the walkthrough for the first occurrence of a sub-procedure at the lowest level of granularity, but for subsequent occurrences, he used a macro to symbolize the detailed steps.

One process concern recorded by A1 was justified when the PDRs were examined closely. Early on, A1 was concerned about how to keep track of many usability problems particularly when the walkthrough is done over an extended period of time. Of the 52 problems reported by A1, we (the authors) agreed that 6 of these problems were duplicates. This amounts to 12% of the problems, but this problem might escalate rapidly with larger systems.

3.6.6. The basic capabilities of the CW method (7 notes)

A1 voiced an early concern, raised in Wharton, et. al. (1994) that CW could not address global design issues. Evidently nothing in his experience with CW helped to quell that concern, for in his final report, A1 reiterates, "through the focus on the narrow path determined by the sequence of correct actions, global design issues are not explicitly addressed." Furthermore, A1 says the focus on correct sequences of actions prevents CW from addressing the ease of recovery from error. His suggestion for explicitly including error-recovery task scenarios is an effort to overcome this problem.

Finally, A1 was concerned that the technique does not provide guidance for rating the frequency and severity of usability problems. Indeed, even if this concerned had not been articulated evidence from the

PDRs is overwhelming. On the 42 PDRs submitted, none of the estimates of either frequency or severity were accredited to the CW technique.

3.7. The quality of A1's walkthroughs

All of A1's products were examined in detail by John Rieman, a developer of the CW method. Rieman's opinion of the work was that "A1's final paper showed an excellent understanding of the...technique...[His] walkthroughs were fairly good...but they had some flaws." (personal communication, 16 Dec 1994) Rieman thought that A1 often failed to recognize steps where the user may not have the right goal (question 1). In addition, Rieman thought A1 was too strict in his failure criterion for question 3. For instance, A1 listed a failure when the user's goal was to create a glossary entry, but the button was labeled "Gloss." Rieman would have called that a successful instance of label-following rather than a failure. The first flaw would miss usability problems in the interface and the second flaw would produce false alarms. Thus, if Rieman's judgments are correct, we expect that the problems associated with A1's strict label-following criteria may not be encountered by actual users of the system, whereas users may have more difficulty than the analysis predicted with figuring out what to do next.

4. Usability tests

We conducted empirical usability tests[3] to discover whether the problems identified by A1 using the CW actually occurred when people used the Builder application and whether other problems occurred that A1's analysis missed. The Builder application had been in development in parallel with our diary-data collection, and much (but not all) of it had been implemented at the time of these tests.

4.1. Method

4.1.1. User Tasks

We gave our users four tasks. The first was to create a three-page multimedia volume, attached to an ObjectPascal program, from text, pictures and an animation in other Macintosh applications. These pages included entries in the table of contents and glossary. The second task was to modify that volume by switching page 2 with page 3. The third was to delete the second page of that document, and the fourth task was to add another page at the end, enter another glossary entry and modify the definition of an existing entry.

4.1.2. Participants

The participants were four[4] undergraduates from Carnegie Mellon University.

The first was a "pilot" session where the experimenter, User1, did the tasks himself. User1 was an experienced Macintosh user who had read the specification for the Builder, but who had never used it before this session. During this session, he uncovered several usability problems. If this were a formal

---

[3] The user tests were performed by Steven Marks, under the supervision of Bonnie John. We thank him for his efforts.

[4] Prior research has shown that using three or four participants in a usability test maximizes the ratio of the number of usability problems found to the effort involved in running the test (Nielsen & Landauer, 1993; Virzi, 1990).

experiment examining hypotheses about the usability of this interface, we would not include User1's data. However, this work attempts to follow good work practices in a development environment and there useful information about the system would not be ignored simply because it was found before the "real" users arrived in the laboratory. Therefore, we include User1's data as well.

The other three users were solicited through an advertisement on campus bulletin boards A basic assumption of the VolumeViewBuilder application is that the users will be Macintosh users, therefore the advertisement specified that the users be proficient with the Macintosh. The advertisement specifically asked for participants who had experience using some powerful word-processor on the Macintosh.

The four participants represented a wide range of Macintosh skill: three had no problems with a pre-task where they had to cut and paste from one application to another but one was not able to do this without difficulty. This range of skill is not unrealistic for the targeted end-users of the Builder.

Participants were paid five dollars per hour for their participation.

4.1.3. Apparatus

The user test was run on a Macintosh Quadra with a 17 inch color monitor, running system 7.5. MSWord 5.1 was used to hold source text and pictures, and MoviePlayer 1.0 was used to hold source animations. HyperCard 2.2 was used in a short preliminary skills assessment. The test used ObjectPascalGenie2.2d55, the developers' version of the ACSE software that had the most developed version of the VolumeViewBuilder at the time. Each session was videotaped at the User Studies Lab in the School of Computer Science at Carnegie Mellon University.

4.1.4. Procedure

Each session began with an explanation of the study; that we were testing a multimedia authoring tool in the early stages of development and that the results of their participation would be fed back to the developers to improve the system. Each participant signed a consent form.

The participant was then given very simple skills assessment tasks, just to demonstrate his or her proficiency with the Macintosh. The participant was shown a screen with an MSWord file open and visible, and told that both MSWord and HyperCard were currently running (HyperCard was not visible on screen). The first skills assessment task was to switch between MSWord and HyperCard. The second task was to copy a graphic (and only the graphic, with no linefeeds or text associated with it) from the MSWord file onto a card in the HyperCard stack. The experimenter then demonstrated how to select an animation in MoviePlayer, as selecting an entire animation may not be a familiar operation for all Macintosh users. After the brief demonstration, the participant was asked to copy the animation into a new MSWord document to demonstrate his or her newly-acquired knowledge of that procedure.

The participant was then given a short training in how to do a think-aloud verbal protocol. The participant was told to think aloud as he or she worked. The experimenter demonstrated thinking aloud while playing a solitaire game on the computer and the participant practiced thinking aloud with the same game.

After these preliminaries, the participant was shown an ObjectPascal program on the computer screen and told that the system they were testing allowed professors to attach course material to simulation programs. This program was a simulation of fruit fly embryo development that a was to have course material attached to it. They were given three hard-copy pages of text, pictures, and an animation (denoted on the hard-copy by standard VCR-like controls beneath a picture) and asked to create a multimedia volume that

looked like these pages and attach it to the program. They were told that the source text, pictures and animation were in other applications already open on their screen.

The participants were NOT given any specific instructions about the Builder. This instructionless learning paradigm represents how many people use new applications (Carroll & Rosson, 1987), especially on a computer platform that advertises its ease-of-use and commonality among applications. It is also the optimal situation for testing predictions of the Cognitive Walkthrough technique, as that technique focuses on how the interface itself guides the user to perform tasks and learn through performing.

After completing this first task, the participant was asked to do the second task; after that, the third, and so on until all four tasks were done. All tasks together took between one and one-and-a-half hours to complete.

4.2. Analysis and Results

The two authors analyzed the tapes separately using the MacSHAPA observational analysis tool (Sanderson, Scott, Johnston, Mainzer, Watanabe, & James, 1994). We compiled a list of usability problems that is the union of both analyses. A PDR was generated when one or more of the following occurred on the videotape.

- The user articulated a goal and cannot succeed in attaining that goal within 3 minutes (then the experimenter steps in and tells him or her what to do).
- The user articulates a goal, tries several things and then explicitly gives up.
- The user articulates a goal and has to try three or more things to find the solution.
- The user does not succeed in a task. That is, when there is a difference between the hard-copy document the user was given and the Volume the user produced.
- The user expresses surprise.
- The user expresses some negative affect or says something is a problem.
- The user makes a design suggestion.
- The analyst sees something in a group of PDRs that he or she can generalize into a more global problem.

Given these criteria, 60 usability problems were found in the videotapes of the four users[5]. No problems were experienced by all users, 5 were experienced by three users, 16 happened to two users, and 39 happened to only one user: not an unreasonable pattern of results for usability studies.

---

[5] A full list of the usability problems observed in these user tapes can be found in appendix 2.

5. Comparing A1's CW to the Usability Tests

5.1. Problems Available for Comparison

To compare the usability problems predicted by A1's CW analysis to those observed in the empirical usability tests, we must first determine what set of usability problems could possibly have been both predicted and observed. That is, some predicted problem might not be observable and some observed problems might not be able to be predicted because of the Builder implementation used at test.

5.1.1. Predicted problems that could be observed

The implemented version of the Builder differed from the specification that A1 analyzed for several reasons. First, only part of the Builder was implemented, so usability problems predicted in those aspects of the Builder that were not implemented could never be observed (16 problems). Therefore, only 30 problems predicted by A1 were even potentially observable.

Second, the implemented Builder differed from the specification because of the usual development process. That is, as the developers implemented the system, they discovered places where the specification was ambiguous or incomplete and made the decisions necessary to complete the code. Alternatively, as they coded, they discovered interface procedures or functions that they thought were better than those in the specification, discussed the new ideas with the project team, and implemented those instead. A1 predicted 12 problems in areas of the Builder specification that did not perform as specified at the time of test, so those problems could not be observed. So, of the 30 problems that A1 predicted, 40% were designed away in the normal development process.

Thus, the Builder application actually implemented provided the opportunity for users to experience 18 problems predicted by A1.

5.1.2. Observed problems that could be predicted

Of the 60 problems observed in the user tests, many could not be predicted by a CW. For instance, actual runtime bugs in the code that caused the system to crash or cause spurious highlighting of text, etc. could never be predicted by a pre-implementation analysis technique. Our participants experienced two such runtime bugs (3%).

Also, since the developers did not implement the Builder to the original specification, but changed the system in the course of normal development, any analysis done on an initial, static specification could not predict usability problems with totally new features or procedures. Twenty-three of the 60 usability problems observed (38%) occurred in areas where the implementation did not match the original specification sufficiently to allow prediction. Thus, the system developers introduced usability problems that were not in the specification, perhaps just as many as they removed.

This leaves 37 usability problems out of the observed 60, that could have been predicted by A1's CW, in principle. However, sometimes this prose specification was incomplete or ambiguous, and the developers filled in the details as they coded. Of the 37, 16 are explicitly in the specification, 3 can be assumed from a knowledge of Macintosh™ standard interfaces (which was stated as the default underlying assumption of the specification), 6 are implied by the specification but not explicitly discussed, 8 are incomplete or ambiguous in the specification, and 4 are not mentioned in the specification at all. It is unclear whether an HCI usability analysis technique like CW should find such problem that are implied, not fully specified, or omitted from a specification. In fact, some usability analysis techniques are specifically designed to make

a user interface specification complete (e.g., the User Action Notation specification language, Hartson, et. al., 1990). Such problems range from missing functionality, e.g., that making a glossary entry does not automatically search the document for other occurrences of the word and make links from those occurrences into the glossary, to lower-level user-feedback issues, e.g., the highlighting of the table of contents item does not always track the user's navigation through the volume. Since prose specifications are very common for user interfaces, usability inspection techniques are designed to be used with prose specifications, and these problems are unavoidable with a natural-language specification, then CW should be able to identify such problems.

In the remainder of this discussion, we will compare these 37 observed usability problems to the 18 problems predicted by CW that we considered potentially observable above (Figure 6).

5.2. Hit's, Misses & False-Alarms of A1's CW analysis

5.2.1. Problems predicted by A1's CW and observed in the user tests

Of the 37 observed usability problems that could have been predicted by A1's analysis, 2 were predicted precisely (5%), and two were predicted vaguely by a single CW PDR (5%). (Since the specifics of the vaguely-predicted observed problem was not specifically predicted by A1, we will count them as misses in the next section as well.)

An accurate prediction made by A1 was that users would not be able to associate the "Windows" menu and its "Show Volume" item with their goal of opening a new volume in which to put the text, pictures and animation (CW-39 in Figure 6). Indeed, three of the four users had so much difficulty accomplishing this first step in the task that the experimenter had to stop them after 3 minutes and give them the answer. A1 also suggested that the item be called "Open Volume" and it be under the "File" menu, which matches precisely what the users were saying as they all searched that menu first.

A1's walkthrough also predicted that users would have difficulty finding the menu item to add a page to the volume. Indeed one user tried scanned every menu several times looking for some item to add a page and did not see this command although he stared at the Windows menu for over a second. He eventually found the item after reading each menu carefully two and three times.

Lastly, A1 predicted that users would be confused about which actions to find in the palette and which to find in the menus (CW-1). This is a rather vague PDR that does not point to a particular problem. User2, User3 and User4 routinely explore both buttons in the palette and many menus when they did not know what to do, indicating that they have no clear idea about the division of functions. We have credited the CW with predicting two observed usability problems with this sweeping PDR, but we believe this is generous given the weak evidence. It is possible that A1's concern might have caused designers to completely re-think the menus and palette, had they considered it early in the design process.

5.2.2. Problems not predicted by A1's CW but observed in the user tests

In contrast to the small number of problems predicted above, A1's CW missed 35 of the 37 observed, but potentially predictable, usability problems (95%). Figure 6 classifies all of the observed usability problems by the CW question that could have predicted them (indicated by dots in the appropriate column). Sixteen pertain to Question 1, whether the user will have the right goal at the right time. Five pertain to Question 2, whether the action to accomplish that goal is available in the interface. Three pertain

to Question 3, whether the user will associate the goal with the action. And six pertain to Question 4, whether the user will get feedback that he or she is making progress if they take that action. Seven do not seem to fall into these categories, as will be explained below. As discussed above, two of these problems were predicted precisely by the CW and two were predicted so vaguely that we are considering it unpredicted for this part of the analysis (indicated by parentheses around the x's and CW ids).

Seven of the problems found by users do not seem to fall within the province of the CW technique. For instance, U-01 involves the system interrupting the user to ask for a name for an auto-save file (Sys in Figure 6) which was only implied by the specification's reference to the auto-save feature. Problem U-03 was a performance issue (Perf, in Figure 6); the user complained that scrolling was slow, something CW does not address. We have classified four problems as "unfulfilled expectations" (UnEx). That is, the situation was a CW success story for all four standard questions, but the user expected the system to do more than it actually did: be able to search for text (U-03), find all occurrences of glossary terms (U-37), update the glossary when whole pages are removed (U-38), and ignore white-space in glossary entries (U-41). Finally, we identified an "analyst inferred user need" (Anal) for page-layout assistance; the users did not articulate a need for this feature, but we inferred it from the large amount of time the users spent fussing with the size and position of frames on the page. It is unlikely that the Cognitive Walkthrough could ever predict performance problems, or the myriad of things that users will attempt to do or expect the system to do for them. However, CW might be able to help an analyst infer user needs from highly repetitive or seemingly meaningless activities (like fussing with the size and placement of frames or visually searching for a word). We will return to this discussion in section 5.3.

It is more difficult to explain why the other 28 problems found by the users were not predicted by A1's CW. After examining the predictions A1 made that were not evident in the users' behavior, we will return to this question, bringing other data from the case study to shed light on the problem.

5.2.3. Problems predicted by A1's CW but not observed in the user tests

Of the 18 problems A1 predicted, 15 were not reported in PDRs made through observation of the user tests. However, there are three categories of these unobserved problems. The first are not actual false alarms, rather, they are an artifact of the usability test itself. The second involves a series of predictions based on button-appearance that was slightly changed by the developers, but was so close to the original specification that we decided to track their occurrence in the user tests. Finally, the third are true false alarms, as A1's CW predicted that users would have problems and they didn't.

Seven CW PDRs fell into the first category because the task we gave to the users did not require them to exercise the features discussed in the PDRs. Three of these had to do with saving the file in different ways (execute-only so that the code associated with the volume cannot be changed, CW-3; text-only, CW-4; and for end-user-use only so the frames in the volume cannot be changed, CW-36), and the task did not ask the user to save the file in any of these exotic ways. In fact, the auto-save feature was active by default and the users recognized that their file was being saved automatically, so they never brought up a save dialog box and never even saw the options for exotic saves. Three CW PDRs predicted that manipulating different windows (views) would be difficult (CW-5, CW-6 and CW-7), but we did not have multiple windows within the Builder application in the users' task so there was no opportunity to observe these problems. Finally, A1 predicted that users would want to undo sequences of actions, not just one (CW-35). Although our users never articulated this desire, our tasks may not have been complex enough to warrant the feature. These artifacts of the user task remind us that usability tests themselves must be carefully designed and should not be taken as identifying all possible problems (usually just those that could be observed in a single hour of a novice-user's time!).

User No.	Feature	Description	No. users found	CW No.	CW Question 1	CW Question 2	CW Question 3	CW Question 4	Other Source	Omission	Commission	Local	Global	Fund. concept	error-recovery
U-01	Application	Save dialog comes up repeatedly until user names document	2						Sys		•	•			
U-02	Application	More than a single object appears to be active at any time	2			•					•		•		
U-03	Application	There is no text search facility	2						UnEx	•			•		
U-05	Application	Scrolling is very slow	1						Perf		•		•		
U-06	Application	Can't set default font size	1		•					•		•			
U-08	Application	User expects Builder to work like a word processor in many ways (e.g., automatically add pages, arbitrary text in the ToC).	1		•						•			•	
U-12	Bookmarks	User doesn't see bookmark appear after hitting bookmark button	1					•			•	•			
U-13	Frames	User doesn't know to make a frame first before pasting in content (e.g., text, movies)	3	(CW-1)	•		(x)					•			
U-15	Frames	Frames won't drag between pages	2		•					•		•			
U-17	Frames	There is no support for page layout (like rulers, alignment, centering, etc.)	2						Anal	•			•		
U-22	Frames	User wants a command key for removing frames	1			•				•		•			
U-24	Frames	User does not recognize the need to make and manipulate frames even when closely examining the frame tools	1		•						•			•	
U-30	Frames: Text	Should share common function keys with other text editors	1			•				•			•		
U-31	Frames: Text	Cannot change the font or size of text in a whole frame by selecting the frame and issuing font or size command	1		•					•		•			
U-32	Frames: Text	User does not know that formatting can be done inside the Genie	1			•					•	•			
U-33	Frames: Text	User wants to make the blinking cursor "get out" of the frame	1		•						•			•	
U-34	Frames: Text	User does not recognize that "format" "font" and "size" refer to properties of text (Distinction between roles of palette & menu is unclear)	1	(CW-1)			•(x)				•			•	
U-37	Glossary	Glossary doesn't find other occurrences of word	2						UnEx	•		•			
U-38	Glossary	Removing a page with a glossary entry does not remove the glossary entry from the glossary pane	2						UnEx	•		•			
U-40	Glossary	Can't edit a word in text frame that is already in the glossary	1		•						•	•			•
U-41	Glossary	Glossary function doesn't strip off white space	1						UnEx		•	•			•
U-43	Glossary	Can't drag & drop entries into glossary pane	1		•					•		•			
U-44	Pages	Remove-page removes the page with a visible, selected frame, no matter how slightly it is visible.	3					•			•	•			
U-45	Pages	User expects Add-Volume-Page to add the page at the end of the volume, not after the page with a selected frame	2					•			•	•			
U-46	Pages	Add-page should scroll to the page just added	2					•			•	•			
U-47	Pages	Can't adjust page-breaks	1		•						•			•	

User No.	Feature	Description	No. users found	CW No.	CW Question 1	CW Question 2	CW Question 3	CW Question 4	Other Source	Omission	Commission	Local	Global	Fund. concept	error-recovery
U-48	Pages	Can't re-order pages except through brute force reconstruction	1		•					•		•			
U-49	Pages	User does not understand the concept of a "page" in the volume.	1		•						•			•	
U-50	Pages	User wants a command key for making a new page	1			•				•		•			
U-51	Pages	Several operations act on the last visible page rather than the dominant page (bookmark, remove-page).	1					•			•		•		
U-52	Pages	Can't find the Add-Volume-Page menu item in Windows menu	1	CW-29			•x				•	•			
U-53	ToC	Can't edit the ToC entry directly	2		•					•		•			•
U-54	ToC	Only one ToC entry per page	2		•						•	•			•
U-56	ToC	Selecting a ToC frame in the text pane does not always update the ToC pane's highlighting	1					•			•	•			
U-57	ToC	User can't figure out how to "make" the ToC	1		•						•			•	
U-58	ToC	User expects ToC to allow arbitrary text, not only all of a frame	1		•						•	•			
U-60	Volume	Users expect "Open Volume" in File menu, not "Show-Volume" in Windows menu to access the volume	3	CW-39			•x				•	•			
	Application	Save as: "Execute only" is irrelevant	0	CW-3	x				NTO		•	•			
	Application	Save as: "Text only" is irrelevant	0	CW-4	x				NTO		•	•			
	Application	Different views of the file is unnatural	0	CW-5	x				NTO		•			•	
	Application	Windows menu name will not be associated with showing different views; change to "Show"	0	CW-6			x		NTO		•	•			
	Application	"Bring to Front" will not be associated with showing a window; Change to "Show as Front Window"	0	CW-7			x		NTO		•	•			
	Frames	Users will expect frames to be created with a default size	0	CW-10	x				FA	•		•			
	Frames	Clipboard does not indicate what type of information is ready to paste	0	CW-11				x	FA	•		•			
	Frames	Move-frame icon will not be meaningful to users	0	CW-13			x		FA~		•	•			
	Frames	Create-frame icon will not be meaningful to users	0	CW-14			x		FA~		•	•			
	Glossary	Re-sizing bar on glossary pane not recognizable as an action	0	CW-26		x			FA		•	•			
	ToC	"TOC" will not be associated with "Table of Contents"	0	CW-31			x		FA~		•	•			
	Undo	Can only undo one action, would like to undo a sequence of actions	0	CW-35	x				NTO	•			•		
	Volume	Saving to explorer can't be undone	0	CW-36		no credit to CW			NTO	•		•			
	Volume	Page-up/page-down icons not meaningful	0	CW-37			x		FA~		•	•			
	Volume	"Go to Page" item in Find menu too far down	0	CW-38			x		FA		•	•			

Figure 6. The potentially predictable usability problems found in the user videotapes and the potentially observable usability problems predicted by A1's CW.

Other Source of User Problems: Sys=goal introduced by system. UnEx=Unfulfilled expectations. Perf=system performance issue. Anal=inferred by analyst.

Status of Unobserved Predicted Problems: NTO = no opportunity in the tasks. FA = false alarm. FA~ = false alarm as tested, but spec was slightly different.

A1 predicted that users would have problems with five buttons: create-frame (CW-13), move-frame (CW-14), the page-up/page-down buttons (CW-37), and ToC (table of contents) (CW-31). He said that the pictorial icons were not meaningful for the first four and the label was meaningless for the last. The developers made very slight changes in the course of implementation to each of these buttons, so we examined the users' behavior to see if problems occurred. The users seemed to have no problems using these buttons. However, since the predictions were made on slightly different buttons than the users saw, we cannot count these as verified false alarms, rather, just as an indication that problems with icons and labels might not be as severe as A1 thought.

Finally, 4 of the 15 unobserved CW PDRs seemed to be actual false alarms (27%). Our usability tapes have explicit evidence that users found it quite easy to size a frame when creating it (CW-10), paste all types of information into frames (CW-11), resize the glossary pane (CW-26), and find the "Go to page" menu item (CW-38).

5.3. Explaining the walkthrough's effectiveness.

Although this case study displays Cognitive Walkthrough as a technique that is learnable and usable by an analyst not trained in psychology or HCI (John & Packer, 1995), it is shown here to be disappointing in its ability to predict actual user problems. A1 predicted about 5% of the problems, missing about 95%, with about 25% verifiable false alarms. We look now to some of the detailed information collected in our case-study to ask why this effectiveness came about. In particular, we can look at the content of the PDRs, impressions that A1 had about his own analysis in the diaries and the walkthroughs themselves to help form hypotheses about why some problems were predicted and others weren't. We also have an independent assessment of the quality of A1's walkthroughs by one of the developers of the CW technique, John Rieman, which may provide even more insight into the misses and false alarms. In this section, we will take several different cross-sections of the data to help explain how the misses and false alarms might be avoided in future walkthroughs.

5.3.1. Sins of commission or omission

One hypothesis for the large percentage of misses is that an inspection method like the CW might only be able to detect usability problems of *commission* rather than *omission*. That is, since the CW works from a specification of the system and tasks defined by the analyst, perhaps it can detect usability problems committed by the developers' specification of the interface, but not problems with features omitted by the developers. Users on the other hand, have prior knowledge and their own goals beyond what the developers or analysts think of and therefore think-aloud studies uncover omissions as well as commissions. If this hypothesis were true, then there would be a preponderance of unpredicted usability errors attributable to omission of a capability, and A1's predictions would not predict such omissions.

There is little evidence supporting this hypothesis. A1 predicted 4 problems of omission out of his 18 predictions, two of which were verifiable false alarms and two of which were not called for by the task. Three problems (CW-10, CW-11 & CW-35) came directly from a walkthrough, but the other PDR (CW-36) was not attributed to the CW technique directly by A1. Rather, he said he used his own experience instead of the CW technique to identify that problem. In all cases, A1 brought his own prior knowledge to bear on the problem and could indeed see potential problems of omission, so it is the case that an analyst using CW can predict problems of omission.

However, of the 37 observed usability problems, 13 are problems of omission and none of these were predicted by A1's CW, so, in practice, these types of problems were indeed missed by the analysis. Perhaps such problems are identified only through idiosyncratic prior knowledge or goals belonging to a single user, and no analyst's prior knowledge could be expected to produce any one user's problems of omission. If this were the case, then each of the problems of omission would have occurred to only one user during the study. Again, the data does not support this hypothesis, as 6 of the 13 problems of omission were found in the behavior of two users (46%), which is substantially higher than the percentage of problems of commission that occurred to multiple users (29%). Therefore, it seems that separating user problems into commission or omission does not help us identify the source of the misses in A1's analysis.

5.3.2. Did A1's doubts manifest themselves?

A1 had several doubts about the CW technique that were unresolved at the end of the his analysis. Can any of these doubts account for the misses or false alarms in his results?

A1 was concerned about his ability to pick task scenarios that would find all the usability problems. He used the target document given to him as a source for the tasks. In fact, he did a CW for each step in creating 15 pages of the 55-page target document, and added some navigation and modification tasks that changed glossary entries and deleted and added pages. But he was still uncomfortable with his task selection. In his final report, he suggested including tasks that started from where a user had made an error, to check the procedures for error-recovery.

Given A1's degree of uncertainty, perhaps his choice of tasks to walkthrough caused him to miss some usability problems. Of the observed usability problems that A1's CW missed, his page-creation and page-modification tasks included the opportunity to discover 31 of the 37 of the observed usability problems. On the other hand, he did not include error-recovery tasks in his walkthrough. The users made many errors in their work and error-recovery led to 4 usability errors that were not discovered with errorless performance (11% of the observed problems). So, although including error-recovery tasks is a good idea in principle, pragmatically, it is likely to produce only modest improvement in a CW's hit-rate.

A second concern that A1 carried throughout his experience with CW, was that the technique focused on such detailed procedures that it may not be able to detect global problems (also discussed in Wharton et. al, 1994). To explore this concern, we classified the observed usability problems into *local*, where the problem involved a single localized procedure or user goal, *global*, where the problem would occur with in several procedures or relevant to several types of goals, or *conceptual*, where the problem seemed to have deeper roots (i.e., the developers and users had different models of the way the Builder worked). If A1's concern identifies an underlying cause of CW missing problems, then there should be a preponderance of missed problems in the global and conceptual categories. There is some evidence for this hypothesis. The usability tests also concentrated on local problems (24 out of 37 were local) and the CW also predicted non-local problems (3 out of 18 could be considered non-local). The ratio of non-local misses (37%) is more than twice the ratio of A1's non-local predictions (17%), so finding a way to focus on more global issues would help the CW technique.

5.3.3. Did A1's modification to the technique help or hurt?

A1 introduced the use of macros to reduce the tedium involved in walking through repetitive tasks. His macros were called CREATE-FRAME and CREATE-GLOSSARY-ITEM. The first highlights the basic orientation of the Builder: it is a frame-based editor. The second indicates that creating many glossary items can be very repetitive. It is possible that when an analyst feels the need to use a macro, it may be an

indication of a more global issue in the interface. For instance, although there is nothing inherently wrong with a frame-based editor, the CREATE-FRAME macro highlights the philosophy, pointing the analyst to the question: Will the target users have the same philosophy? Two observed problems stemmed from users not having this philosophy (U-08 & U-24), both of which are global problems. Likewise, the need to use the CREATE-GLOSSARY-ITEM might focus the analyst on the question: Why is this procedure so repetitive? Relieving the tedium of this procedure was also noticed by the users, as two of them wanted the glossary to find other occurrences of the word and make links automatically (U-37). Thus, A1's macros might have masked the glossary problem in this walkthrough, but if analysts add these two questions each time a macro is used, it might help detect some global issues as was suggested above.

5.3.4. Were an expert's intuitions about A1's CW correct?

The preceding discussion indicates that A1's intuitions about his own walkthrough process do not fully explain why his CW was not as effective as we would hope an usability inspection analysis would be. Perhaps this is because A1 was a novice with the technique. John Rieman, one of the developers of the CW technique examined A1's working notes and final report and provided us with his intuitions as to its effectiveness (John Rieman, personal communication, 16 December, 1994), and perhaps those impressions explain the misses and false alarms.

Rieman thought that A1 was too lenient in his judgments that users would intuitively know the next sub-goal in a procedure. For instance, Rieman wondered how a user would know that *first* a title must be created in a frame and *then* the frame can be entered into the Table of Contents. Therefore, Rieman predicted that users would have more failures with the first question of the CW than A1 recorded. There is some evidence that this was indeed the case. Of the 37 observed problems, 16 of them were in the realm of CW's first question, and none of these were predicted by A1's CW. These problems represent almost half (46%) of the unpredicted problems.

Rieman also thought that A1 was too strict in his definition of potentially successful "label-following." Rieman said

> "[A1] often assumed the user had exactly the right goal ("Add a Glossary Term"), then identified a problem because there wasn't an exact label match ("Gloss" instead of "Glossary"). We're looking for a match between a concept and a label, not a word and a label." (John Rieman, personal communication, 16 December, 1994).

Thus, Rieman predicts that A1 would produce false alarms with respect to finding the right menu, menu-item, or icon. Indeed, two of the four verifiable false alarms were attributed to a failure in label-following, and the additional pseudo-false alarms were all in the label-following category.

Thus, it seems that an expert's inspection of a CW can identify problems with a novice analyst's process fairly well. This suggests that CW experts have more knowledge about the process than is codified in the existing publications, suggesting in turn, that perhaps personal training, some feedback, or improved documentation is likely to improve the effectiveness of a novice's walkthrough.

6. Summary and Conclusions

This detailed analysis of the content and process of A1 evaluating a multi-media authoring tool has provided several interesting lessons about the learnability, usability and effectiveness of the Cognitive Walkthrough technique, and some are about the interactive-system development process. Because this is a case-study, however, it is important to remember that these lessons take the form of hypotheses, to be

investigated through further research, both through numerous converging case-studies and through more controlled investigations. In particular, this was a case-study of an individual doing a CW, whereas the preferred method is for the CW to be done in a group. Keeping those cautions in mind, the lessons learned are as follows.

About the Cognitive Walkthrough technique:

- The analyst felt that the CW technique was learnable and usable in a reasonable amount of time despite his lack of training in HCI or psychology, **BUT,**
- The CW analysis predicted only about 5% of the observed usability problems, missed about 95%, and had a false-alarm rate of about 25%.
- Neither features of the usability problems themselves, nor the analyst's intuitions about deficiencies in his walkthrough, fully explain why the analysis was disappointingly ineffective, **BUT,**
- Several suggestions come from A1's experience:
    (1) Use error-recovery tasks in the walkthroughs as well as creation and modification tasks.
    (2) Use macros when needed, but ask the questions "What does this say about the philosophy of the system and will the target users have the same philosophy?" and "Why is this procedure so repetitive?" to help focus on more global issues. **AND,**
- A CW-expert's intuitions about the analyst's walkthrough were accurate enough to suggest that improvements to the technique and/or training materials for the technique could improve the effectiveness of a novice analyst, by:
    (1) Helping to identify realistic user-goals.
    (2) Understanding how users follow labels.

About the interactive-system development process:

- About 40% of the problems predicted from the original specification were not evident in the implemented application because the developers changed their minds in the course of development. Likewise, almost 40% of the observed problems were not in the original specification. Thus, developers introduce problems as well as design them away in the normal development process. Any usability evaluation technique needs to be kept up with changes in the product, and the development process needs to allow for such updates.
- The results from a standard think-aloud usability test (the most commonly-used method of evaluation in the software-development industry today), should not be viewed as a gold-standard, because they are as dependent on choice of task as any inspection method.

Finally, this case study approach used many different types of information to converge on a picture of the process A1 went through to produce his analysis. This story itself can help designers learn and use CW by providing realistic expectations about the process, the difficulties and insights, and the length of time it takes to perform an analysis. Such expectations are missing from a textbook description of a technique,

and yet are a valuable part of any decision to embark on incorporating a new usability analysis technique into the interactive-system development process.

In the development of increasingly powerful and accessible information systems, predicting usability will become even more pragmatically and economically important. This case-study shows that one predictive evaluation technique, Cognitive Walkthrough, is not yet ready for every development effort, but points along a path to being a useful tool in system design.

Acknowledgments

We thank Robin Jeffries for providing her original PDR form and helpful suggestions, "A1" for his thoughtful and thorough CWs, and John Rieman for his evaluations of the CWs discussed herein.

References

Bell, B., Rieman, J., and Lewis, C. (1990) Usability Testing of a Graphical Programming System: Things We Missed in a Programming Walkthrough. In CHI'90 Proceedings, Seattle, WA, ACM, NY.

Butler, K, Jacob, R. J. K., & John, B. E. (1995) *Introduction and Overview of Human-Computer Interaction.* Tutorial materials, presented at CHI, 1992 (Monterey, California, May 3- May 7, 1992), INTERCHI, 1993 (Amsterdam, The Netherlands, April 24 - April 29, 1992), CHI, 1994 (Boston MA, April 24-28, 1994) and CHI, 1995 (Denver CO, May 13-18, 1995) ACM, New York.

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

Carroll, J. M. and Rosson, M-B. (1987) The paradox of the active user. In J M. Carroll (Ed.) *Interfacing Thought.* Cambridge, MA: MIT Press.

Cuomo, D. L. & Bowen, C. D. (1994) Understanding usability issues addressed by three user-system interface evaluation techniques. *Interacting with Computers,* 6, 1, 86-108.

Desurvire, H. W. (1994) Faster! Cheaper!! Are usability inspection methods as efficient as empirical testing? In Jakob Nielsen and Robert L. Mack (eds.) *Usability Inspection Methods.* New York: John Wiley.

Gallagher, S. & Meter, G. (1993) *Volume View Interface Design.* Unpublished Report of the ACSE Project, School of Computer Science Carnegie Mellon University, May 4, 1993.

Gong, R., & Kieras, D. (1994). A Validation of the GOMS Model Methodology in the Development of a Specialized, Commercial Software Application. In *Proceedings of CHI*, 1994, Boston, MA, USA, April 24-28, 1994). New York: ACM, pp. 351-357.

Gray, W. D., John, B. E., & Atwood, M. E. (1993) Project Ernestine: Validating a GOMS analysis for predicting and explaining real-world task performance. *Human-Computer Interaction, 8,* pp. 237-309.

Hartson, H. R., Siochi, A. C., & Hix, D. (1990). The UAN: A user-oriented representation for direct manipulation interface development. *International Journal of Man-Machine Studies*, 31, 477-494.

Howes, A., and Young, R.M. (1991) Predicting the Learnability of Task-Action Mappings. In *Proceedings of CHI, 1991*, New Orleans, LA, ACM, NY.

Jeffries, R., Miller, J.R., Wharton, C., and Uyeda, K.M., (1991) User interface evaluation in the real world: A comparison of four techniques, In *Proceedings of CHI, 1991*, New Orleans, LA, ACM, NY.

John, B. E. & Kieras, D. E. (1994) The GOMS family of analysis techniques: Tools for design and evaluation. Carnegie Mellon University School of Computer Science Technical Report No. CMU-CS-94-181 (also appears as the HCI Institute Technical Report No. CMU-HCII-94-106).

John, B. E. & Packer, H. Learning and using the Cognitive Walkthrough Method: A case study approach. In *Proceedings of CHI, 1995* (Denver, Colorado, May 7-11, 1995) ACM, New York.

Karat, C.-M. (1994) A comparison of user interface evaluation methods. In J. Nielsen and R. L. Mack (eds.) *Usability Inspection Methods.* New York: John Wiley.

Lewis, C., Polson, P., Wharton, C., & Rieman, J. (1990) Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *Proceedings of CHI, 1990*, Seattle, WA, ACM, NY.

Mack, R., Nielsen, J. (1992) Usability Inspection Methods: Summary Report of a Workshop held at CHI'92.. IBM Technical Report #IBMC-18273. IBM T.J. Watson Research Center, Yorktown Heights, NY.

Nielsen, J., (1993) *Usability Engineering*, San Diego: Academic Press Inc.

Nielsen, J. & Landauer, T. K. (1993) A mathematical model of the finding of usability problems. In *Proceedings of INTERCHI, 1993* (Amsterdam, April 24 - April 29), ACM, New York.

Nielsen, J. (1994) Heuristic evaluation. In J. Nielsen and R. L. Mack (eds.) *Usability Inspection Methods.* New York: John Wiley.

Pane, J.F., & Miller, P.L. (1993). The ACSE multimedia science learning environment. Proceedings of the 1993 *International Conference on Computers in Education*, Taipei, Taiwan.

Polson, P.& Lewis, C. (1990) Theory-based design for easily learned interfaces. *Human Computer Interaction,* 5, 191-220.

Rieman, J. (1993) The diary study: A workplace-oriented research tool to guide laboratory efforts In Proceedings of INTERCHI', 1993 Amsterdam, ACM, NY.

Rowley, D. E., & Rhoades, D. G. (1992) The Cognitive Jogthrough: A fast-paced user interface evaluation procedure. In *Proceedings of CHI, 1992*, Monterey, CA, ACM, NY.

Sanderson, P. M., Scott, J. J. P., Johnston, T., Mainzer, J., Watanabe, L. M., & James J. M. (1994). MacSHAPA and the enterprise of Exploratory Sequential Data Analysis (ESDA). *International Journal of Human-Computer Studies*, 41, 633-68.

Virzi, R. A. (1990) Streamlining the design process: Running fewer subjects. In *Proceedings of the Human Factors Society 34th Annual Meeting*. pp. 291-294.

Wharton, C., Bradford, J., Jeffries, R., Franzke, M. (1992) Applying Cognitive Walkthrough to more complex user interfaces: Experiences, issues and recommendations. In *Proceedings of CHI, 1992,* Monterey, CA, ACM, NY.

Wharton, C., Lewis, C. (1994) The role of psychological theory in usability inspection methods. In J. Nielsen and R. L. Mack (eds.) *Usability Inspection Methods.* New York: John Wiley.

Wharton, C., Rieman, J., Lewis, C., & Polson, P. (1994) The Cognitive Walkthrough Method: A practitioner's guide. In J. Nielsen and R. L. Mack (eds.) *Usability Inspection Methods.* New York: John Wiley.

Yin, R. K. (1994) Case study research: Design and methods (*2nd ed., Applied Social Research Methods Series Vol. 5)*. Thousand Oaks, CA: Sage Publications.

Appendix 1

The following table lists the 46 Usability Problems found by A1's Cognitive Walkthrough.

It gives a brief description of the problem and the actual text of A1's Problem Description Report. In addition, this records whether the feature had been implemented as specified, the design had changed, or was not yet implemented at the time of test. Whether the tasks given to the user provided an opportunity to discover this problem and the rational behind these judgments are also recorded.

No.	Feature	PDR text	Summary Description	Implemented	Design changed	Not yet implemented	Task opportunity	Reasons for categories
CW-1	Application	PDR#05: User will be confused what actions to find in button palette and what in the (augmented) standard MAC menu. (Fig 13, p 17, V. V. Spec)	Distinction between roles of palette & menu is unclear	•			+	Task = + because using palette and menus is inherent in any task
CW-2	Application	PDR#06: Fig 1, p 4, Volume View Spec: "Save" Volume View document dialog box: confusing. • What does "Everything" mean • Execute only, text only irrelevant => remove those [we split into three PDRs]	Save as: "Everything" is meaningless		•			Design Change: Dialog box no longer has an "everything" choice
CW-3	Application	PDR#06: Fig 1, p 4, Volume View Spec: "Save" Volume View document dialog box: confusing. • What does "Everything" mean • Execute only, text only irrelevant => remove those [we split into three PDRs]	Save as: "Execute only" is irrelevant	•			-	Task = - because users don't have to save
CW-4	Application	PDR#06: Fig 1, p 4, Volume View Spec: "Save" Volume View document dialog box: confusing. • What does "Everything" mean • Execute only, text only irrelevant => remove those [we split into three PDRs]	Save as: "Text only" is irrelevant	•			-	Task = - because users don't have to save
CW-5	Application	PDR#29: Switching between different views of the volume (volume view / code view / ...) is unnatural / not straight forward at the least. I don't have an idea how to circumvent this problem though.	Different views of the file is unnatural	•			-	Task = - because users don't have multiple windows or views to manipulate
CW-6	Application	PDR#25: Suggestion: "Windows" menu rename to "Show" (label following!) and rename subentry "Bring to Front" to "Show as Front Window" and put "Show Volume" as "Open Volume" under the "File" menu (draw on Mac Experience!) [we split into 3]	"Windows" menu name will not be associated with showing different views; change to "Show"	•			-	Task = - because users don't have multiple windows or views to manipulate
CW-7	Application	PDR#25: Suggestion: "Windows" menu rename to "Show" (label following!) and rename subentry "Bring to Front" to "Show as Front Window" and put "Show Volume" as "Open Volume" under the "File" menu (draw on Mac Experience!) [we split into 3]	"Bring to Front" will not be associated with showing a window; Change to "Show as Front Window"	•			-	Task = - because users don't have multiple windows or views to manipulate
CW-8	Bookmarks	PDR#31: Deleting Bookmarks: Edit - Clear contradicts Mac experience - user will expect to find "Delete" in "Edit" menu.	Using "Clear" in edit menu to delete a bookmark will be confusing		•			Design change: Bookmark button is a toggle. Can't use Clear to delete a bookmark.

No.	Feature	PDR text	Summary Description	Implemented	Design changed	Not yet implemented	Task opportunity	Reasons for categories
CW-9	Cursor	PDR#09: Fig. 4, p. 9 design document: Why not the regular text editing mouse point over code (|).	Why not text-editing | over code?			•		Not implemented: Runtime frames
CW-10	Frames	PDR#01: Creating of frames: [in spec = no default size, starts at nothing] From Mac experience one would expect that there are 2 separate actions: 1. place frame -> pops up in standard size with handle bars. 2. resize it by dragging on the handle bars.	Users will expect frames to be created with a default size	•			+	Task opportunity = + because users must create frames
CW-11	Frames	PDR #02 &#22: Importing from other applications (e.g. PICTs) will be confusing, since user has no feedback as to whether the appropriate thing is currently in the clipboard, ready to be pasted into the chosen frame. (p.26 V.V. spec, 1st paragraph)	Clipboard does not indicate what type of information is ready to paste	•			+	Task opportunity = + because users must paste different types of information
CW-12	Frames	PDR#03: Fig. 22, p26 in Volume View Spec: User will be confused why this dialog box pops up, and labels in dialog box are not clear. Besides, this additional dialog box seems to be unnecessary (see back).	Dialog box for handling a large PICT is confusing		•			Design change: Full PICT is pasted in and can then be re-sized
CW-13	Frames	PDR#04: Icon for "Move Frame" button and "Create Frame" not meaningful, not in the "users" metaphor repertoire. (Fig. 13, p. 17, Volume View Spec) [we separated into two PDRs]	Move-frame icon will not be meaningful to users		• +		+	Design change: Now "MOVE" is on the button - but icon the one in the spec. Task = + because users must move frames.
CW-14	Frames	PDR#04: Icon for "Move Frame" button and "Create Frame" not meaningful, not in the "users" metaphor repertoire. (Fig. 13, p. 17, Volume View Spec) [we separated into two PDRs]	Create-frame icon will not be meaningful to users		• +		+	Design change: Now cross-hairs are on the icon - but icon still close to the one in the spec. Task = + because users must create frames.
CW-15	Frames	PDR#10: Icon for locking/unlocking frames (p. 16 design document) is not a meaningful metaphor. If icon ?? e.g. a key...I'd suggest to put this feature under Frame - Lock/Unlock.	Locking/unlocking icon is not visually meaningful		•			Design change: Simplified icon visually and button now says "LOCK" or "UNLOCK"

No.	Feature	PDR text	Summary Description	Implemented	Design changed	Not yet implemented	Task opportunity	Reasons for categories
CW-16	Frames	PDR#12: Selecting more than one frame, p.20 design document: Effect of the unlock/lock feature determined by upper left selected frames. Builder will not be creator of this detail and wonders what happens once he uses the feature.	Locking/unlocking more than one frame already in different states is confusing.			•		Not implemented: Selecting multiple frames.
CW-17	Frames	PDR#40: Xref tag appears at some location as lock / unlock frame tag and may not be visible.	X-ref and lock tags at same place		•			Design change: Locked frame is now indicated by a distinctive hashing of the frame-outline.
CW-18	Frames: Code	PDR#33: Specifying a slider (p. 25, fig. 20) User will be confused how to determine the value for increment.	Determining slider increment will be confusing			•		Not implemented: Runtime frames
CW-19	Frames: Code	PDR#34: Creating slider: No possibility to name a slider, to distinguish multiple sliders on one page.	Should be able to name sliders			•		Not implemented: Runtime frames
CW-20	Frames: Runtime	PDR#17: Restriction on number of call stacks/ drawing i/o frames on one page 21, last paragraph in design document doesn't make sense from a user's point of view. Due to the limited amount of space on each page there is a natural limit...	Why restrict no. of runtime frames on a page?			•		Not implemented: Runtime frames
CW-21	Frames: Runtime	PDR#32: "Drawing Frames": 1) User may forget to create one and wonder why the graphic output of a simulation doesn't appear 2) Transform to - Drawing Frame Action, 3) "Drawing" no label following possible [we split into 2]	User may forget to include a drawing frame and wonder why the graphic output of a simulation doesn't appear			•		Not implemented: Runtime frames
CW-22	Frames: Runtime	PDR#32: "Drawing Frames": 1) User may forget to create one and wonder why the graphic output of a simulation doesn't appear 2) Transform to - Drawing Frame Action, 3) "Drawing" no label following possible [we split into 2]	"Drawing" will not be associated with the graphic output of a simulation			•		Not implemented: Runtime frames
CW-23	Glossary	PDR#19: Glossary button "Gloss" doesn't support label-following by user.	"Gloss" will not be associated with glossary		•			Design Change: Button now says "GLOSSARY"

No.	Feature	PDR text	Summary Description	Implemented	Design changed	Not yet implemented	Task opportunity	Reasons for categories
CW-24	Glossary	PDR#27: Editing Glossary items: no explicit feedback from system that a change by just typing in the glossary pane will propagate to all affected links to that entry	The effects of editing a definition in the glossary pane are not clear		•			Design Change: Cannot directly edit a glossary definition and dialog box is clear about the effects of editing.
CW-25	Glossary	PDR#28: Resizing of Glossary pane: Why only in up-direction? User will expect from Mac experience to be able to resize it in any direction! Thicker margin not apparently visible - user will be confused. [we split into 2]	User will expect to be able to resize glossary pane both up and over		•			Design Change: Can resize in both directions
CW-26	Glossary	PDR#28: Resizing of Glossary pane: Why only in up-direction? User will expect from Mac experience to be able to resize it in any direction! Thicker margin not apparently visible - user will be confused. [we split into 2]	Re-sizing bar on glossary pane not recognizable as an action	•			?	Task = ? because tasks don't explicitly ask users to resize panes and entries may not be long enough to motivate them to do so on their own.
CW-27	Glossary	PDR#21: Text in dialog box p.32, Fig 31 design document confusing - the part in the parentheses.	Duplicate-entry dialog box has confusing phrase		•			Design Change: The confusing sentence in parentheses is not longer in the dialog box.
CW-28	Help	PDR#08: Confusion between balloon Help and Volume View Help, maybe less from the semantics (since user is supposed to be familiar with balloon help), but because of the similarity and spatial proximity of the icons.	Volume-Help icon looks too much like Balloon-Help icon and too close to it		•			Design Change: To new Mac-standard of having all helps in a menu at far left
CW-29	Pages	PDR#20: No user will find "Add Volume Page" under the "Edit" menu	"Add Volume Page" not findable in "Edit" menu, have a "Page menu"		• +		+	Design Change: Add-Volume-Page is now under the "Windows" menu. Task = + because users have to make new pages
CW-30	Pages	PDR#42: Remove pages: No warning after selecting Edit - Remove Volume Page	No warning after selecting Edit - Remove Volume Page, a destructive operation		•			Design Change: Now there is a confirmation-request dialog box

No.	Feature	PDR text	Summary Description	Implemented	Design changed	Not yet implemented	Task opportunity	Reasons for categories
CW-31	ToC	PDRs#11, 16, 18. 11:TOC symbol doesn't encourage label following. 16: Speak the user's language: TOC -> Table of Contents 18: User will have difficulties associating TOC with Table of Contents - no label following possible!	"TOC" will not be associated with "Table of Contents"		• +		+	Design Change: Button now has an outline icon, but still says ToC. Task = + because users have to make ToC
CW-32	ToC	PDR#15 &#35: Design document, p. 34 " Modifying Position to TOC entry in hierarchy". Promotion / Demotion of entries is forbidden, but no feedback to the user (in the first place, it's questionable why this is not permitted; [we split into two]	No indication to user that up/down moving of ToC entry is not permitted			•		Not implemented: ToC hierarchy
CW-33	ToC	PDR#15 &#35: Design document, p. 34 " Modifying Position to TOC entry in hierarchy". Promotion / Demotion of entries is forbidden, but no feedback to the user (in the first place, it's questionable why this is not permitted; [we split into two]	Why have the restriction that Up/down movement of ToC entries is not permitted?			•		Not implemented: ToC hierarchy
CW-34	ToC	PDR#35: Editing the TOC hierarchy: 1) no cue how to do that visible (-> method replies on user's head) 2) constraint that only lateral movements are allowed is artificial [we split into 3]	No visual cue that ToC entries can be dragged			•		Not implemented: ToC hierarchy
CW-35	Undo	PDR#41: Undo of sequences of actions: Not possible, only last mouse/key click	Can only undo one action, would like to undo a sequence of actions	•			?	Task = ? because our tasks may be too simple to want to undo a
CW-36	Volume	PDR#07: Design doc., p.6, - "Saving a Volume View with the user level set to explorer cannot be undone". Violates the guideline that every action should be undo-able.	Saving to explorer can't be undone	•			-	Task=- because don't ask them to save
CW-37	Volume	PDR#23: Navigating through pages confusing #23-No label following on p.22, Fig 16 in design document with respect to page up/down possible.	Page-up/page-down icons not meaningful		• +		+	Design Change: Now the icons have the word PAGE on them as well as the arrows, but still close to buttons in spec. Task = + because users need to navigate through their volume.

No.	Feature	PDR text	Summary Description	Implemented	Design changed	Not yet implemented	Task opportunity	Reasons for categories
CW-38	Volume	PDR#24: Navigating through pages confusing #24-Find menu all messed up - user wouldn't find the most efficient commands "Goto Page"	"Go to Page" item in Find menu too far down	•			?	Task =? because 3-page document maybe too small to need "Go to Page"
CW-39	Volume	PDR#25: Suggestion: "Windows" menu rename to "Show" (label following!) and rename subentry "Bring to Front" to "Show as Front Window" and put "Show Volume" as "Open Volume" under the "File" menu (draw on Mac Experience!) [we split into 3]	Mac users will expect "Open Volume" in File menu, not "Show-Volume" in Windows menu to access the volume	•			+	Task = + because users have to access a volume
CW-40	Volume	PDR#26 & #36: ...Necessity to choose Windows - Show Volume to get into builder mode with the current volume, contradicts Mac experience that you get into the correct application just by clicking on a file created by this application	User will not expect to have to explicitly open an existing volume		•			Design Change: Now if a volume exists and it was open when the file was closed, it is opened into the volume automatically
CW-41	X-ref	PDR#13 &#38: Changing destination of Cross References: Can only be done by deleting the X reference and recreating it with the right destination (p.30 design document)	Changing the destination of a cross-reference must be done by deleting the cross-reference and creating a new one			•		Not implemented: Cross references
CW-42	X-ref	PDR#14: Fig. 26 & Fig. 28 , p. 29 design document : Cross reference tag isn't meaningful to users - why should the user expect that he can trade the sources of a cross reference by clicking on this tag.	Cross-reference icon on a frame is not meaningful			•		Not implemented: Cross references
CW-43	X-ref	PDR#16: Speak the user's language: XRef -> Cross reference TOC -> Table of Contents [also see CW-11]	"X-REF" on button will not be associated with creating cross-references			•		Not implemented: Cross references
CW-44	X-ref	PDR#30: Necessity to click on Xref button a second time, after having selected the destination, will create problems.	Terminating click on X-REF button will be a problem			•		Not implemented: Cross references
CW-45	X-ref	PDR#37: Deleting Xref: Mac experience would lead to assume that Select - Edit - Delete performs this action but NEVER Select - XRef !	Users will not expect the X-REF button to be used as a toggle to delete a cross-reference as well as create it			•		Not implemented: Cross references
CW-46	X-ref	PDR#39: Consistency problem: Select -> XRef button common action sequence both to create and delete a XRef. => problems!	Using the same action sequence to delete a cross-reference as well as create it is a problem			•		Not implemented: Cross references

Appendix 2

The following table lists the 60 Usability Problems found by in the user tests.

It gives a brief description of the problem, which user's behavior exhibited that problem, and which analyst found the problem (encoded as <inital><PDR#> in the user's cell). It records that part of the specification which covered this aspect of the interface and whether the implementation was consistent with that specification (if something is not mentioned in the specification, it is judged to be consistent). In addition, if the problematic aspect was consistent with the spec, this table records whether it was explicit in the spec, implied by the spec, ambiguous or incomplete in the spec, assumed from Macintosh™ standards, or not mentioned in the spec at all, and the rationale for these judgments.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-01	Application	Save dialog comes up repeatedly until user names document	M1 B1	B2			p. 40: Set Autosave Parameters... is in File menu, no other mention of autosave.	•			•				Since there is a mention of autosave parameters, it is implied that some sort of autosave will happen
U-02	Application	More than a single object appears to be active at any time		M5		M4	p. 22 selecting frames section, selected text: p. 35 [glossary]"select the additional word...in Volume pane", p. 38 [ToC] "select the text of the entry in the Volume pane", selected glossary: p. 35: "select the text of the definition". many more	•			•				There are many separate parts of the spec that talk about different things being highlighted (frames, text, glossary entries, etc.), with no explicit discussion of their interaction, so it is implied that they are potentially active all at the same time.
U-03	Application	There is no text search facility		B24		B21	No mention in spec of "search"	•						•	Since the spec does not mention a text search capability at all, this capability is classified as not in the spec.
U-04	Application	Sporadic text hi-lighting problem		M6					•						Pure bug: text highlighting should be predictable by Mac standards of text-editing.
U-05	Application	Scrolling is very slow		B8			p. 1: "The goal of the redesign of the Volume View is to address the problems with the existing prototype. The primary objectives are to: ...Improve the performance...Display time."	•				•			Impossible to tell if the redesign improved the performance of ACSE, but it is still too slow for the user.
U-06	Application	Can't set default font size		B9			Spec says nothing about default fonts or sizes of font	•						•	Since the spec does not mention defaults, or changing defaults, this capability is classified as not in the spec.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-07	Application	There is no on-line help		B19			p. 21: "Clicking the Help button will display the on-line Genie Help system. The Help system interface will be discussed in a separate document."		•						Help was never specified or implemented.
U-08	Application	User expects Builder to work like a word processor in many ways (e.g., automatically add pages, arbitrary text in the ToC).				B5	Nothing about the philosophy is in the spec, i.e., whether the inspiration was MSWord, HyperCard, etc.	•						•	Since the philosophy was not discussed, this problem is classified as not being in the spec.
U-09	Application	Show-design item in Windows menu crashes the application		M13 B17					•						Pure bug: nothing should crash the application.
U-10	Bookmarks	Bookmark is placed on last visible page, not prominent page	M8 B10				p. 22 "To mark a particular page with a Bookmark, click the Bookmark button in the Tool and Button Palette. An arrow will appear to the left of the corresponding location on the scroll bar."		•						If it were implemented as spec'd then the bookmark would return to the exact spot *across* pages, if that were indeed where it was when marked.
U-11	Bookmarks	Remove-bookmark icon too similar to make-bookmark icon	M9 B11				p. 22 "To delete a Bookmark, select the Bookmark you wish to delete by clicking on it. Press the Delete key, or choose Clear from the Edit menu."		•						The only way to delete a bookmark now is to hit the Bookmark button again.
U-12	Bookmarks	User doesn't see bookmark appear after hitting bookmark button				B16	p. 22 "An arrow will appear to the left of the corresponding location on the scroll bar."	•		•					Feedback to hitting the bookmark button is explicit in spec, but the user didn't see it.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-13	Frames	User doesn't know to make a frame first before pasting in content (e.g., text, movies)		M2 B8	B2	B2	p. 26 "first select the text you want and copy it to the clipboard. Next, select the frame in which the text will appear. Choose the Paste command from the Edit menu." Almost same words on p. 29 about pasting PICTs and movies		•		•				The procedure in the spec is exactly what is giving the users trouble.
U-14	Frames	Can't select all frames on a page		M10 B16		M8 B18	pp. 22-23: section called "Selecting more than one frame."		•						The spec gives several ways to select multiple frames, none of which are implemented.
U-15	Frames	Frames won't drag between pages		M15 M17 B18		B8	p. 20: "Move Tool ... move a frame from one location in the Volume pane to another." p. 26 "Position the hand over the frame you wish to move, and click and drag the mouse. Release the mouse button when the frame is positioned in the desired		•			•			The spec doesn't say you can't drag frames between pages, but it doesn't say you can, either.
U-16	Frames	Can't delete a frame with the delete key		M18		B7	p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						Is the use of the delete key an operation common to Macintosh applications? If so, then not implemented as spec'd. Implementation doesn't allow Cut or Clear either; only allows "remove volume frame " in the Windows menu (which was NOT in the spec).
U-17	Frames	There is no support for page layout (like rulers, alignment, centering)	B6 B7	B5			No mention in the spec of terms like "layout" "center" or "align"	•						•	Since none of these terms are mentioned, this capability is classified as not in the spec.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Implˈtion consistent with spec	Implˈtion inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-18	Frames	Can't select whole frames as objects (e.g., to copy, paste)		M14			p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						Spec says you can select frames and work on them as objects, but not implemented that way.
U-19	Frames	Shift-click doesn't select multiple frames		M11			pp. 22-23: section called "Selecting more than one frame."		•						The spec gives several ways to select multiple frames, none of which are implemented.
U-20	Frames	User expected the Select-All menu item to select all frames in the document		M12			p. 23: "Located under the Edit menu are two commands controlling selection; Select All on Page, and Select All in Document."		•						These menu items are not implemented at all; they do not show up in the Edit menu.
U-21	Frames	Cutting, pasting, copying & deleting frames is inconsistent for different types of frames		B21			p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						These operations would be consistent if the system were implemented as spec'd.
U-22	Frames	User wanted a command key for removing frames		B22			p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."			•					Clear doesn't seem to have a standard keyboard shortcut, so making it Mac™ standard won't solve this problem, but if the delete key worked to remove the frame, that would have solved the spirit of this problem (but not the letter of the

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-23	Frames	Remove-Volume-Frame hard to find in Windows menu				B9	p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						Windows menu does not have a Remove-Volume-Frame item in the spec. In fact, there is no such item in the spec in any menu, as the spec allowed frames to be removed with "Clear" or "Cut".
U-24	Frames	User does not recognize the need to make and manipulate frames even when closely examining the frame tools			B3		pp. 18-19 two figures with the Frame-Tool icons in them		•	•					The tool-icons say "Select" under the pointer, "move" under the hand, and Frame" under the rectangle with the cross-hairs at lower-left; possibly better than spec'd. However, these are similar to what was spec'd so we will look for this as a problem.
U-25	Frames	User has trouble resizing frame; can't grab or hang on to frame			B6		P. 25: "Handles will appear around the frame to indicate that it is the current selection. These handles function in the same manner as object handles in MacDraw. Click and drag the handles in the desired direction to resize the frame."		•						Grabbing the whole side of a frame, as it is implemented, might be even easier than the handles might have been. But grabbing the corner might be harder. Can't tell. Probably harder to recognize available actions without handles.
U-26	Frames: Movie	Can't delete a movie frame with the delete key				M2	p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						Is the use of the delete key an operation common to Macintosh applications? If so, then not implemented as spec'd. Implementation doesn't allow Cut or Clear either; only allows "remove volume frame " in the Windows menu (which was NOT in the spec).

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-27	Frames: PICT	Can't remove a graphic frame with the delete key		M19			p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						Is the use of the delete key an operation common to Macintosh applications? If so, then not implemented as spec'd. Implementation doesn't allow Cut or Clear either; only allows "remove volume frame " in the Windows menu (which was NOT in the spec).
U-28	Frames: PICT	Can't remove a graphic frame with "cut"		M20			p. 22 "Choosing any of the operations common to all Macintosh software (Cut, Copy, Clear, etc.) or a specific Volume command will perform that operation on whatever is currently selected."		•						Implementation doesn't allow Cut (or Clear either), only allows "remove volume frame " in the Windows menu (which was NOT in the spec).
U-29	Frames: Text	Can't use bold, italics, etc.		M27 B26	B8		p 27: Once you've typed in the desired text, you may format it through the use of the Font and Style commands, located under the Format menu. (... "bold" and "underline" are grayed out....they are used to denote cross-references and Glossary entries.)		•						Spec allowed italics and other formats in the Style menu; implementation has no Style menu.
U-30	Frames: Text	Should share common function keys with other text editors		B4			Spec says nothing about function keys	•				•			Since the spec mentions nothing about function keys, this is classified as assumed from Macintosh™ standard.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-31	Frames: Text	Cannot change the font or size of text in a whole frame by selecting the frame and issuing font or size command				B14	p. 27 "Once you've typed in the desired text, you may format it through the use of the Font and Style commands, located under the Format menu."	•				•			Spec doesn't say what you have to select to get the font and style menu items to be active; could be specific text, could be a frame.
U-32	Frames: Text	User does not know that text formatting can be done inside the Genie			B4		p. 27 "Once you've typed in the desired text, you may format it through the use of the Font and Style commands, located under the Format menu."	•		•					It's explicit in the spec that you *can* format inside the Genie (although the procedure itself is ambiguous by not saying what you have to select: text or frame).
U-33	Frames: Text	User wants to make the blinking cursor "get out" of the frame			B5		Spec never talks about blinking text cursor	•					•		Leaving the cursor blinking at the last position in the text is a Mac™ standard. Still it confused the user.
U-34	Frames: Text	User does not recognize that "format" "font" and "size" refer to properties of text			B5		p. 43: Format Main Menu: "Font -> Hierarchical menu of fonts, Size -> Hierarchical menu of font sizes"	•		•					User just doesn't seem to know these words.
U-35	Frames: Text	Users have difficulty getting text frames to be the right size for their contents	B2	B3		B10	p. 25 " If a text frame is resized, the text inside the frame will reformat itself to the new size....the frame cannot be resized in such a way that the text inside would not fit in the frame."		•						Implemented so re-sizing width re-positions text, but keeps length same, so the frame can be too small and have scroll-bars. If this were implemented, procedure would be to make very long, make as narrow as you want, then make shorter until it stops you.
U-36	Glossary	Glossary is case-sensitive	M7 B9	M8 B13		M7 B17	p. 35: "If the word or phrase is currently a Glossary entry (case-sensitivity will not be an issue here)..."		•						This was an oversight by the implementers.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-37	Glossary	Glossary doesn't find other occurrences of word	M3 B3	M7 M9 B14			p. 35 "To link an additional instance of a word or phrase to a Glossary entry, select the additional word or phrase (for example, another instance of the word "drosophila") in the Volume pane and click the Glossary button."	•		•					The procedure in the spec is exactly what the users object to.
U-38	Glossary	Removing a page with a glossary entry does not remove the glossary entry from the glossary pane	M12 B14			M12	p. 41: "Remove Volume Page...Delete the volume page which contains the currently selected frame. " P. 26: 'Delete Volume Page' will delete all pages with selected frames on them. "	•				•			This particular effect of remove-volume-page is not mentioned in the spec, so it is classified as being incomplete in the spec.
U-39	Glossary	Can't delete a glossary entry with the delete key	M5	M18			p. 36: "select the word 'drosophila' in the text of the Glossary, and press the Delete key or choose Clear from the Edit menu. The Glossary entry and accompanying definition will be deleted from the Glossary. "		•						The procedure in the spec is what the users wanted.
U-40	Glossary	Can't edit a word in a text frame that is already in the glossary	M4 B4				Nothing on glossary entries, but ToC: p. 38: "select the text of the entry in the Volume pane...Replace the selected text with the desired header by typing at the keyboard." Nothing about interaction with glossary entries that might be in ToC	•				•			The spec says nothing about the interaction of changing text and glossary entries; it talks about each one separately but is incomplete when it comes to the interaction.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-41	Glossary	Glossary function doesn't strip off white space		M25 B25			p. 34: "To add an entry to the Glossary, select it and click the Glossary button in the Tool and Button Palette."	•			•				The method is implied in the spec, in that is says "select it [the word or phrase]", not "select it and any white-space before or after."
U-42	Glossary	Can't directly edit glossary entry				M11 B15	p. 35: "move the mouse pointer to the Glossary pane, and select the text of the definition (in this case, "common fruit fly"). Replace the text with the desired definition by typing at the keyboard."		•						The procedure in the spec is what the user wanted.
U-43	Glossary	Can't drag & drop entries into glossary pane		B23			p. 34: "To add an entry to the Glossary, select it and click the Glossary button in the Tool and Button Palette."	•		•					User just wants this other method.
U-44	Pages	Remove-page removes the page with a visible, selected frame, no matter how slightly it is visible.	M11 B12		M4	B19	p. 41: "Remove Volume Page...Delete the volume page which contains the currently selected frame. " P. 26: 'Delete Volume Page' will delete all pages with selected frames on them. "	•		•					The name of the menu item is actually spec'd in two ways, "Remove Volume Page" and "Delete Volume Page" but it is implemented as "Remove Volume Page." Although this is a conflict in the spec, it is irrelevant to the users' problem.
U-45	Pages	User expects Add-Volume-Page to add the page at the end of the volume, not after the page with a selected frame			M3	M9	p. 26: "Add Volume Page" will add a page after the last page with a selected frame on it. p. 41: Edit menu, Add Volume Page item, "Add a new Volume Page after the page with the currently selected frame"	•		•					The effect in the spec is exactly what confused the user.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-46	Pages	Add-page should scroll to the page just added	B5	M26 B7			p. 26: "Add Volume Page" will add a page after the last page with a selected frame on it. p. 41: Edit menu, Add Volume Page item, "Add a new Volume Page after the page with the currently selected frame"		•			•			The spec tells you how to add a page, but doesn't say where the pane will be left after add-volume-page, so it is classified as incomplete.
U-47	Pages	Can't adjust page-breaks		M16			p. 17 "Please note that the user may not change their page size after it has been set."	•		•					User wants to do this for a naive reason anyway (to fit more on the page, not adjust to a different Mac screen).
U-48	Pages	Can't re-order pages except through brute force reconstruction			M2		p. 37 "To move a section of the Volume (and its accompanying header) up or down in the overall hierarchy of the Volume, you will need to manually position the frames in the desired area of the Volume"	•		•					Manipulating the ToC could be like using an outliner, which would allow simple re-ordering of pages. But it was explicitly not specified that way.
U-49	Pages	User does not understand the concept of a "page" in the volume.				B3	This is the first thing a Builder does, but rationale not discussed. p. 16 "After the New command is selected, you will be prompted to enter the page size of your Volume."	•			•				The concept of pages is not discussed in the spec, only implied. The procedure for setting page size is not implemented as spec'd, but this is irrelevant since pages *are* there, set to a particular screen size, and the user doesn't understand them.
U-50	Pages	User wants a command key for making a new page		B6			p. 41: in Edit Main Menu: "Add Volume Page" with no shortcut after it.	•		•					User just wants this shortcut.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Impl'tion consistent with spec	Impl'tion Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-51	Pages	Several operations act on the last visible page rather than the dominant page (bookmark, remove-page, add-page).				B20 B13	See other entries above. They will act on the pages with selected frames and frames stay selected after page is not prominent	•			•				The philosophy of where page-commands will act is not explicit in the spec, only implied by looking across several commands.
U-52	Pages	Can't find the Add-Volume-Page menu item in Windows menu				M3 B6	p. 41: in Edit Main Menu: "Add Volume Page"		•	•					Although this is not implemented to spec, we will look for it to be predicted anyway because it is not at all clear that putting "Add Volume Page" in the Windows menu is an improvement over the Edit menu.
U-53	ToC	Can't edit the ToC entry directly		M6		M5 B12	p. 38: "select the text of the entry in the Volume pane...Replace the selected text with the desired header by typing at the keyboard. The Table of Contents will automatically be updated to reflect the change. "	•		•					Procedure given in spec is exactly the thing giving the users the problem
U-54	ToC	Only one ToC entry per page		B12		B11	Spec does not say anything about a restriction to one ToC entry per page		•			•			This was classified as consistent with spec because nothing was said about a restriction. Another possible interpretation is that "everyone assumes" that if no restriction is mentioned there is none, so this should be classified as inconsistent with spec.
U-55	ToC	Can't delete ToC entry with the delete key		B11		B12	p. 38 "To delete a Table of Contents entry, select the desired entry in the Table of Contents pane. Press the Delete key, or choose Clear from the Edit menu."		•						The users wanted to do the procedure in the spec.

No.	Feature	Description of observed problem	User1	User2	User3	User4	Reference in Specification	Implition consistent with spec	Implition Inconsistent with spec	Explicitly in spec	Implied in spec	Ambiguous or incomplete in spec	Assumed from Macintosh™ interface	Not mentioned in spec at all	Explanation and/or elaboration
U-56	ToC	Selecting a ToC frame in the text pane does not always update the ToC pane's highlighting	M13				nothing in spec to say how selecting things in volume pane effects ToC pane. E.g., p 22: "selected frame will appear surrounded by handles, to indicate that it is the current selection." nothing about ToC pane.	•				•			This is classified as "incomplete" because the navigation and selection of frames sections talk about what happens in the Volume pane, but not what happens in the ToC pane when you do these things.
U-57	ToC	User can't figure out how to "make" the ToC		M3 B10			p. 36: "To enter this phrase in the Table of Contents, select it and click the Table of Contents button in the Tool and Button Palette."	•			•				This is classified as "implied" because the spec doesn't explicitly say "You don't have to do anything off-line to 'make' the ToC" it just tells you how to enter ToC headers.
U-58	ToC	User expects ToC to allow arbitrary text, not only all of a frame				B4	p. 36: "Table of Contents entries can consist of a word or short phrase, and must be the only item in a text frame."	•		•					The restriction in the spec is exactly what's giving the user the trouble.
U-59	ToC	The user does not see the ToC button as a toggle				B13	p. 38: "To delete a Table of Contents entry, select the desired entry in the Table of Contents pane. Press the Delete key, or choose Clear from the Edit menu."		•						In the implementation, the ToC button is a toggle -- can't use Clear or delete to get rid of a ToC entry.
U-60	Volume	Users don't know to use "show volume" in "Windows" menu to open a new volume		M1 B1	M1 B1	M1 B1	p. 44: Show Volume is in Window menu	•		•					The menu arrangement in the spec is exactly what's giving the users problems.