7 Evaluating the Interface

The previous chapters discussed the design and implementation of the CodaConsole interface as well as the techniques employed by it. This chapter presents an evaluation of the usability of that interface using a technique developed by the human-computer interaction (HCI) community.

For the benefit of those readers not familiar with HCI evaluation techniques, I begin this chapter by briefly summarizing the range of usability evaluation techniques available, focusing on the technique used in my evaluation. For further information on these and similar techniques, Nielsen [37] provides an excellent overview.

I then present the usability test, detailing the goals, method, results, analysis, and findings of the study. The test considers two perspectives, that of expert Coda users and that of novice Coda users. Briefly, the goal of the test was to determine the ease with which users were able to master the interface, their ability to understand the feedback provided by the interface, and their ability to operate the interface.

After presenting the results of the usability test, I then reflect on the usability test itself, focusing on the lessons I learned about the process. I conclude the chapter with a summary of the main results.

7.1 HCI Evaluation Techniques

The HCI community has developed numerous evaluation methods. These range from simple (and quick) techniques like heuristic evaluation [36] and cognitive walkthroughs [57] to ones that are more involved like usability testing [10] and field trials. *Heuristic evaluation* aims to find usability problems by having a small set of evaluators judge the interface's compliance with a set of accepted usability principles. *Cognitive walkthroughs* judge the ease of learning, particularly exploratory learning, of an interface. Both of these methods are *discount* usability engineering methods[23] because they can be performed in little time and at low cost [37]. *Usability testing* is an evaluation method whose goal is to discover usability problems by observing users interacting with the software. It requires a working interface (though that interface can be a mock-up) as well as more time and money, but offers more realistic and detailed data. *Field trials* evaluate software in "the real world". By that, I mean the users are the actual people who use the software every

[23] Nielsen categorizes heuristic evaluation as a discount usability method. Though he does not classify cognitive walkthroughs, this method also fits his definition.

day and the tasks used in the evaluation are the actual tasks those people do during the period of observation. Field trials require significant amounts of time and money as well as a production-quality system, but offer the most realistic data.

Usability testing is, in some sense, a compromise between the discount usability methods and field trials. It offers significant advantages over discount methods, but at significantly lower cost than a field trial. Furthermore, it can be performed relatively early in the life cycle of the software. For these reasons, I chose this method for my usability evaluation. In the rest of this section, I discuss the purpose of usability tests and describe two important issues to consider while designing such a test.

7.1.1 Purpose

The purpose of a usability test is to discover *the most serious usability problems* in a piece of software by observing users interacting with it. The goal is neither to prove that the software is usable nor to obtain statistically significant results. The goal is to discover usability problems so that they can be addressed and corrected. Thus, qualitative data, such as comments made by users as they work with an interface, are often as enlightening as quantitative data, such as the time it took them to complete a task. Data sources, then, include timings, responses to questions, counts of the number of actions taken and even comments made about the interface.

7.1.2 Selecting Participants

In order for the results of a usability test to be meaningful, the participants need to be representative of the people who are expected to use the software. Figure 7.1, often used by Randy Pausch [40], illustrates why this is important. The pie chart on the left shows that the percentage of all computer users who are computer scientists is extremely small[24]. The fundamental problem is that the intuitions of computer scientists do not necessarily apply to the other 99.99% of computer users; in fact, those intuitions hinder their ability to make systems usable to non–computer scientists. If we were to assume other computer scientists were representative of all users, the likelihood of serious usability problems would be high, even though we had performed usability testing.

By extension, this same argument also applies to developers of a system. Because developers know more about the interface's domain than computer scientists in other areas, their intuitions do not necessarily apply to other computer scientists. They simply have too much domain knowledge. Thus, recruiting participants who are representative of the expected user population is a crucial part of usability testing.

---

[24] Not drawn to scale because if it were, "our" slice of the pie would not be visible.

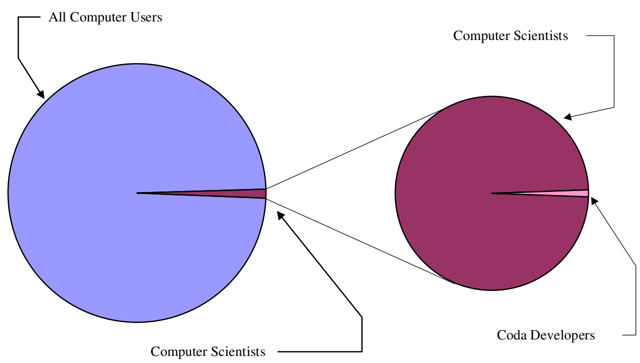


Figure 7.1: Computer Scientists are Special People
This figure illustrates that computer scientists represent a very small percentage of all computer users. It further illustrates that Coda developers represent a very small percentage of computer scientists. Coda developers have background knowledge not available to other computer scientists. Likewise, computer scientists have background knowledge that is not available to other computer users.

7.1.3 Selecting Tasks

A similar argument shows the importance of presenting representative tasks during a usability test. The critical observation here is that usability testing can only discover problems in the areas of the interface under study, though problems may generalize to other areas of the interface that were not studied. If the tasks presented to participants are not representative of actual usage patterns, serious usability problems may go undetected until the interface is released for general use.

7.2 Usability Test: Goals

The first objective of my usability test was to identify areas of the interface and tutorial that were particularly problematic. My goal was to create a list of usability problems, categorized by their severity [10]. The method with which I decided to identify these problems was through observations of people using the interface, and analysis of the qualitative and quantitative data collected during that observation.

The second objective of my test was to gauge whether or not the interface was meeting the learning and performance goals I had set. In particular, my usability test focused on three major issues: the ease of mastering the interface, users' ability to understand the feedback provided, and their ability to operate the interface.

Ease of mastering the interface:

- Can users learn to use the interface in approximately one hour?

Ability to understand feedback:

- Can users understand problems indicated by the interface?
- Can they determine what actions might resolve those problems, if any?

Ability to operate:

- Are users able to define and hoard tasks?
- Are users able to provide advice to the system?
- Are users able to configure individual events?

In answering these questions, my goal was to obtain an indication of how well people with backgrounds similar to those of Coda users would be able to perform activities similar to those required of Coda users. Providing conclusive answers to these questions would obviously require a study of actual Coda users interacting with the production-quality interface and is beyond the scope of this thesis.

7.3 Usability Test: Method

To answer these questions, I performed a usability test[25] in which I observed a total of 14 people interacting with the interface. The first five participants were pilot users. The interface, materials, and/or procedures used for these users differ somewhat from those described in this chapter and contained in Appendix B. The remaining nine participants all used the same interface, materials and procedures. In this section, I discuss the demographics of the participants, the procedure used during the test, and the measurements I made.

7.3.1 Participants

Because my goal was not to release Coda and this interface to the general public, but rather the more modest goal of making Coda more easily accessible to other computer scientists, my user population was other computer scientists. The users for my test were recruited from among computer science graduate students at Carnegie Mellon. Because, at the time, Coda ran only on Unix-based operating systems, all participants were required to have substantial Unix experience and, in fact, all used Unix systems on a daily basis. In addition, I screened users for their experience with the Andrew File System. All participants were required to have substantial experience with AFS, using it on a daily basis. All novices and all but one expert stored either their e-mail directory or their home directory in AFS. Because AFS is well entrenched in the CMU community, this was an

[25] This test was reviewed and approved by the Carnegie Mellon University Provost's office in accordance with the requirements of Public Law 99-158 as implemented by Part 46 of Title 45 of the Code of Federal Regulations and General Assurance No. M-1462-XM.

easy requirement to meet. I did not screen users based upon their HCI experience.

I initially defined a *novice* user as a person who knew nothing about Coda prior to participating in the test. However, due to the difficulty of finding qualified users that met this stringent requirement[26], I was forced to relax it somewhat. The novice users knew very little about Coda. Most of them knew that it was a file system, and that it was related in some way to AFS. None of them had any direct experience with Coda. I defined an *expert* user as a person who had substantial experience using Coda. All of the expert users had Coda laptops and had operated disconnected for substantial periods of time over a period of at least a year.

Although I posted a request for participants for the test on both an electronic bulletin board and on a department-wide, electronic chat system, most participants heard about the test through word of mouth. I recruited a total of 14 users to participate in the test, offering them Coda shirts for doing so. All but two of those users were computer science graduate students. Of the two who were not graduate students, one (C1) was a faculty member and one (S1) was a nontechnical member of the support staff with some undergraduate-level computer science background. Neither of these users could be considered either experts or novices. One pilot user (P5) and the three expert users (E1, E2, and E3) had used Coda. No other user had done so. Three pilot novices (P2, P3, and P4) had some HCI experience. Two novices (N1 and N2) and the faculty member (C1) had extensive HCI experience. No other users had HCI experience. One user (T1) was originally considered a novice, but his data was eventually thrown out because it became clear, during the Tutorial, that he knew more about Coda than I had originally thought.[27]

7.3.2 Treatment

I began each session of the experiment by asking the participant to sign a consent form. Next, the participant filled out a brief background questionnaire, providing some basic demographic information. Once that was completed, the experimenter demonstrated a verbal protocol[28] using a computer game (Klondike). The participant then played Othello briefly to practice thinking aloud. During this practice verbal protocol, the experimenter positively reinforced those comments (both positive and negative) made by the user that were specific in nature, and provided concrete examples of positive and/or negative feedback if none were forthcoming. After a few minutes of practice, the participant began a detailed, interactive tutorial of the CodaConsole interface. The content of the Tutorial is shown in Screens 1-62 of Appendix B. This Tutorial was expected to require approximately one hour to complete. After a break, the participants performed a number

[26] At least one Coda paper is now covered in a class required of all computer science graduate students.

[27] The specific comments that led me to this conclusion are documented in Appendix C.

[28] A *verbal protocol* is a technique used to understand what users are thinking and where users run into problems as they work with an interface. Users are simply asked to think aloud as they work. Although this is awkward for users at first, it has been shown to be extremely effective once users become accustomed to talking as they work [11]. One participant commented to me that he continued talking aloud for awhile after completing the test.

of exercises ranging from identifying problems to preparing for a disconnected session. These Exercises are contained in Screens 63-83 of Appendix B. The Exercises were grouped into three phases. During Phase One (Screens 65-74), the participant answered basic questions about the state of the system. During Phase Two (Screens 75-80), he defined and prioritized three tasks. During Phase Three (Screens 81-83), he reacted to problems indicated by the interface. The Exercises were also expected to require approximately one hour to complete. After completing the Exercises, the participant answered a brief survey evaluating the interface. Finally, the experimenter debriefed the participant, allowing the participant the opportunity to ask questions. The materials used in this test can be found in Appendix B. Both the Tutorial and Exercise segments of the test were videotaped, and the audio from all three segments was also recorded.

7.3.3 Measurements

The first measurement I collected was the time (in seconds) required to teach users about the details of the indicator light interface. I measured these times for each screen of the on-line Tutorial. In addition, I measured some higher-level times such as the time the user spent learning about each indicator as well as the time the user spent on the entire Tutorial. Note that this is not quite the same as the sum of the times of the specific screens because those times do not account for the time necessary for the Tutorial to trigger events or display the individual screens.

The second measurement I collected was the time (in seconds) required for the user to complete each exercise. I measured the time for each complete exercise, not at the granularity of individual questions within an exercise. As with the Tutorial times above, I also measured some higher-level times such as the time needed for each of the three phases as well as the time spent on the entire Exercise session. Again, because of the time needed to trigger events or display individual screens, the summary times may not equal the sum of the individual times.

The third measurement captured the user's ability to use the interface to answer questions or perform some action. In addition, it measured the user's ability to describe a problem that had been indicated through the interface, as well as the steps necessary to resolve it, if appropriate. For this measure, I collected the user's response to each exercise. A response could be a written answer to a question or a change to one of the configuration files. I then scored the user's responses according to the key contained in Appendix B.

The final measurement captured the user's efficiency in answering the questions or performing the required actions. For this measure, I counted the number of steps the user took to complete each exercise. A step corresponds to double-clicking an indicator light, clicking a button, or typing a pathname, as well as similar activities. The baseline measurement was that of the author performing the same actions. The specific actions performed during the baseline measurement are contained as the Par Score of Appendix B. It is important to note that each par score is likely to represent a lower bound on the number of steps necessary to answer the question because the author was not only an expert using the interface, but also had foreknowledge of the expected answers to the exercises. Although an expert without foreknowledge of the exercises would have provided a better standard of comparison, such an expert did not exist.

All but the first participant were videotaped during their sessions. The first participant, P1, informally piloted the Tutorial and Exercises. No video or audio recordings were made because the experiment did not occur in the User Studies Lab. From the transcripts of the recorded sessions, I identified comments made by each user. These comments are recorded in Appendix E in two forms: a numerical listing by comment identifier number and a table summarizing which users made each comment during the three segments (Tutorial, Exercises, Debriefing) of the test.

7.4 Usability Test: Results

Overall, the results of the usability test were highly positive. Novice Coda users were able to use the system almost as well as expert Coda users. They learned about Coda in approximately one hour and they were able to successfully complete exercises typical of those required of users of the new interface. In this section, I revisit the original objectives of the test, examining whether or not users were able to master, understand, and operate the interface.

7.4.1 Learnability

The first set of questions relates to the ease of learning how to use the interface. The first objective of the test was to determine whether or not users could learn to use Coda and the interface in approximately one hour. Indeed, they could. The expert and novice participants in the test each took approximately one hour. The fastest user, N4, took just under 42 minutes; the slowest, N2, took just under 63 minutes.

Another goal of the usability test was to identify areas of the Tutorial that users found difficult. All but two screens that users found difficult were directly related to one or more usability problems in the interface. Improving the Tutorial may have helped users, but would not have fundamentally solved the problem. The other two screens (those numbered 23 and 28) were probably falsely identified as difficult by limitations in my analysis. The analysis presented in Section 7.5.1 below examines this issue in detail.

In order to confirm this result, I examined the qualitative data collected as part of the questionnaire administered after completion of the Exercises (see Appendix B). From these data, I learned that users felt the Tutorial was basically complete. On a scale of one to five (where 5 is “very complete”)¸ the Tutorial received a score of 3.8. Four of the six users commented that reintegration was not covered, though it was used during the Exercises. This omission was a deliberate attempt to see how users would react to “learning on the fly”. Based upon their reactions, it appears that they did not appreciate the experiment. Users thought the Tutorial was easy to understand (4.2 out of 5). Users unanimously felt that the Tutorial allowed them to grasp the scope of the interface.

Finally, five of six users felt it was presented at an appropriate level of detail. The sixth user, an expert, felt there was too much detail, specifically too much Coda background. Because the Tutorial was written for a novice audience, this is not surprising.

Based upon the quantitative and qualitative data I collected, the Tutorial achieved its objects of teaching users about Coda and the interface in about an hour. Further, no serious difficulties directly attributable to the Tutorial were uncovered.

7.4.2 Understandability

The second set of questions relates to the user's ability to understand the information presented by the interface. Eighteen of the twenty four exercises address this issue. These exercises include most of Phase One (those asking users to describe the current network connectivity, to identify which tasks are currently hoarded, etc.) and all of the exercises in Phase Three (which asked users to explain events notified via the indicator lights). Isolating the users' scores on these exercises from Figure 7.2, I find that users did extremely well. All but one user scored 97% correct or better, and that user scored 90% correct. Furthermore, the data show that, with the exception of just one of these exercises (SpaceStatus), no two users answered the same exercise incorrectly. These users convincingly demonstrated their understanding of the information presented by the interface and of what, if any, actions might resolve those problems.

Further, these users silenced any concerns over whether or not users could understand the feedback provided during event notifications. Phase Three of the Exercises triggered 12 different events. For each event, users were asked to fill out an "incident report", consisting of the urgency of the event, a description of the problem, and a description of any actions that might resolve the problem. The six expert and novice users filled out 71 incident reports describing these events—one user failed to report an event that transitioned from a "bad" state to a "good" state. Of these 71 incident reports, only three contained incorrect information. Users clearly understood the error conditions and what, if any, steps were possible to resolve those problems.

7.4.3 Operability

The third set of questions relates to the user's ability to operate the interface. Eight exercises address this issue. These exercises include three from Phase One (which involved configuring events and providing advice, and which were excluded from the discussion above) and all of Phase Two (which involved defining and prioritizing tasks). Although users scored well on these exercises, their scores declined somewhat (see Figure 7.2). Five of six users scored 90% or better, and the sixth user scored 84%. Furthermore, the data show that, with the exception of two of these exercises (ProvideHoardWalkAdvice and DefineBugsTasks), no two users answered the same exercise incorrectly. Although these scores are very encouraging, users clearly had more difficulty operating the interface than they did understanding the feedback it provided.

Problem Identifier	Pts	Novice Users 1	Novice Users 2	Novice Users 4	Novice Users Avg	Expert Users 1	Expert Users 2	Expert Users 3	Expert Users Avg	Overall Avg	Overall StdDev
Describe current network conditions	3	3	3	3	3.0	3	3	3	3.0	3.0	0
Describe “Tasks unavailable” event & configuration	6	5	6	6	5.7	6	6	6	6.0	5.8	0.4
Fix configuration for this event	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Determine token expiration time	1	1	1	1	1.0	1	1	1	1.0	1.0	0
Describe hoarding details	12	12	12	12	12.0	12	12	12	12.0	12.0	0
Provide hoard walk advice	2	2	1	0*	1.0	1	1	1	1.0	1.0	0.6
Describe current space status	3	2	3	3	2.7	1	2	3	2.0	2.3	0.8
Describe “Weakly Connected Cache Miss” event & “Disconnected Cache Miss” event	8	6	8	8	7.3	8	8	8	8.0	7.7	0.8
Fix configuration for above events	4	4	4	4	4.0	4	4	4	4.0	4.0	0
Define “Fixing Bugs” task	9	7	8	7	7.3	9	8	7	8.0	7.7	0.8
Define “Header Files” task	5	5	5	5	5.0	5	5	5	5.0	5.0	0
Define “CV” task	6	5	6	6	5.7	6	6	6	6.0	5.8	0.4
Prioritize newly defined tasks	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Request immediate hoard walk	1	1	1	0*	0.7	1	1	1	1.0	0.8	0.4
Describe “Filling RVM” problem	3	3	3	3	3.0	3	3	3	3.0	3.0	0
Describe “Fixed RVM” problem	2	Ø	2	2	1.3	2	2	2	2.0	1.7	0.8
Describe “Task Unavailable” problem	3	3	3	3	3.0	3	3	3	3.0	3.0	0
Describe “Hoard Walk Begin” event	2	2	2	1	1.7	2	2	2	2.0	1.8	0.4
Describe “Hoard Walk End/Task Available” event	2	2	2	2	2.0	2	2	1	1.7	1.8	0.4
Describe “Operating Weakly Connected” problem	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe “Weak Miss” event	2	2	1	2	1.7	2	2	2	2.0	1.8	0.4
Describe “Operating Disconnected” problem	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe “Disconnected Cache Miss” event	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe “Operating Strongly Connected” problem	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe “Tokens Expiry Event”	3	3	3	3	3.0	3	3	3	3.0	3.0	0
Describe “Reintegration Pending Tokens” event	3	3	3	3	3.0	3	3	3	3.0	3.0	0
Total Points	92	83	89	86	86.0	89	89	88	88.7	87.3	2.4
Percentage Correct	100	90	97	93	93.3	97	97	96	96.7	95.0	2.9

Figure 7.2: Correctness Ratings

This table shows the score each user received on each of the exercises as well as their total and percent correct. The empty set notation (Ø) indicates the user did not perform this exercise. User N4 explained precisely which files she would have requested to have fetched during the “Provide hoard walk advice” exercises; unfortunately, she failed to actually select those objects to be fetched in the interface. Had she completed the exercise as she described, her score would have been a 1. Because of her failure to complete this exercise, she was also unable to complete the “Request immediate hoard walk” exercise—though she did the most reasonable thing given the situation.

A different metric, which also gauges the user's ability to operate the interface, compares the number of actions users need to perform each exercise with the number needed by an expert. Figure 7.3 shows this data. In general, users need approximately twice as many actions as a user who is experienced with the interface and who has foreknowledge of the expected answers. Users clearly found the hoard walk advice and task definition aspects of the interface more challenging than other aspects of the interface. This trend is not surprising because these aspects of the interface require more problem solving. One interesting observation is that expert users required 30 steps more than novice users, on average. Largely, this appears to have been the result of two anomalous data points (E1's performance on defining the "Fixing Bugs" task and E3's performance on defining "Header Files" task). This observation should not be taken as an indication that expert users experienced more difficulty than novice users.

The test set out to answer three specific questions related to operability. I wanted to learn whether or not users could define and hoard tasks, provide advice to the system, and configure individual events. Users were largely able to define and hoard three realistic tasks (see Figure 7.2). The most disturbing trend I observed was the high number of users who failed to hoard the `compress`-related programs. Users were able to provide advice to the system, both with respect to weakly connected demand fetches and with respect to hoard walks. Users were also able to configure events, largely without difficulty. The problems users experienced are discussed in detail in the next section.

7.5 Usability Test: Analysis

As the previous section documents, the participants successfully met all of the objectives set forth for the test. As with any user interface, however, there is always room for improvement. One of the important goals of the test was to identify areas of the interface on which to concentrate our future efforts. Because users performed so well, I had to examine the data more closely to identify these areas.

This section presents this analysis in detail. The analysis is based upon the data collected during the course of the test. The raw data is contained in Appendices C and D. The details of the analysis are contained in Appendix E. The results of this analysis, the findings, are contained in Appendix F and summarized in Figure 7.4.

These findings are categorized by their *scope* and *severity* [10]. The scope indicates how widespread a problem is. A problem with *local* scope is limited to a single window whereas a problem with *global* scope applies to multiple windows. A problem with global scope is generally considered more critical than one with local scope. The severity of a problem indicates how critical the problem is. Dumas and Redish define four different severity levels. A problem with severity level 1 prevents the completion of a task. One with severity level 2 causes users significant delay or frustration. One with severity level 3 presents only a minor effect to usability. One with severity level 4 represents a more subtle problem and may point to potential enhancements to the interface.

Problem Identifier	Par	Novice Users 1	Novice Users 2	Novice Users 4	Novice Users Avg	Expert Users 1	Expert Users 2	Expert Users 3	Expert Users Avg	Overall Avg	Overall StdDev
Describe current network conditions	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe "Task Unavailable" event & configuration	3	5	3	6	4.7	6	3	3	4.0	4.3	1.5
Fix configuration for this event	4	3	4	3	3.3	4	4	4	4.0	3.7	0.5
Determine token expiration time	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe hoarding details	4	6	2*	4	4.0	6	4	3	4.3	4.2	1.6
Provide hoard walk advice	10	35*	46	31	37.3	20	36	26	27.3	32.3	9.0
Describe current space status	2	2	2	2	2.0	2	2	2	2.0	2.0	0
Describe "Weakly Connected Cache Miss" event & "Disconnected Cache Miss" event	3	7	5	3	5.0	3	3	15	7.0	6.0	4.7
Fix configuration for above events	8	12*	9	8	9.7	9	17	8	11.3	10.5	3.5
Define "Fixing Bugs" task	36	62	47	40	49.7	96	44	58	66.0	57.8	20.5
Define "Header Files" task	16	26	26	25	25.7	47	37	100	61.3	43.5	29.0
Define "CV" task	21	73	46	47	55.3	57	31	37	41.7	48.5	15.0
Prioritize newly defined tasks	14	5*	33*	38	25.3	27	24	37	29.3	27.3	12.2
Request immediate hoard walk	3	3	3	7	4.3	3	5	7	5.0	4.7	2.0
Describe "Filling RVM" problem	2	3	6	6	5.0	4	6	6	5.3	5.2	1.3
Describe "Fixed RVM" problem	0	Ø	0	0	0	0	2	2	1.3	0.8	1.1
Describe "Task Unavailable" problem	3	7	10	5	7.3	7	2	3	4.0	5.7	2.9
Describe "Hoard Walk Begin" event	0	1	0	0	0.3	0	0	1	0.3	0.3	0.5
Describe "Hoard Walk End/Task Available" event	0	3	5	8	5.3	7	4	4	5.0	5.2	1.9
Describe "Operating Weakly Connected" problem	2	2	4	2	2.7	3	1	6	3.3	3.0	1.8
Describe "Weak Miss" event	1	3	3	2	2.7	3	1	2	2.0	2.3	0.8
Describe "Operating Disconnected" problem	1	5	4	3	4.0	0*	0	7	2.3	3.2	2.8
Describe "Disconnected Cache Miss" event	2	2	2	2	2.0	2	4	2	2.7	2.3	0.8
Describe "Operating Strongly Connected" problem	0	2	0	2	1.3	2	1	0	1.0	1.2	1.0
Describe "Tokens Expiry Event"	0	1	3	2	2.0	2	3	2	2.3	2.2	0.8
Describe "Reintegration Pending Tokens" event	4	1	2	10	4.3	4	2	4	3.3	3.8	3.3
Total	143	273	269	260	267	318	240	343	300	284	39
Efficiency (Score/Par)	1.0	1.9	1.9	1.8	1.9	2.2	1.7	2.4	2.1	2.0	0.3

Figure 7.3: Efficiency Ratings

This table shows the number of steps each user took to perform each of these required tasks. The empty set notation (Ø) indicates that the user did not perform the task. Numbers which are underlined represent exercises for which this user was more efficient than the interface expert. Numbers marked with an asterisk are explained in the paragraph below.

User N1 requested significantly less data be fetched during the "Provide Hoard Walk Advice" exercise though he required more steps to provide that advice. User N1 hit certain buttons multiple times in a row (as many as eight). Consecutive events such as this are counted as two actions for the purposes of measuring the user's efficiency. Primarily this affects the two "event configuration" exercises and the three "task definition" exercises. User N1 prioritized tasks as he defined them so his efficiency on the "Prioritize newly defined tasks" exercise is artificially low. User N2 has an artificially high rating on this same task because he went back to reorganize the "Define 'Header Files' Task" while performing this task. User N2 has an artificially low efficiency score for the "Describe hoarding details" exercises because he resized the window and avoided scrolling. User E1 already had the "Network Information" window visible when he performed the "Describe 'Operating Disconnected' problem" exercise.

ID	Description	Scope	Severity	Page
1	Users confused by event notification	Global	1	414
2	Some help windows aren't helpful	Local	1	414
3	Users fail to include needed programs in definition	Local	1	415
4	Event Configuration Tab of Control Panel layout inappropriate to task	Local	1	415
5	Certain characters, if used in task names, can crash the interface	Local	1	415
6	Password visible on the Authentication Information window	Local	1	416
7	Users do not recognize the hoard walk advice request as a request for advice	Local	1	416
8	User does not recognize that the Hoard Walk Advice offers more details	Local	1	416
9	Indicator changes to Normal urgency level before appropriate to do so	Local	1	417
10	Users unsure whether hoard walk is proceeding during request for advice	Local	1	417
11	User prioritized tasks inappropriately	Local	1	417
12	Users want undo facility	Global	2	418
13	Users find the behavior of "Save" and "Commit" confusing	Global	2	418
14	Users are frustrated by the placement of windows on the screen	Global	2	419
15	Interface loses definitions under certain conditions	Global	2	419
16	Users experienced difficulty while prioritizing tasks	Local	2	419
17	Users have difficulty entering data into "Data Definition" window	Local	2	420
18	Interface loses pathnames	Local	2	420
19	Users experience difficulty finding an element in predefined list	Local	2	421
20	User has difficulty unselecting element from the contains list	Local	2	421
21	Submitting a password takes excessively long	Local	2	421
22	User believes colors mean too many different things	Global	3	422
23	Interface offers no feedback to indicate a window is already visible	Global	3	422
24	Windows do not resize gracefully	Global	3	422
25	Users confused when help windows grab and maintain focus	Global	3	422
26	Users confused that changing name and saving creates a new definition	Global	3	423
27	Users dislike pathnames entry	Global	3	423
28	Hoard Walk and Advice indicators confused	Global	3	423
29	Hoard Walk and Task indicators confused	Global	3	423
30	User experiences window overload	Global	3	424
31	Interface mixes upper- and lower-case letters	Global	3	424
32	User confused by terminology	Global	3	424
33	Meters shown at 1% have no visible color	Global	3	425
34	<Enter> does not make the help window disappear	Global	3	425
35	Numbers are sometimes unreadable on meters	Global	3	425
36	Users confused by addition of empty entry widget	Global	3	425
37	User surprised indicator lights are dynamic	Local	3	426
38	User confused by the behavior of the Hoard Walk indicator	Local	3	426
39	Users confused by two-color blinking	Local	3	426
40	Users somewhat confused because the indicator stops blinking	Local	3	427
41	User disagrees with default notification for certain events	Local	3	427
42	Users search for a way to configure events	Local	3	427
43	Users unsure how an event will be notified	Local	3	428
44	User assumes system uses a 24-hour clock	Local	3	428
45	Users confused by the meters in the "Space Information" window	Local	3	428

(continued on next page)

(continued from previous page)				
46	User confused by “questionnaire already running” dialog	Local	3	429
47	The Tix tree widget requires three clicks to expanding/contracting nodes	Local	3	429
48	User dislikes having fetch estimates expressed in seconds	Local	3	429
49	User confused by “unselect” dialog	Local	3	429
50	Task definitions unnatural	Local	3	430
51	Users confused by the cache meter in the “Task Information” window	Local	3	430
52	Newly hoarded task may not be visible in list of hoarded tasks	Local	3	430
53	Users confused by the task meters	Local	3	430
54	User finds it strange that a single click selects an item for the contains list	Local	3	431
55	Interface double-selected elements under certain circumstances	Local	3	431
56	Extraneous underlining appears in lists of the “Task Definition” window	Local	3	431
57	Priorities need to be labeled	Local	3	431
58	Meter may hide long task names	Local	3	431
59	User types a pathname into the name of a program definition	Local	3	432
60	User doesn’t believe system verifies existence of executable programs	Local	3	432
61	Meters do not cover orders of magnitude or relative information very well	Global	4	432
62	User requests balloon help	Global	4	433
63	User believes task interface to be overly complicated	Global	4	433
64	Interface is cognizant of vertical screen real estate, but not horizontal	Local	4	433
65	Users would like help on the indicator lights window	Local	4	434
66	User concerned about missing an event notification	Local	4	434
67	User would like feedback regarding the “health” of Venus	Local	4	434
68	User would like different configuration settings	Local	4	434
69	User would like a set of system defaults	Local	4	435
70	User wants to modify all event notifications at once	Local	4	435
71	User would like to be able to stop long transfers over weak networks	Local	4	435
72	User would like a count-down timer on advice requests	Local	4	435
73	User wants expert help for Space problems	Local	4	436
74	The Network help screen should explain why the network is degraded	Local	4	436
75	User wants the Task help screen to explain why a task is unavailable	Local	4	436
76	System should predict if hoarding a task would cause space problems	Local	4	436
77	User would like a tree view of tasks	Local	4	437
78	The “Data Definition” window should indicate the size of subtrees	Local	4	437
79	User wants more from program definitions	Local	4	437
80	User expects interface to track and automatically hoard missing objects	Local	4	438
81	Bogus fetch statistics confuse user	Local	4	438
82	Users confused when a file becomes unavailable	Local	4	438
83	User uncertain whether or not a program is available	Local	4	438
84	Users express concern over difficulty knowing what data is required	Local	4	439
85	User would like a graphical representation of what is in the cache	Local	4	439
86	User would like a visual representation of Venus’ on-going activities	Local	4	439
87	Users may feel limited by constraint that only tasks can be hoarded	Local	4	440

Figure 7.4: Summary of Findings

This table summarizes the findings that appear in Appendix F. For each, I give the finding identifier number, a brief description of the finding, and the page number of Appendix F on which the complete description can be found.

This section begins by presenting the analysis of quantitative data collected during the Tutorial and the Exercises from the six expert and novice users. It then presents a summary of the qualitative data collected from a questionnaire administered immediately after these participants completed the Exercises. It continues by presenting an analysis of the comments elicited from users during the course of the test. The analysis of the comments includes data from all participants, including the expert, novice, pilot, and miscellaneous users.

7.5.1 Tutorial

In this first section, I use the timings collected as the expert and novice users learned about the interface to identify individual screens of the Tutorial that caused these users difficulty. I then reanalyze the data, isolating novice data from expert data to identify screens that presented difficulties to only one of the two groups.

7.5.1.1 Identifying Difficulty Screens

To identify difficult screens, I use a simple heuristic. I begin by calculating each user's reading rate on each screen of the Tutorial. I then calculate each user's average reading rate on this material as well as the standard deviation. I hypothesize that the distribution of each user's individual screen reading rates is approximately normal, making the average and standard deviation both meaningful metrics. The heuristic then defines a *difficult* screen for a particular user to be one for which that user's reading rate on that screen is more than one standard deviation below that user's average reading rate. A *very difficult* screen is more than two standard deviations below the user's average. I define *easy* and *very easy* in a similar manner. By examining the number of users who found each screen very easy, easy, neutral, difficult, or very difficult, I can estimate the difficulty of each screen of the Tutorial.

The key to this metric is calculating a user's reading rate on each screen of the Tutorial, given the following pieces of information:

- *Dwell time*—the time a user spent on the screen
- *Word count*—the number of words users were required to read
- *Operator count*—the number of operations (e.g., pointing the mouse, clicking a button, or typing a pathname) required to complete the screen

The operator count is derived from a Keystroke-Level Model [7] of each screen of the Tutorial (see Appendix E). The reading rate is calculated by the following equation:

$$ReadingRate = \frac{WordCount}{DwellTime - \sum_{k \in operators} N_k * T_k}$$

Basically, the time spent reading is calculated by subtracting the time spent performing actions from the time spent on the screen.

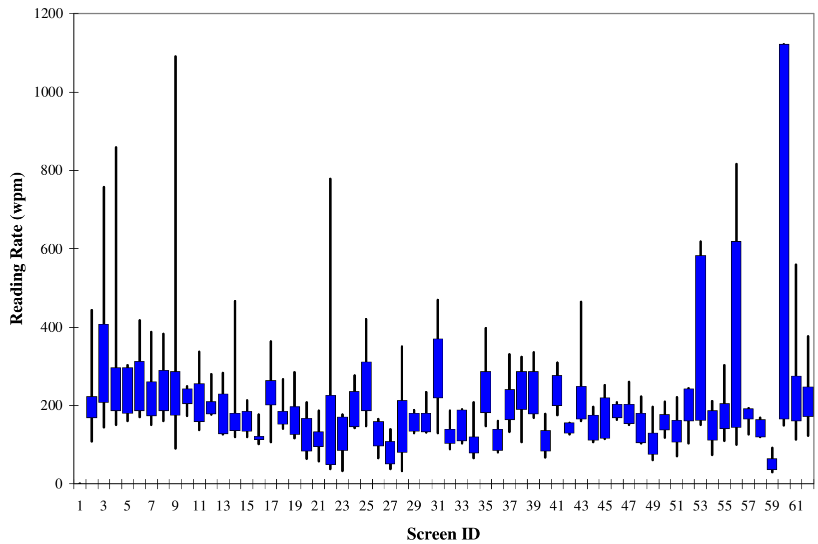
Figure 7.5(a) shows the range of reading rates observed over each screen of the Tutorial. The reading rates generally fall in the range of 100-300 words per minute. Given the educational background of the participants, I would expect their mean reading rate to be about 275 wpm for nontechnical topics [6]. The figure shows that our participants fall well below this expected rate on a number of screens—an indication that they were experiencing difficulty understanding the content of those screens. Perhaps the most striking observation is the number of extremely high reading rates observed on a few of the screens, some of which exceed 1000 wpm! The highest reasonable reading rate is about 650 wpm [7] so users who exceed this rate probably did not read every word on that particular screen. For this reason, I threw out any reading rates exceeding 650 wpm (a total of seven from four different users: N4, E1, E2 and E3). Figure 7.5(b) shows the data reanalyzed.

Using these data, I calculated the average reading rates over all screens of the Tutorial as well as the standard deviations for each of the expert and novice users. For each screen and each user, the heuristic identifies whether the given user found the particular screen very easy, easy, difficult, very difficult, or neither easy nor difficult. Figure 7.6 shows the Tutorial screen analysis visually.

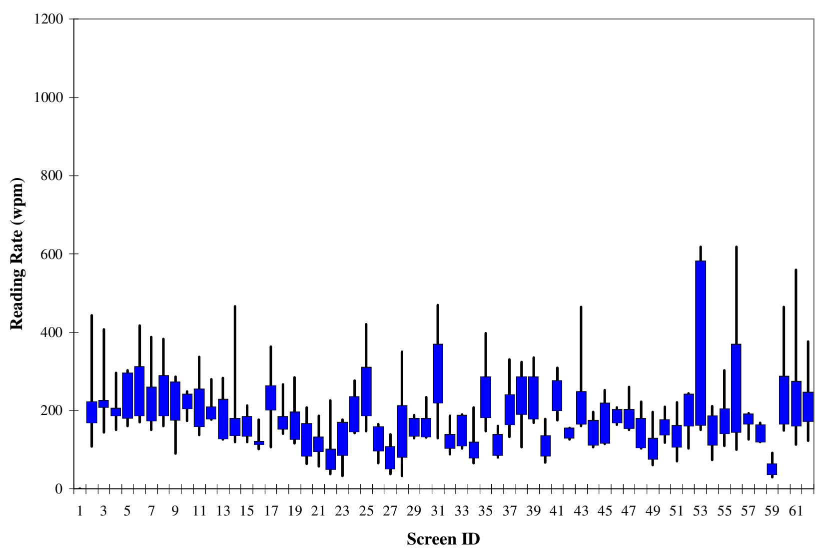
7.5.1.2 Examining Difficult Screens

Although I will not present every piece of evidence this analysis uncovered, I would like to highlight some of the more compelling ones. The analysis identified two screens (those numbered 27 and 59) that were difficult for five out of six of our novice and expert users! It also identified five screens (those numbered 22, 23, 28, 34 and 49) that were difficult for at least three users. In the paragraphs that follow, I will discuss each of these seven screens. The first one will be presented in detail and the remaining ones will be summarized (the complete information for all of them is available in the Appendices).

Screen 27 was identified as difficult for five out of the six novice and expert users! As reproduced in Figure 7.7, Screen 27 of Appendix B presents the way in which meta-information can be entered into the *Data Definition* window. This evidence, combined with numerous other pieces of evidence gleaned from other segments of the test, resulted in Finding 17. The detailed report of this finding appears in Appendix F, but is reproduced as Figure 7.8 for convenience. Each finding of Appendix F contains four sections: a brief description, a listing of evidence, a detailed explanation, and recommendations for addressing the problem. The evidence for the problems comes from five sources: violations of known heuristics, results from the Tutorial or Exercises, comments made on the Evaluation Questionnaire administered at the end of the test, and comments made spontaneously during the course of the test. A brief description of each piece of evidence is available in Appendix E. For this finding, every segment of the test contributed one or more pieces of evidence. The finding that resulted from the Tutorial,



(a) Reading Rates (original analysis)



(b) Reading Rates (sans skimming)

Figure 7.5: Range of Reading Rates

This graph shows the range of reading rates observed from the six novice and expert users. Each screen of the Tutorial is shown along the X-axis (Screen ID numbers correspond to those shown in Appendix B). The reading rate, in words per minute, is shown along the Y-axis. The thin bar shown for each screen represents the full range of reading rates; the thick bar represents the range after excluding both the fastest and slowest reader. Note that the initial screen (Screen 1) was not timed and that two users both read Screen 60 at the same, very high, rate. View (a) shows the range of reading rates from an initial analysis. View (b) shows the range after throwing out those data resulting from users skimming (rate > 650 wpm) the given screen.

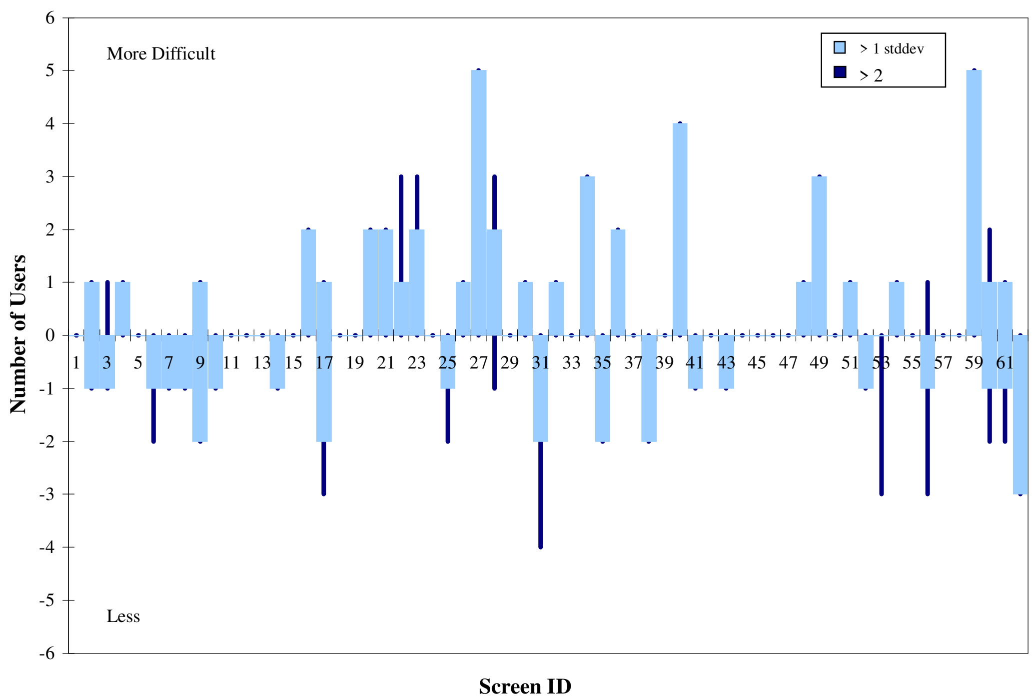


Figure 7.6: Screen Difficulty

This chart shows estimates of the difficulty of each screen of the Tutorial. The numbers on the X-axis represent Screen ID numbers and correspond to the numbered screens of Appendix B. The Y-axis shows the number of users. Each bar represents the number of novice and expert users who found the given screen easy or difficult (depending upon whether the bar appears above or below the X-axis). Each of the gray bars represent the number of users whose reading rate on the given screen was greater or less than one standard deviation above or below their average reading rate. Each black line represents the number of users above or below two standard deviations. Screens without bars were neither easy nor difficult for these users.

→ Please type in pathnames (feel free to make some up) that should be included in the new definition.

If the pathname you type is a file, then, when you hit <Enter>, an empty entry widget will appear, allowing you to enter the next pathname. If the pathname you type is a directory (yes, the system does check), then, when you hit either <Tab> or <Enter>, the meta-information area will become enabled, allowing you to choose the appropriate meta-information for this entry. If you used <Enter>, an empty entry widget will also appear and your cursor will be moved down. You'll need to use the mouse to select the appropriate meta-information. If you used <Tab>, you'll be able to use <Tab> and <Enter> to move around and select the meta-information respectively. If you continue to tab after selecting the meta-information, a new entry widget will appear and your cursor will also be moved down.

→ When you have completed the definition, click the “Save” button.

Figure 7.7: Screen 27

This figure shows the content of Screen 27 of the Tutorial. It is a duplicate of Screen 27 shown in Appendix B, and is reproduced here for convenience.

Finding #17: Users have difficulty entering data into "Data Definition" window
Evidence:

Heuristic Violation: Feedback
Error Messages
Prevent Errors (modal behavior)
Tutorial Results: 6
Exercise Results: 6
Qualitative Results: 2, 7, 9
Comment IDs: 3, 20, 36, 56, 101, 132, 196

Explanation: Users experienced two problems while defining user data elements. The first of these problems occurred when users entered a non-existent directory into the pathname entry area of the window. When this happened, users expected the meta-information area to become enabled (because they had entered the name of a directory). Because the system could not verify that the pathname entered was a directory, it did not enable the directory. There was no other feedback as to the problem. Indeed, the interface didn't even recognize the problem because it assumed the pathname to be a file. The second problem occurred when users tried to specify meta-information using <Tab> and <Enter>. Users were confused about the effect of these keys and were unsure about when each was appropriate to use.

Recommendations: The first problem is easy to remedy. The interface must check that the input pathname exists before checking to see if the input pathname is a directory. If the pathname does not exist, the interface must indicate this information to the user. The indication should take the form of a pop-up dialog box initially, but the interface might allow users to customize the indication to a beep or something similar. The second problem is more perplexing. The cause of the confusion is unclear to me. The <Tab> key advances the highlighting to the next area (as it does in many other windows and in other systems). The <Enter> key selects the current area. The solution may be to abandon these accelerators as "problem causing" and find different ones that cause less confusion. Users made a number of suggestions (indirectly by saying that they "expected" a certain key to have some effect). These suggestions included:

- Using <Back-Tab> to go backwards[1]
- Using the arrow keys to move around the meta-information[2]

One user also expected ^U to delete an entire pathname. Users also wanted to be able to click in the disabled meta-information area, though this desire may have been because they expected the meta-information to have been enabled for non-existent directories.

[1] Note that this expectation suggests that at least one user found the <Tab> intuitive.
[2] The right arrow key, then, would work like the <Tab> key does in the current implementation; the left arrow, then, like <Back-Tab> would.

Figure 7.8: Finding 17

This figure shows Finding 17 from Appendix F. There are four main pieces of information. The first section summarizes the problem. The second section gives a listing of the evidence supporting the finding. I examine five types of evidence: violations of known heuristics, results from the Tutorial or Exercises, comments made on the Evaluation Questionnaire administered at the end of the test, and comments made spontaneously during the course of the test. The third section provides a detailed explanation of the problem and the final section provides recommendations for solving the problem.

Tutorial Result 6, documents that “Five users had difficulty entering data into the ‘Data Definition’ window [Screen 27].” (The other pieces of evidence for this finding will be discussed below.) This evidence contributes directly to the second half of this finding, regarding the use of <Tab> and <Enter> as keyboard accelerators. Given the large number of users experiencing difficulty with this method of input, the recommendation is either to abandon or to supplement these accelerators. The arrow keys provided an obvious alternative set of keyboard accelerators.

Screen 59, reproduced in Figure 7.9, was also identified as difficult for five out of six of the expert and novice users. This screen asks users to examine two event configurations. Not surprisingly, this piece of evidence was supported by numerous other segments of the test. The basic problem is that in order to configure a specific event, users must know on which indicator that event is notified. The layout of this tab of the *Control Panel* is simply inappropriate to the task. It was redesigned during pilot testing based upon a suggestion made by the initial pilot user. The redesign introduced this usability problem. A detailed history of the problem, including the original complaint, appears in Finding 4.

Screen 22, reproduced in Figure 7.10, asks users to double-click on an element in one of two different lists. The KLM model [7] of this screen (see Appendix E for details) assumes that the user picked one of the lists and then double-clicked on the element in that list. If the user actually looked for the element under both lists before double-clicking on one of them, the additional time would not have been accounted for and the screen may have been mistakenly identified as difficult. Because it is impossible to model the screen for both of these behaviors, I choose to model the minimal actions required to complete the instructions. This decision leaves the analysis open to the possibility of falsely identifying this screen as difficult. As it turned out, however, this evidence was supported by three different comments (Comment IDs 18, 110 and 123) made by a total of six of the participants. Together, this evidence resulted in Finding 19, a Level 2 usability problem.

Screen 23, reproduced in Figure 7.11, introduced users to the *Data Definition* window by describing its contents verbally. Users apparently had difficulty orienting to this window. For example, when the screen contained a statement like “you should see X”, users then searched for X in the window. The analysis has likely resulted in a false-positive identification of this screen as difficult because the models do not account for the time needed to verify information between the instructions and the interface. This particular screen contained four pieces of information that users may have matched to the window. No evidence from other segments of the test supports this piece of evidence; therefore, no findings resulted from it.

Screen 28 asked users to close any open *Data Definition* windows. It contained a total of 19 words: “We have now completed our look into defining user data. Please close all windows titled ‘User Data Definition’.” This window lacks technical content and users had recently closed other windows (in Screens 14 and 17) so the explanation for their difficulty must lie elsewhere. Other similar screens (Screens 17, 53, 56) contained extremely similar content and directions, but did not present difficulties to users. The main difference between Screen 28 and these other screens is the wording “Please close

→ Under the "Advice" indicator, please select the "Read Disconnected Cache Miss" event.

Notice that this event is not configured to notify the user; therefore, the notification options are all disabled.

Please examine how the "Operating Disconnected" event (under the "Network" indicator) is configured. How would you describe this event and its configuration? The next screen gives the answer for you to verify your understanding of event configurations.

Figure 7.9: Screen 59

This figure shows the content of Screen 59 of the Tutorial. It is a duplicate of Screen 59 shown in Appendix B, and is reproduced here for convenience.

Now, let's look at the user data for the "writing thesis" task in more detail.

→ Please double-click on the "thesis" element in either the predefined user data list or the "`writing thesis' contains" list.

Figure 7.10: Screen 22

This figure shows the content of Screen 22 of the Tutorial. It is a duplicate of Screen 22 shown in Appendix B, and is reproduced here for convenience.

Double-clicking on the name of a user data set pops up a "User Data Definition" window showing the definition of the selected user data. The user data definition contains pathnames to files and/or directories. In the "thesis" definition, you should see three directories:

```
/coda/usr/bovik/thesis/
/coda/usr/bovik/thesis/dissertation/
/coda/usr/bovik/bib/
```

The first of these is the top-level of Harry Bovik's thesis area. The second is the directory in which he stores his dissertation. The third is the top-level of his bibliography directory.

You'll notice an empty entry widget after the bibliography directory. This empty entry is not part of the definition -- it just allows you to continue adding more paths to the definition.

Figure 7.11: Screen 23

This figure shows the content of Screen 23 of the Tutorial. It is a duplicate of Screen 23 shown in Appendix B, and is reproduced here for convenience.

all windows…”. If users had not deviated from the directions, they would have had just a single window titled ‘User Data Definition’ opened on their monitor. One possible explanation for their difficulty then is that they were confused by the wording and expected to see more than one open window. In fact, this explanation is supported by comments (Comment ID 34) made by one participant (T1). Another explanation is simply that they had more windows open and had to select the correct one to close (though that window should have been the one on top). The KLM models do not account for this selection. Because this screen contains just 19 words, any modeling errors, any approximation errors inherent in the KLM technique or any unrelated pause during reading (e.g., sneezing or yawning), will be pronounced in the reading rate. The screen simply contains too few words to amortize the effect of those errors. I conclude that this screen represents a false-positive identification.

Screen 34, reproduced in Figure 7.12, lead users through the process of modifying a task definition. Ten users commented on many aspects of this process during the course of the test (see Comment IDs 5, 6, 7, 23, 62, 123 and 181). Together, this evidence resulted in no fewer than five separate usability findings (see Findings 19, 20, 26, 54, 56) related to the *Task Definition* window. Any or all of these findings could have contributed to the difficulty users experienced on Screen 34. The recommendations for addressing these findings include a redesign to present the definitions in a tree format.

Screen 49, reproduced in Figure 7.13, asked users to complete the request for hoard walk advice, to notice that the indicator turned yellow, and to close the *Hoard Walk Information* window. Two primary problems caused users difficulty. Participant E1 noticed some bugs in the Tix widget library that required three clicks to expand a node of the tree; he also had problems with the small active area of the expansion/contraction widget. The second problem was caused by confusion over the *Task Available* event notification. A number of findings contributed to the difficulty users had with this screen; these are Findings 1, 10, 39, 40 and 47. Recommendations for addressing these problems include redesigning the way in which flashing event notifications occur and limiting the indicator from which this request for advice is accessible.

7.5.1.3 Comparing Performance of Novice and Expert Users

The next question I considered was whether or not these data showed a difference between novice and expert performance. Figure 7.14 represents a revised version of Figure 7.6 in which no distinction is made between very difficult and difficult or between very easy and easy. The different colored bars shown for each screen in this graph represent the contributions of novice and expert users. This graph shows that novice and expert users were fairly consistent in their performance, but the results of three screens were particularly interesting.

Screens 20 and 21 are interesting because two of the three novices found them difficult whereas no expert users did. Screen 20, reproduced in Figure 7.15, introduces the *Task Definition* window to users. One possible explanation for why the expert users did not

Your next goal would be to define the “writing with scribe” task. Now that you've saved the “writing with latex” task, you can modify it to create this new task. (Or you can go up, click the black down arrow, select the task named “New...”, and start from scratch.) We'll assume that you'll be modifying this task.

First, we should change the name of this task.

→ Click in the “Task Names:” entry area, fix the name, and hit <Enter>.

Next, we should remove LaTeX and xdvi since we can no longer use those tools.

→ Click on “latex” in the Programs area under the contains list.
→ Hit either the <Backspace> or <Delete> key.
→ Repeat for the “xdvi” entry.

Then, we will want to select the newly defined ScribeTools program definition.

→ Click on the “ScribeTools” in the Programs predefined list.

Finally, we want to save this definition.

→ Click the “Save” button.

Figure 7.12: Screen 34

This figure shows the content of Screen 34 of the Tutorial. It is a duplicate of Screen 34 shown in Appendix B, and is reproduced here for convenience.

→ Please request that all the tasks be fetched by using the “Fetch?” button.
→ Then, click on the “Finish Hoard Walk” button in the lower right corner.

Notice that the “Hoard Walk” indicator turns yellow (the warning color) briefly while the hoard walk finishes its simulated fetching of the requested files.

This completes our overview of the “Hoard Walk” indicator.

→ Please close the window labelled “Hoard Walk Information”.

Figure 7.13: Screen 49

This figure shows the content of Screen 49 of the Tutorial. It is a duplicate of Screen 49 shown in Appendix B, and is reproduced here for convenience.

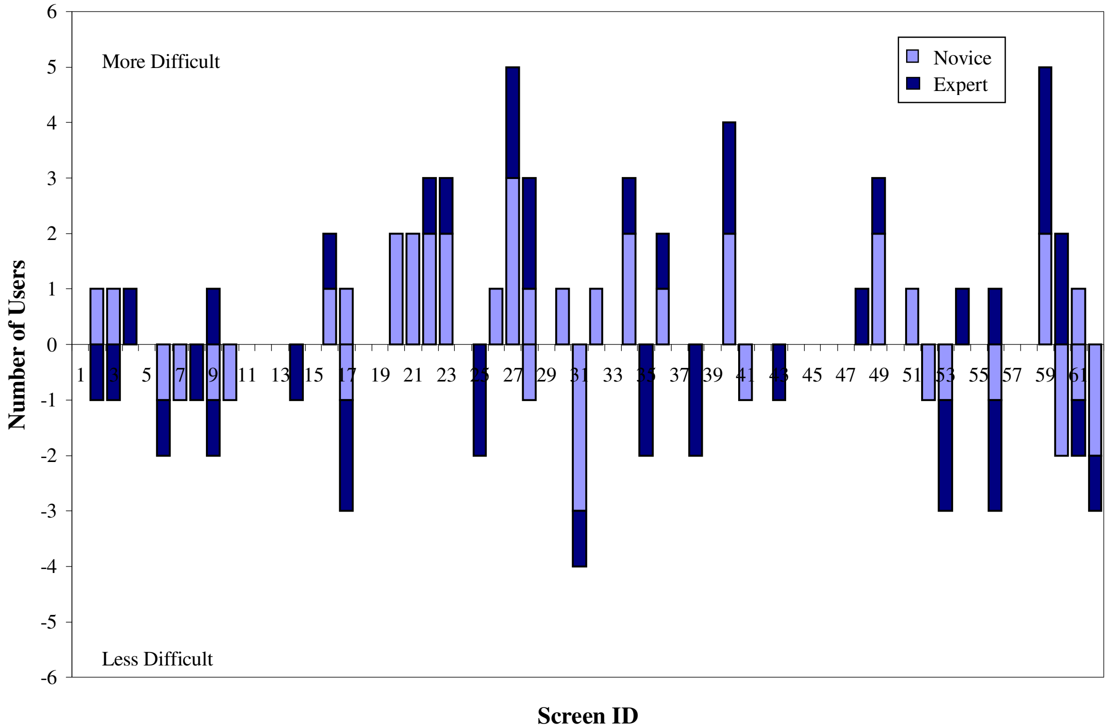


Figure 7.14: Comparison of Expert and Novice Contributions

This chart shows the contributions of expert and novice users to the difficulty rating of each screen of the Tutorial. The numbers of the X-axis represent screen numbers and correspond to the numbered screens of Appendix B. The Y-axis shows the number of users. Each bar represents the number of users who found the screen easy or difficult. The light-colored segments of each bar represent novice users; the dark-colored bars represent expert users. No distinction is made between difficult and very difficult or between easy and very easy in this chart.

find this screen difficult is because they may have heard about on-going research into "task-based hoarding" so the concept might have been seeded, allowing them to grasp the details more readily. Screen 21, reproduced in Figure 7.16, informs users that they can open task definitions by double-clicking in either of two places. For this screen, expert users should not have had any advantage.

Screen 60, reproduced in Figure 7.17, is also interesting because two of the three experts found it difficult while two of the three novices found it easy. Screen 60 contains the "answer" to a question asked on the previous screen (Screen 59 of Figure 7.9). I have no explanation for why experts found reading the "answer" difficult.

In general, the analysis does not show obvious evidence that novice performance and expert performance differs tremendously. Both novices and experts contribute to most of the screens categorized as easy and as difficult. The few screens where their performance differs do not suggest any problems that need to be addressed.

7.5.2 Exercises

In this section, I use timings as well as correctness and efficiency ratings to identify individual exercises that caused users difficulty. I examine the number of users receiving incorrect scores on each exercise. I then examine the number of users who required an excessive number of steps to complete each exercise. Finally, I look at the relative amount of time each user spent on each exercise as compared to the total time that user spent on the Exercises. The data presented in this section were collected from the six expert and novice users.

7.5.2.1 Number of Incorrect Responses

The first heuristic for identifying troublesome exercises simply counts the number of users receiving less than full credit for a given exercise. Figure 7.18 shows this metric visually. The X-axis represents exercise identification strings. The numbers on the Y-axis represent the number of users receiving less than full credit. Each bar represents the number of users who answered the given exercise incorrectly.

As seen in this chart, only three exercises caused more than one user to respond with incorrect answers. These exercises were ProvideHoardWalkAdvice, SpaceStatus, and FixingBugsTasks. I examine each of these exercises in detail.

The first exercise, ProvideHoardWalkAdvice (Screen 71 as reproduced in Figure 7.19), appears to have been poorly designed. This exercise set up a scenario and asked users to provide hoard walk advice in preparation for disconnected operation. The advice request showed three tasks needing data, two of which were relevant to the work described in the background scenario. In addition, those two tasks both needed to fetch data for the editing subtask. Users were expected to deduce that a previous (but still usable) version of the editor was available and that the data needing to be fetched represented a new (and

Double-clicking on the name of a task pops up a “Task Definition” window showing the definition of the selected task. This window has three main sections: one for subtasks, one for programs, and one for user data. Each of these three sections is split into two parts. The list on the left side shows a complete listing of all currently defined tasks, programs, or user data respectively. The list on the right side shows those definitions that are included in this task.

Let's look at the last main section---the one labelled “User Data”. The list on the left shows the name of every set of user data that has been defined to date (including “advice sources”, and “thesis”). The list on the right shows the subset of user data sets that have been included in the “writing thesis” task definition (just “thesis”). The other two sections, the one labelled “Subtasks” and the one labelled “Programs”, are similar.

By looking at the three lists on the right, we see that the definition for “writing thesis” contains the “writing” subtask, the programs for “ScreenCapture”, and the “thesis” user data.

Figure 7.15: Screen 20

This figure shows the content of Screen 20 of the Tutorial. It is a duplicate of Screen 20 shown in Appendix B, and is reproduced here for convenience.

In order to view another task definition, double-click on its name in either the “Task Information” window (as you just did) or even in the “Task Definition” window (often more convenient).

Figure 7.16: Screen 21

This figure shows the content of Screen 21 of the Tutorial. It is a duplicate of Screen 21 shown in Appendix B, and is reproduced here for convenience.

The “Operating Disconnected” event is configured to notify the user with a critical urgency level by flashing the “Network” indicator light but not beeping or popping up a window.

Figure 7.17: Screen 60

This figure shows the content of Screen 60 of the Tutorial. It is a duplicate of Screen 60 shown in Appendix B, and is reproduced here for convenience.

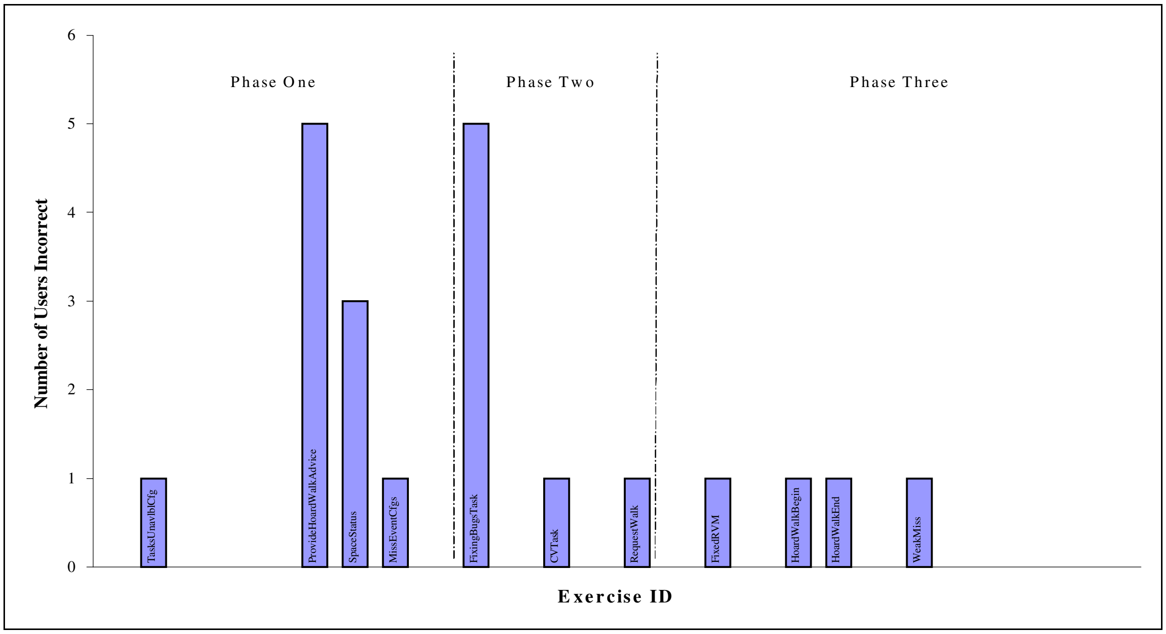


Figure 7.18: Number of Incorrect Responses

This chart shows the number of users answering each exercise incorrectly. The X-axis shows each exercise identifier; the Y-axis shows the number of users. The height of the bar represents the number of users who answered the given question incorrectly. Those exercises without a bar were answered correctly by all six novice and expert users. Only three exercises caused more than one user problems.

Now, suppose you are attending a PI Workshop (attendance required to continue to receive grants from this particular funding agency) in Nowheresville, IA. You had to arrive a day and a half early to get cheap tickets and now you're bored to tears. Thankfully, you have plenty of work to do while you wait, but unfortunately, you were unable to hoard prior to leaving for the airport.

You need to hoard two tasks at this time. First, a colleague asked you to preview a submission for SOSP16. You need to pick up the latest version and read it so that you can send comments back to the authors. If you have time, you'd also like to work on a few letters of recommendation. These two tasks have already been defined ("SOSP16 Paper" and "Letters of Recommendation").

You have selected two tasks to be hoarded and have recently made a connection via an (expensive) long-distance telephone line. Just a minute ago, you requested that a hoard walk be performed. Now, you notice that the hoard daemon has requested advice. Please examine the request, decide what objects should be fetched at this time, and describe your reasoning below.

Figure 7.19: Screen 71

This figure shows the content of Screen 71 of the Exercises. It is a duplicate of Screen 71 shown in Appendix B, and is reproduced here for convenience.

presumably unnecessary) version. Information vital to this conclusion was available under the *Task* indicator. Users had just completed an exercise in which they looked at the relevant information, but few of them made the connection.

Users received one point for not selecting the irrelevant task and a second point for not selecting editing. All six of the novice and expert participants realized[29] that the third task was irrelevant to the scenario. Five of the six, however, fetched the editing subtask. Reconsidering the exercise, the level of understanding needed to complete it correctly is very high indeed. The real goal of this aspect of the exercise was to observe whether or not users could give detailed hoard advice. Examining the advice given by each user in this light, I find that all six users explored the details and that four of them specified more detailed advice than just “fetch the two relevant tasks”. Overall, the user’s ability to understand what the hoard daemon was planning to fetch is encouraging. From my experience with this exercise however, I must conclude that if we expect users to avoid fetching new versions (upgrades) of binaries, we must provide them with more explicit information. Findings 75, 82 and 83 all address this basic issue.

The second exercise, SpaceStatus (Screen 72 as reproduced in Figure 7.20), asked users about the status of the cache, the local disk, and the RVM. Three users performed one or more components of this exercise incorrectly. It appears that the questions were poorly worded. Two users misinterpreted the first question of this exercise, regarding the current space status, and responded that the cache was two thirds full (which, incidentally, was correct). Two users also answered the second question, regarding how much disk space is dedicated to the Coda cache, incorrectly. One of these users answered in the wrong units, percent rather than MB, but provided the correct value for those units. The other user gave the size of the local disk, perhaps reading the value from the wrong meter or perhaps not reading the entire question. Although three users answered these questions incorrectly, their mistakes suggest that the two questions involved were easily misinterpreted. I conclude that the interface is not the primary cause of trouble.

The final exercise, FixingBugTasks (Screen 76 as reproduced in Figure 7.21), asked users to hoard data related to fixing Coda bugs. The results of this exercise are quite disturbing because only a single user completed it successfully. All of the remaining users failed to hoard the `compress`-related programs (the bug reports were stored in compressed format). Two of them also failed to include the editing task as a subtask, though they

Please describe the current space status.
How much disk space is dedicated to the Coda cache?
What percentage of the RVM space is utilized?

Figure 7.20: Screen 72

This figure shows the content of Screen 72 of the Exercises. It is a duplicate of Screen 72 shown in Appendix B, and is reproduced here for convenience.

[29] One user (N4) identified those objects she intended to select, but failed to complete the actions required to carry out those selections. My analysis is based upon her reasoning rather than her actions.

You are behind on addressing recent bugs reported by users. You'd like to take this occasion to catch up. You'll need the Coda sources, the bug reports, and the build environment.

The sources to the code that you must modify are located in three places:

```
/coda/usr/bovik/src/venus/
/coda/usr/bovik/src/advice/
/coda/usr/bovik/src/console/
```

You've squirreled the original reports away in

```
/coda/usr/bovik/src/bugs/
```

The programs you need to build Coda have been defined as ``building coda''.
The last time you hacked disconnected, you realized that you had forgotten to include gdb in your building Coda definition. It can be found in

```
/coda/misc/gnu-comp/i386_mach/omega/bin/gdb
```

Please create a task for fixing Coda bugs.

Figure 7.21: Screen 76

This figure shows the content of Screen 76 of the Exercises. It is a duplicate of Screen 76 shown in Appendix B, and is reproduced here for convenience.

each hoarded it separately at a lower priority. The failure of so many users to completely specify task definitions highlights a serious deficiency in the interface (see Finding 3), one that must be addressed. The failure of two users to hoard the editing task as a subset of their definition or even as a high priority top-level task suggests users may not fully understand the consequences of their chosen priorities (see Finding 11).

7.5.2.2 Number of Inefficient Responses

The second heuristic for identifying troublesome exercises counts the number of users requiring an excessive amount of time to complete a given exercise. I consider users to have taken an excessive amount of time if the number of steps they required to complete an exercise exceeded twice that required by an expert user of the interface with knowledge of the correct answers (2*par). Figure 7.22 presents this metric visually. The X-axis shows exercise identification strings. The numbers on the Y-axis represent the numbers of users requiring excessive amounts of time. The dark gray segments of each bar in the chart represent users exceeding 2*par. The light gray segments represent those exceeding 3*par.

As seen in this chart, a number of exercises required excessive numbers of steps for multiple users. In this section, I examine the causes for five of these exercises: ProvideHoardWalkAdvice (Screen 71), MissEventCfgs (Screen 73), HeaderFilesTask (Screen 77), CVTask (Screen 78), and PrioritizeTasks (Screen 79). The other exercises identified were either only troublesome for a single user or had par values under 5 with no individual user’s score exceeding 10. In either of those cases, I did not consider the exercise problematic.

The first exercise identified by this metric, HoardWalkAdvice (Screen 71 as reproduced in Figure 7.19), had a par of 10 and a range of user scores from 20 to 46. This exercise asked users to provide detailed advice to the hoard daemon. Users spent time exploring the details of what objects needed to be fetched. Five to ten node-expansion/contraction pairs would account for 10 to 20 points. Add to that some allowance for users to select some elements and deselect them during the normal process of problem solving, and the fact that so many users exceeded par by 10 to 36 points is not unreasonable. From these data alone, I would not conclude that this exercise presents a significant problem to users, though it could play a supporting role when combined with other evidence of a problem. No other evidence of such a problem appeared during the course of the user test.

The second one identified was the MissEventCfgs (Screen 73 as reproduced in Figure 7.23) exercise. In this exercise, users were asked to describe the configuration of the Weakly Connected Cache Miss Advice event and the Disconnected Cache Miss Advice event. Two users took excessively many steps to complete this exercise. In both cases, their activities reveal that they had difficulty finding these events in the *Event*

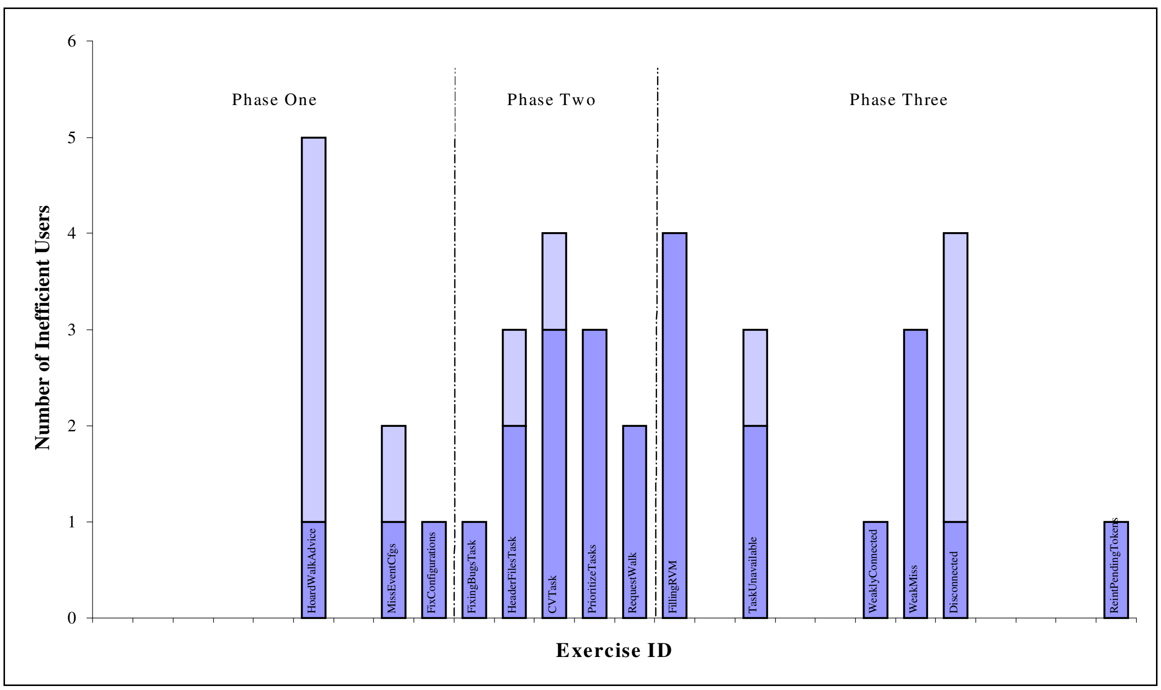


Figure 7.22: Number of Inefficient Responses

Each bar in this chart shows the number of users whose actions were inefficient, meaning the number of actions required to complete an exercise exceeded par substantially. The X-axis shows each exercise identifier; the Y-axis shows the number of users. Each bar represents the number of users who performed inefficiently on a given exercise. The dark gray segment of the bar represent those users exceeding 2*par; the light gray segments represent those exceeding 3*par. Users experienced the most inefficiency during Phase Two and while providing hoard walk advice. The tasks in Phase Three could be completed with a small number of actions. Any additional exploration could cause a user's efficiency to exceed two or three par.

Configuration tab of the *Control Panel*. The fundamental problem is that to find an event on this window, the user must know on which indicator that event is notified. Often it is not too difficult to guess because the name of one of the indicator lights appears in the name of the event (e.g., One or More Tasks Unavailable). In the case of these two miss events, however, three indicators are possible—*Network*, *Space*, and *Advice*. This exercise provides another piece of evidence for Finding 4.

Users who experienced numerous minor inefficiencies caused this metric to identify the third and fourth of these exercises, HeaderFilesTask (Screen 77 as reproduced in Figure 7.24) and CVTask (Screen 78 as reproduced in Figure 7.25). Some users encountered bugs that caused the system to lose data, forcing them to reconstruct it. Some explored the definitions to see what was available for them to reuse (as the instruction, "Don't forget—you can reuse data, program, and task definitions!", in Screen 75 told them to do). These minor inefficiencies account for those users exceeding 2*par. The user (E3, an expert) exceeding 3*par on the HeaderFilesTask exercise used `@sys` in a pathname, failing to heed the instruction of Screen 75 to "pretend that the machine you are running on is an i386_mach box". Although the other two experts also failed to heed this directive[30], they each recognized their deviation from the directions relatively quickly. User E3, however, struggled for quite some time until he finally realized the problem. The interface contributed substantially to his confusion because of the lack of feedback when a directory does not exist (see Finding 17). The instructions also contributed to the confusion in not specifically mentioning the type of machine actually in use. The user exceeding 3*par on the CVTask exercise (N1) experienced many minor inefficiencies, one of which was also the result of the lack of feedback for nonexistent directories (this time caused by a typographical error). Although no single problem accounts for all of the inefficiencies experienced by users during these two exercises, the combination of a small number of serious bugs and lack of appropriate feedback accounts for much of it. Future versions of the interface must address these problems (discussed in Findings 15 and 17).

The final exercise identified by this metric was the PrioritizeTasks (Screen 79 as reproduced in Figure 7.26) exercise. In this exercise, users were asked to prioritize the tasks they defined in preparation for disconnection. Three users exceeded 2*par on this exercise. Two of them experienced difficulty with the interface while prioritizing their tasks. Two of them (one of those two users and the third user) were forced to reorganize their task definitions before prioritizing them to make the HeaderFilesTask a top-level task rather than a data definition within the BugFixingTask. They were forced to reorganize their definitions because only tasks can be prioritized and because the instructions asked them to give the HeaderFilesTask a lower priority. The instructions strongly suggested the appropriate organization, yet the intuition of these users led them to a different organization. This behavior suggests that users may feel limited by the constraint that only tasks may be prioritized and that this constraint may need to be reconsidered (see Finding 87).

---

[30] It's interesting to note that all three expert users attempted to use `@sys` whereas not a single novice user did so. Because `@sys` is a feature of not only Coda but also AFS, this difference illustrates a point made early in this chapter that developers (and their friends) are not representative of all computer scientists.

Please describe the “Weakly Connected Cache Miss Advice” and the “Disconnected Cache Miss Advice” events and their configurations in the space below.

Figure 7.23: Screen 73

This figure shows the content of Screen 73 of the Exercises. It is a duplicate of Screen 73 shown in Appendix B, and is reproduced here for convenience.

Oh, and the last time you hacked from home, you wanted to take a look at some header files that you had forgotten to hoard. They aren't really important, but it'd be nice to have them around. Unfortunatley, even though each individual header file is relatively small, together they take up quite a lot of space -- so you don't want to just add them to your ``building coda'' programs since then they might be hoarded in preference to really important source files. They are located in several places:

```
/coda/project/coda/alpha/include/
/coda/project/coda/alpha/i386\_mach/include-special/
/coda/misc/c++/i386\_mach/alpha/include/
/coda/misc/tcl/common/beta/include/
```

Figure 7.24: Screen 77

This figure shows the content of Screen 77 of the Exercises. It is a duplicate of Screen 77 shown in Appendix B, and is reproduced here for convenience.

You need to write your CV before Tuesday when you will be talking to a representative from AT\&T Labs (assuming, of course, that the visitor can get here and that you can get to school). Since you haven't yet written your CV, you've borrowed the file for your advisor's CV to use as an example. Unfortunately, he still uses Scribe! Since you're time-pressured, you've decided to stick with Scribe. Please create a task for writing your CV. The necessary files are located in:

```
/coda/usr/bovik/personal/cv/
```

Unlike LaTex, Scribe outputs PostScript directly. To run Scribe, type ``scribe cv.mss''; it will produce a ``cv.ps'' file that you can preview with ``gv''. You may assume that the binaries for both scribe and gv are located in /coda/misc/bin.

Figure 7.25: Screen 78

This figure shows the content of Screen 78 of the Exercises. It is a duplicate of Screen 78 shown in Appendix B, and is reproduced here for convenience.

Finally, you'll need to prioritize these tasks. Recognize that the data needed for all these tasks may or may not fit into your cache -- the priorities you choose now will decide what (if anything) gets left behind. To help you in prioritizing these tasks, we've listed all of the tasks you would like to work on during the blizzard, as well as what they are intended for. (The list below is in no particular priority order.) If you haven't already hoarded these tasks, please do so now. When you have finished, please use the space below to describe your reasons for choosing the priorities you did.

Fix bugs reported by users.
Various header files to support bug fixing.
Write your CV to give to this visitor on Tuesday.

Figure 7.26: Screen 79

This figure shows the content of Screen 79 of the Exercises. It is a duplicate of Screen 79 shown in Appendix B, and is reproduced here for convenience.

7.5.2.3 Relative Exercise Dwell Time

The third heuristic for analyzing user performance during the Exercises looks at the relative amount of time each user spent on each exercise as compared to the amount of time that user spent on the entire set of exercises. Figure 7.27 shows the data visually. Along the X-axis, I list exercise identification strings. Each bar along the Y-axis represents the percentage of time one user spent on the given exercise as compared to their total time spent on the Exercises. Each exercise shows six bars, one per user.

The chart shows that this metric confirms my previous analysis. Users spent a relatively large amount of time on six of the seven exercises previously identified as difficult. The fact that the seventh exercise, SpaceStatus, is not identified here supports my conclusion that the interface did not cause users difficulty. This metric shows that users did not spend a large amount of time on this exercise. This fact suggests that neither the window nor the questions confused users, supporting my earlier conclusion that the interface was not the primary cause of trouble with this exercise.

One obvious feature of this chart is the large percentage of time participant E1 spent on the DefineBugsTask exercise. This user explored the definitions during the process of defining this task, as he was instructed to do in Screen 75. This exploration is simply reflected here.

This chart does make two interesting points. First, expert users seem to spend less time than novice users providing hoard walk advice, as evidenced by the fact that the slowest expert completed this exercise more quickly than the fastest novice. My explanation for why this might be is that this idea was familiar to each of them prior to participating in the test. Second, with that one exception, the novice users and the expert users spend similar relative amounts of time on each exercise.

7.5.3 Evaluation Questionnaire

In this section, I present a summary of data collected from the questionnaire administered immediately following completion of the Exercises. The data presented were collected from the six expert and novice users. I highlight those responses that address the interface.

Users found the interface easy to use. On a five point scale, where five was "very easy" and 1 was "very difficult", users rated the interface a 4.5. One expert participant commented that he was somewhat confused regarding which indicator to open. The Debriefing revealed that the user was confused regarding the distinction between the *Task* and *Hoard Walk* indicators (see Finding 29).

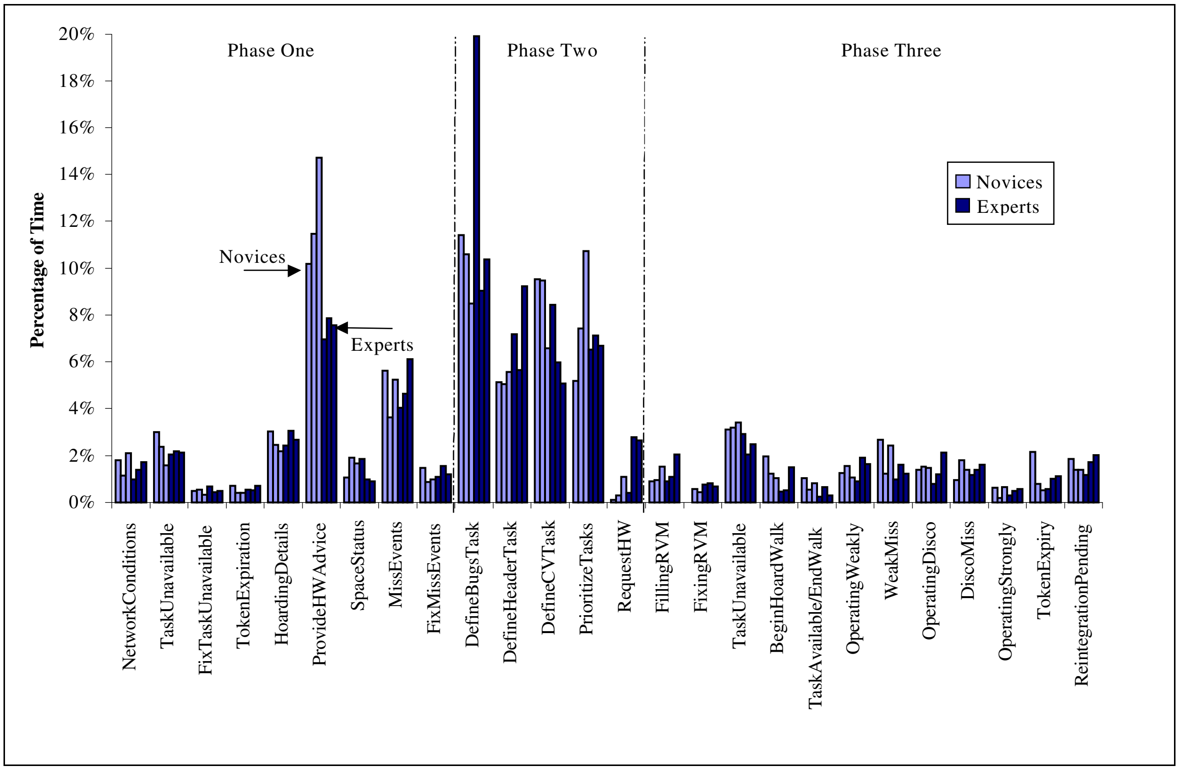


Figure 7.27: Relative Dwell Time

This chart shows the percentage of time each user spent on each exercise. The X-axis identifies the exercises. The Y-axis shows percentage of time relative to the total time the user spent on the Exercises segment of the test. For each exercise, the six bars represent participants N1, N2, N4, E1, E2, and E3 respectively. The light-colored bars represent the novice users; the dark-colored bars represent the experts. The maximum percentage of time spent on any individual exercise was just under 20%. The interesting observation about this graph is that it is not particularly interesting. The expert and novice users both spend about the same relative amount of time on each exercise. The only notable difference is that the novices tend to spend a larger percentage of their time providing hoard walk advice than the experts do. In addition, user E1 spent an excessively large percentage of his time defining the "BugsTask". This user explored the definitions extensively during the process of defining the task. He was complying with the suggestion to reuse definitions.

Users ranked their ability to find the features they wanted in the indicator lights a 4.2 (again, on a five point scale where 5 meant "very easy"). The two users who ranked this question as "neither easy nor difficult" both commented that it was difficult to find events in the *Event Configuration* tab of the *Control Panel*. These comments provide an additional piece of evidence for Finding 4.

Users reported that they were likely to keep the indicators lights visible on their monitor all (or almost all) the time. Their average ranking was a 4.3 out of 5 (where 5 meant "very likely"). This ranking is meaningful because users highly value "screen real estate".

Four users felt that the interface provided an appropriate amount of detail, but two felt that it provided too little. In particular, the two users commented that help was not helpful (see Finding 2) and that the "no such file" indication was too subtle in the *Data Definition* window (see Finding 17).

Users found the concept of hoarding easy to understand; their average ranking was a 4.5 out of 5. They also found that preparing for disconnected operation was easy; their average ranking was a 3.8 out of 5. Furthermore, they were highly confident in their preparation; their average ranking was 4 out of 5. These rankings suggest that users were completely unaware of the subtle mistakes they had made in creating their task definitions.

When asked which aspects of the interface they liked the most, users responded with one or more of the following:

- The ability to define tasks (4 users)
- The ability to see the status of the system (3 user)
- The ability to configure event notifications (2 users)

When asked which aspects of the interface they liked the least, users generally responded with unique answers. In fact, the only answer to be reported more than once was a complaint regarding the way in which meta-information was specified in the *Data Definition* window (see Finding 17).

7.5.4 Verbal Protocol

As part of the analysis, I transcribed portions of the verbal protocol collected from all thirteen users for whom I have recordings, including pilot and miscellaneous users and the novice user whose quantitative data was thrown out (see Section 7.3.1). Afterall, just because a user was a pilot, does not mean that their comments are not valid. My goal in transcribing these tapes was not to have a complete record of each session, but to have a record of the critical incidents. In particular, I did not transcribe segments during which the user read verbatim from either the Tutorial or the Exercises except when necessary to provide context. Appendix D contains the transcripts of each session. Figure 7.28 contains an excerpt from the transcripts of participant E1. As seen in this figure, the

Time	Transcript	ID
	So, walls…I'm guessing walls is this red outline (around the xterm).	285
7:29	[CodaConsole started] Okay, something just popped up. A little box. Control Panel, Tokens, Space, Network, Advice, everything is green. I'll bet that's good.	
8:47	[Color blind– other cultures] OK. So, I can do those colors. I wonder…so, traffic lights have that, th– the ordering and I wonder if something like that might work here too.	
12:55	One of what servers? Had the color been red instead of yellow, the indicator would have been showing that our connection to at least one server had been severed. OK. Double-click on the network indicator for more detail regarding our connection to individual servers. Aha! So that will tell me. I see. OK. Cool.	
13:29	So what if we have 100 Mb/s? Can that be scaled I wonder.	50
14:15	Soon the bandwidth to scarlatti will drop to 0, which it just did, and Oh! It's blinking yellow and red? Ah, I thought it would have just changed red. This indicator flashes red and the network information window automatically updates to show the change in scarlatti's estimated bandwidth. OK. Oh, so it stops flashing... So maybe that's just to draw my attention.	9 33
16:00	[First impressions of Task Information window.] So, cache space. Alright. So the task, I'll read, instead of trying to figure it out. Although, let's see, so it looks like there are 97 MBs available and that 70% of it is either available, 70% of that 97 is available or 97 MB is available. Let's read.	269
	It's too bad this window manager seems to pop everything up up there.	1
22:10	[First impression of Data Definition.] So it looks like a hoard database shown in widget form.	
22:15	So I wonder if there's they have the old hoard now and forever or just now notation as well, I wonder, I'll bet it's just now and forever.	
22:30	So I wonder if they gray out if it's a file.	
23:00	Meta-information is a weird name here.	209
23:30	Ah, yes, so they are grayed out when nothing's there.	
	Interesting that it's not <Tab> [after naming the data definition, you hit <Enter> to move the cursor down into the definition area.]	3
24:35	[Task10: <Tab>/<Enter> explanation for data definition meta-information.] Alright, I'm confused. "If the pathname you type is a directory, then, when you hit either <Tab> or <Enter>, the meta-information area will be enabled." Alright, so let's go see if I can rename this. Ah... Better not try that. <Tab> just selects the window and another <Tab>. OK. Hit either <Tab> or <Enter> the information window "will become enabled, allowing you to choose the appropriate meta-information for this entry. If you used <Enter>, an entry empty entry." OK. So <Tab>, <Tab> sends me to the meta-information entry as does <Enter> and <Enter> additionally adds an empty entry widget. OK. "You'll need to use the mouse to select the appropriate meta-information. If you used <Tab>", mouse, no keyboard shortcuts, "you'll be able to use <Tab> and <Enter> to move around and select the meta-information." Ah! OK. "If you continue to <Tab> after selecting the meta-information, a new entry widget will appear." Well, let's see.	3

(continues)

Figure 7.28: Excerpt from E1 Transcripts

This figure shows an excerpt from the transcripts of the verbal protocol produced by participant E1 during the Tutorial. The complete transcripts appear in Appendix D. The left-hand column of the table gives the timestamp relative to the beginning of the Tutorial, when known. The middle column shows the comments made by the participant, though any comment appearing within square brackets is a note from the transcriber. The right-most column provides indices into the comment listing of Appendix E.

transcripts contain timestamps relative to the beginning of the segment (when those timestamps are known). In addition, they code individual comments made by the user, providing the ID of the comment listed in Appendix E.

Appendix E contains a complete listing of the comments coded from these transcripts in numerical order. This appendix also provides a table documenting which users made each comment during which segments (Tutorial, Exercises, Debriefing) of the test, an excerpt of which is shown in Figure 7.29.

The metric I used to analyze this data was the frequency with which comments were reported. Figure 7.30 shows this information. The X-axis shows the number of users. Each bar represents the percentage of all comments reported by a given number of users. From this chart, I can identify compelling comments as those made by many users. The chart shows, visually, what percentage of comments was made by a given number of users. As is typical in usability testing, most comments were made by just a single user—in this case 58% of them. In fact, of the remaining 42% of comments, more than half were made by just two users. Less than a quarter of the comments were made by three or more users, yet five comments were made by more than half of the users! For the purposes of identifying evidence of usability problems, my analysis examined any comment made by three or more users. Because of space limitations, I discuss only those comments made by five or more users.

As seen in Figure 7.29, ten users found two-color blinking (Comment 9) confusing. When an event configured to flash arrived, the appropriate indicator light would blink between the old urgency color and the new urgency color for about ten seconds. So, for example, if the current state of the *Task* indicator is green, then when the *Task Unavailable* event arrives, the *Task* indicator will flash green-red-green-red-etc. for ten seconds before settling on red. The idea was to show the previous state as well as the new state. Unfortunately, users found it confusing because, unless they remembered the old state, they could not determine what the new state was until the indicator settled on the final color. This evidence lead to Finding 39.

Nine users found that they had to guess which indicator a particular event was associated with when trying to configure an event (Comment 17). This comment, as well as two related comments (those numbered 162 and 206) provides the final piece of evidence for Finding 4. For more information regarding this problem, please refer to Section 7.5.2.2 and Appendix F.

Eight users commented that they wanted the interface to tell them the size of the subtree in the *Data Definition* window so that they could select meta-information more intelligently (Comment 57). Without a doubt, this information would improve the usability of the interface. Unfortunately, obtaining the information efficiently could prove problematic (see Finding 78).

Eight users also commented that they did not know anything about reintegration (Comment 239). The lesson here is that participants did not appreciate being asked to go beyond the lessons covered in the Tutorial. I did so to see how well users would generalize what they had been taught and apply it to a similar situation. They did very well, but clearly indicated that the topic had not been covered.

ID	Pilots 2	Pilots 3	Pilots 4	Pilots 5	Novices 1	Novices 2	Novices 4	Experts 1	Experts 2	Experts 3	Misc. C	Misc. S	Misc. T	Number Reporting	Unique User
1				T	T			M		A	T		T	6	
2	(See Comments #80 and 209-243)														
3				T	D	T		A	E	F		T		7	
4						T		F		E			T	4	
5					T	T		T	T	T			T	6	
6					F	F		A			E			4	
7			E					T		E				3	
8								T		F			F	3	
9	T		E	T	T		T	T	F	T	E		A	10	
10						T		T						2	

(continues)

Figure 7.29: Excerpt from Table Summarizing Comments

This figure shows an excerpt from a table in Appendix E that summarizes the comments made by participants during the course of the usability test. The leftmost column shows the Comment ID number (as listed in Appendix E). The next 13 columns indicate whether a given user made the comment, and, if so, during which portion of the test. A T indicates that the comment was made during the Tutorial, an E indicates Exercises, a D indicates Debriefing, an F indicates both Tutorial and Exercise, an M indicates both Tutorial and Debriefing, an L indicates both Exercises and Debriefing, and an A indicates Tutorial, Exercises, and Debriefing. The final two columns show the total number of users making the comment and user who made a unique comment (a comment made by only one user).

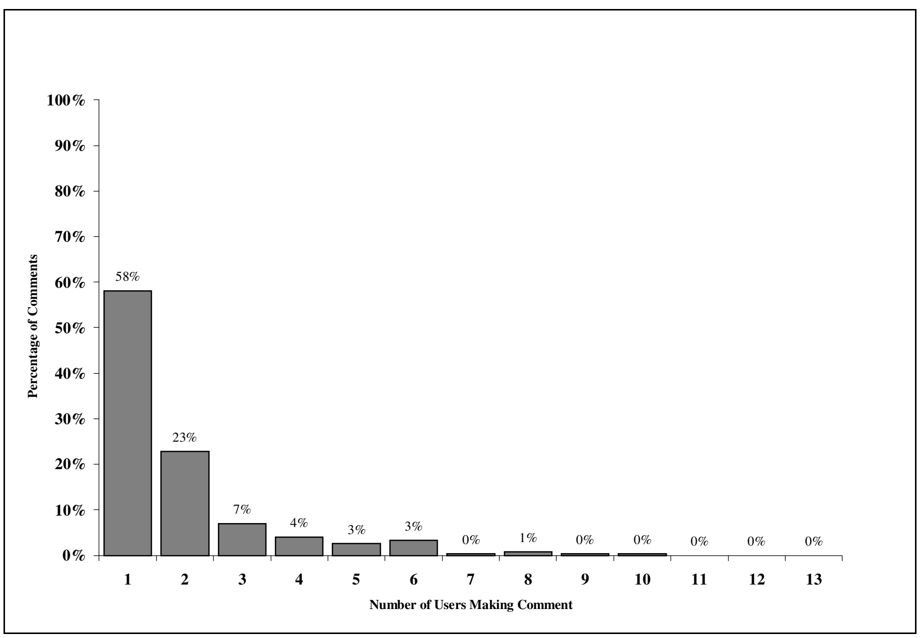


Figure 7.30: Comment Frequency

This chart shows the frequency with which verbal comments were made. Each bar represents the percentage of comments made by the given number of users. Thus, 23% of comments were reported by two users while only 1% of comments were reported by eight users. No comments were made by more than ten users.

Seven users had difficulty entering data into the *Data Definition* window (Comment 3). In particular, they had problems understanding when to use <Tab> and when to use <Enter>. This comment, combined with evidence from every other segment of the test, lead to Finding 17. For a detailed explanation of this problem, please refer to Section 7.5.1.2.

7.6 Usability Test: Findings

The results of the test show that the interface allowed the novice Coda users to use Coda with approximately the proficiency of the experienced Coda users. Novices scored 90% or better on exercises typical of those required of Coda users. They successfully diagnosed problems and possible solutions, defined tasks, and provided advice to the system about hoard walks and demand fetch requests. The interface clearly met each of its objectives.

Not surprisingly, the test also provided a wealth of information about how users interact with the interface and where they encountered problems. Analysis of this evidence reveals areas in which the interface could be improved. This analysis resulted in 87 usability findings. I classified each finding according to its scope and severity (see Section 7.5). Figure 7.31 shows the number of findings in each classification. 24% of the findings had severity level 1 or 2, and 26% had global scope.

Appendix F contains a complete listing of the findings, organized by their severity and scope. Because of space limitations, I cannot present each of these findings here. Instead, I present a different classification of these findings, one based upon the type of problem uncovered:

- Issues dealing with the approach
- Bugs in the interface
- Usability problems
- Omissions from the interface (either accidental or intentional)
- Confusing behaviors
- Suggestions for enhancement

For each classification, I highlight a number of findings. The specific ones highlighted were chosen for a variety of reasons: they might be representative of the entire class of problems; they might be particularly important; or they might have an interesting history. The findings presented in this section should be considered a sample of the type of problems I uncovered, *not* a "short list" of problems that need to be addressed.

7.6.1 Issues with Approach

The test uncovered a small number of usability problems that briefly caused me to question the wisdom of translucent caching. The first of these is user confusion over event notifications (see Finding 1). Although users were able to determine the cause of problems during Phase Three of the Exercises, they expressed confusion over what had happened at other points during the test. These observations were not limited to the Tutorial so they cannot be categorized as a learning effect. It is certainly reasonable to believe that user understanding will continue to improve with further exposure to the interface, but that does not solve the fundamental issue that certain event notifications were confusing to users. We must ask whether or not there is something the interface could do to reduce this confusion. In fact, these problems suggest that the interface needs to offer a central place where users can go for a written explanation of the current, or most recent, problems. In essence, just as keyboard accelerators help the user tell the interface what to do more quickly, the indicator lights offer the interface an accelerator to tell the user what has happened. They do not provide complete information, however, and so a more descriptive explanation is warranted.

A second usability problem that caused serious concern was the difficulty users had identifying which programs to include in task definitions (see Finding 3). Although users were highly successful in defining task definitions, five out of six of them failed to hoard a program necessary for one of their definitions. This mistake was one of very few that were made by any of the users, yet it was extremely common (even among the pilots and miscellaneous users). It has the potential to make user-assisted hoarding unusable, but it can be addressed with the addition of a new type of advice that warns users about missing programs. Effectively, the interface would have to issue a “watch” on each hoarded file. When a program accessed an object under watch, Venus would inform the interface. If a program accessing hoarded data is not also hoarded, the interface could indicate a potential hoarding problem to the user through the advice monitor at a reasonable time, perhaps at the end of a voluntary hoard walk.

Severity	Scope	Number of Findings
1	Global	1
	Local	10
2	Global	4
	Local	6
3	Global	15
	Local	24
4	Global	3
	Local	24

Figure 7.31: Classification of Findings

Each finding is categorized by its scope and severity. Scope is a measure of how widespread a problem is. A problem with local scope is limited to a single window; one with global scope applies to multiple windows. Problems with global scope are generally considered more critical than problems with local scope. Severity is a measure of how critical a problem is considered. A problem with severity 1 prevented a user from completing a task. Severity 2 problems caused significant delay or frustration. Severity 3 problems had a minimal impact on usability. Severity 4 problems represent more subtle issues and may suggest potential enhancements to the interface.

7.6.2 Bugs

No matter how well tested an interface seems, there is always another bug lurking in the shadows. During the course of the usability test, I found a total of 24 interface bugs (listed in Appendix E). Six of these bugs are critical: one crashes the interface (see Finding 3) and five cause the interface to lose data (see Findings 15 and 18). A number of others are urgent in that the system simply behaves incorrectly. For instance, in certain situations selecting an element in a task definition results in the element appearing in the list twice (see Finding 55). Similarly, when the user types a password into the interface to obtain tokens, the password is echoed in the window using a text color that is almost, but not quite, identical to the background color of the entry widget (see Finding 6). The monitor on which the test was performed allows the user to read the password, whereas the monitor on which the interface was developed did not. A number of other reported bugs were cosmetic in nature, though none of them were reported frequently enough to warrant an official Finding. For example, the labels associated with the radiobuttons on the *Event Configuration* tab of the *Control Panel* do not line up correctly (see Comment ID 79). Similarly, the widgets on that same window visibly shift when a new event is chosen from the listing (see Comment ID 135).

7.6.3 Usability Problems

The data showed that users experienced significant difficulty finding particular events in the *Event Configuration* tab of the *Control Panel* (see Finding 4). Two of the six participants completed the MissEventCfgs exercise inefficiently. During the course of the test, a few users resorted to brute-force search to find an event and one user failed to complete the exercise correctly because of this difficulty. The fact that nine users commented on the difficulty of finding events provides further evidence of this usability problem. This window of the interface must be redesigned such that users are not required to select the indicator on which an event is notified while configuring that event. The redesign must, however, show on which indicator the event will be notified.

Another important discovery was the difficulty users had entering data definitions (see Finding 17). In particular, seven users commented that they were confused by the distinction between <Tab> and <Enter>. Although the test clearly demonstrated that the means of entering these definitions was confusing for more than half of the users, it offered little in the way of explanation for their confusion or suggestions for improvement. The behavior of <Tab> and <Enter> is consistent with other uses of those keyboard accelerators. Thus, the recommended solution is to allow additional keyboard accelerators, such as the arrow keys, for those users confused by the behavior of <Tab> and <Enter>.

Suspiciously enough, a second important discovery also relates to data definitions. That is, when users enter the pathname of a directory that does not exist, the system provides only indirect feedback of the error, by not enabling the meta-information (see Finding 17). Six users commented about this lack of feedback. Furthermore, two users exceeded 3*par on a task definition exercise largely because of this lack of feedback. I hypothesize that this problem may be at least partially related to the previous one because if the meta-information is not enabled, <Tab> does not allow the user to move the cursor around the meta-information options. After entering a nonexistent directory, users may have been confused because <Tab> did not then work. The recommended solution to this problem is to add a dialog error message to inform users when a pathname that has been entered does not exist. A higher cost solution to this problem would be to add pathname browsing.

The data also show that a number of users had difficulty prioritizing tasks (see Finding 16). Three users exceeded 2*par on the relevant exercise. Two of those users' inefficiency on this task can be directly attributed to problems related to using the prioritization mechanism. Further support for this problem comes from comments made by those two users plus two others who did not exceed 2*par on the relevant exercise. The interface either needs to use traditional drag-n-drop rather than the substitute version, or the *Task Definition* window needs to be revised to avoid the necessity of drag-n-drop.

7.6.4 Omissions

Another class of problems resulted from features left out of the interface: some purposely and some inadvertently. Perhaps the most critical omission in the interface is the lack of an undo facility (see Finding 12). Numerous users accidentally deleted definitions and could not undo their action. Instead, they were required to reconstruct the data they had deleted. A production interface absolutely must have at least a one-level undo facility.

Users also discovered some minor omissions in the interface. They encountered two help windows that were overlooked and provided no useful information, and one that was never updated to reflect a change in the interface (see Finding 2). I also noticed[31] that, if the Space help window is visible on the monitor, it does not dynamically update when its parent window gets updated (see Finding 2). For instance, if it was showing that the cache space is at a warning level and a critical cache space event arrives, the help window does not update to show the messages related to the new space status. Finally, users encountered a few windows that did not increase the size of their widgets in proportion to an increase in their own size. These windows had been overlooked when I added the resize feature to other windows in the interface (see Finding 24).

7.6.5 Confusing Behaviors

Perhaps the most confusing behavior exhibited by the interface is the way in which it flashes the indicator lights to indicate the arrival of an event (see Finding 39). Two-color flashing, from the previous urgency color to the new urgency color, does not allow users to determine the current state of the indicator light until the flashing stops. Two possible solutions are to flash one color longer than the other (e.g., illuminate the new urgency

[31] No users reported this problem.

color for twice as long as the old urgency color) and to abandon two-color flashing altogether (e.g., flash from dark gray to the new urgency color).

Users discovered that the interface offers no feedback regarding the completion of saves or commits (see Finding 13). One of the users was so discombobulated by this behavior that he clicked the save button eight or ten times in a row every time he saved a definition. One common way to provide such feedback, progress meters, is probably inappropriate for application to this problem because the save occurs so quickly that users would be unlikely to notice the meter's brief appearance. Another way to provide the same information, in a way more appropriate to the task at hand, is using the emacs asterisk. This approach uses the asterisk to indicate that the document needs to be saved; once saved, the asterisk disappears.

Although the lack of feedback after completing a save or commit was mildly confusing, it was just the tip of an iceberg (see Finding 13). The interface follows suggestions for maintaining a "unified file model" [8]. In particular, the interface

- Automatically saves definitions when the user closes the definition window
- Does not close a window after the user saves a definition
- Does not pop up the SAVE dialog when the user closes the definition
- Allows the user to abandon changes since opening or creating the definition
- Provides an optimal, manual SAVE command

These behaviors are substantially different from the behavior of traditional interfaces. The suggestions above, however, encompass just those portions of Cooper's complete set of recommendations that seemed to apply to a simple "document" like a definition. Those recommendations not implemented include the ability to snapshot, milestone, rename, place & reposition, specify the stored format, and undo specific changes.

Cooper's recommendations appeal to me because they step back and look at the functionality required of an interface from the perspective of the user rather than from the perspective of the computer. Although his recommendations are still appealing, I now believe that I should have stuck more closely to his model. Keeping some aspects of the traditional file model while adopting some aspects of Cooper's unified file model produces an inconsistent interface that confuses users. The implementation should also include snapshop and milestone capabilities as well as a specialized undo facility. It should not offer a SAVE AS capability, and it should replace in-place naming with in-place renaming. The interface does not need the ability to place & reposition definitions or specify the stored definition format because it manipulates such a specialized and small class of documents.

7.6.6 Enhancements

Users also made numerous suggestions for future enhancements, either directly or indirectly. Some of the suggestions offer minor improvements over the current interface. Other suggestions would require a major redesign of the interface. Four of these suggestions are listed here:

- One user commented that the interface was cognizant of vertical screen real estate, but not of horizontal screen real estate (see Finding 64). This implies that the user might prefer to have the indicator lights appear horizontally across the top or bottom of the monitor rather than vertically along a side.
- More than half of the users commented that they would like to compare the amount of space necessary to hoard the immediate children of a directory as opposed to all the directory's descendants without having to resort to using a shell (see Finding 78).
- Another user suggested a tree view of the task definitions, similar to that shown in the hoard walk advice request (see Finding 77). One difficulty with implementing this view of task definitions is in dealing with the meta-information included in data definitions. Another is that task reuse implies that we do not have a proper tree structure.
- Another user suggested that the interface provide users with a visual display (perhaps a basic `cache-ls`, akin to `ls`, or a graphical tree view) of the contents of the cache (see Finding 85). Undoubtedly, such a feature would be useful for determining whether or not an individual file or directory is available. It also has a high potential for confusing users, however, because large portions of the file system space would necessarily be unavailable in the cache. If users accidentally confused this view of the file system with the full view of the file system, they might become unnecessarily concerned about the "disappearance" of uncached file and directories.

7.7 Reflections

During the course of performing this usability test, I learned much about the process. I experienced difficulty recruiting novice users, largely because I could not find users who fit my definition of novice. In modifying the interface during piloting testing, I introduced new problems and even made one of the windows less usable. I found the verbal transcripts of the sessions to be the most useful set of data. Finally, I found that predictions for the number of usability problem uncovered with increasing users were less than satisfactory, though a slight modification to one of those predictions holds promise and deserves further study. In this section, I will discuss each of these lessons.

7.7.1 Recruiting Users

Recruiting novice users for this test proved to be somewhat difficult.[32] Perhaps my expectations were unrealistic. A friend of mine had recently performed a usability test looking at an interface for remote control of an autonomous vehicle. She had had no problems recruiting users. Based upon her experience, I did not anticipate any problems either. There were, however, a few differences between her test and mine. She could include anybody with a driver's license. My novice users had to be computer science graduate students with expertise in Unix and AFS, but no knowledge of Coda. Little did I know that Coda is now covered in one of the courses required of all graduate students. Her test was highly appealing—remote control of a HMMWV! Mine was using an interface to a file system. Though I had to compromise somewhat on their background knowledge of Coda[33], I eventually found novice users.

Based upon this experience, I would have used a different procedure for recruiting novice users. Just as before, my first step would have been the same—define the "ideal novice user". I would then, however, determine the size of the population of people who fit these constraints. If the size were too small (as in my case), I would reconsider my definition, looking to see if there were an equally good population of "ideal novice users" that might lead to a larger population from which to draw participants. In my case, I could have drawn users from the ECE department (for instance). Although the graduate students in this department may have been less likely to have had Unix and AFS experience, they would also have been less likely to have knowledge of Coda. Because I did not do this, I cannot say that recruiting users would have been any easier had I done this. I do believe that redefining my ideal novice user in this way would have lead to equally valid conclusions—assuming, of course, that the users met the other constraints of my definition, namely extensive Unix and AFS experience.

7.7.2 Pilot Testing

With the exception of P1[34], the pilot users experienced conditions identical to those the novice and expert users experienced. After each pilot user completed the study, I scored the responses as I planned to do for the novice and expert users. I also addressed any problems that I observed as they were performing the test before continuing with the next pilot user. These problems included "bugs" in the Tutorial and Exercises as well as bugs in the interface.

The original design of the *Event Configuration* tab of the *Control Panel* had a single listbox that contained a listing of all events. The first pilot user suggested that this

[32] Recruiting expert users was significantly less difficult because they were already involved with the project and stood to benefit from the interface.

[33] Novice users were allowed to know that Coda was a file system and was a descendant of AFS, but not that it supported disconnected operation.

[34] Recall that P1 tested the prose of the Tutorial and Exercises. Her answers to the Exercises were not recorded.

window should show the indicator on which events would be notified. Based on this suggestion, I redesigned the window as shown in Chapter 4. As piloting continued, the problems with this window (see Finding 4) became obvious.

Another bug that I addressed was the alphabetizing of definitions. The interface originally alphabetized the "New…" definition in the middle of the list. No one had noticed this behavior before participant P2, probably because they had all used lower-case names. During pilot testing, I modified the interface such that "New…" always appeared first. Toward the end of the pilot testing, it became clear that I had introduced a bug (see Finding 31) related to these modifications.

After these two experiences, I took the attitude that piloting was intended to solve problems with the testing procedures and materials, not with the interface. Thus, a number of the usability problems that the novice and expert users experienced came as no surprise to me. I had observed similar problems with the pilot users. I did not address these problems prior to running the novice and expert users, however, for three reasons. First, rushing to address them could have introduced worse problems (as was the case with the *Event Configuration* tab). Second, I did not fully understand the severity or frequency of some problems. Third, the user study lab is a shared resource, one that my experimental setup was monopolizing. I could not address the more substantial problems without halting the test and leaving the lab unused.

Even if I had taken the attitude that pilot testing should be used to solve usability problems, however, it is not clear that I would have noticed some of the problems. Because the materials used during the test changed after almost every pilot user (sometimes substantially), performing the kind of analysis I did in Section 7.4 would have been extremely difficult. Without that analysis, I may not have noticed some of the trends. For instance, if you recall the problems with the ProvideHoardWalkAdvice exercise (as discussed in Section 7.5.2.1), the results of this exercise for the pilot users were extremely mixed. P2 made the same mistake that most of the novice and expert users made. P3 did not recognize the advice request as a request for advice and caused me to change the interface somewhat. P4 answered the question correctly. P5 also made the standard mistake. Looking back at this data, the writing was on the wall. Users were not making the connection. At the time though, it was not as clear especially because the "last" pilot answered the question correctly. I say P4 was the "last" pilot because the changes had come to the point of diminishing returns. I ran P5 only to make sure that nothing in the test was particularly confusing for expert users.

Had I taken this attitude from the very beginning of pilot testing, I could have run one or two fewer pilots. It would have saved a large amount of time and money. Each user required a time investment of at least 15 hours[35] and a financial investment of approximately $100 for lab time, media, and Coda shirt. Of course, had I run two fewer users, I may have also missed some findings.

[35] Running each user required 4-5 hours. Post-processing the raw data required approximately 3-4 hours. Transcribing the audiotapes required another 6-8 hours. Analyzing the resulting data was not done on a per-user basis so is difficult to estimate.

7.7.3 Benefit of Transcripts

While analyzing the data, it became clear to me that the transcripts provided the bulk of the evidence. This observation led me to ask two questions:

- How lop-sided was the evidence?
- What was the balance in terms of the severity of the findings?

In order to explore these questions, I analyzed the origins of the evidence for each of the 87 findings. Figure 7.32 and Figure 7.33 show this analysis visually. Figure 7.32 shows the number of findings revealed by data from particular segments of the test. It classifies those findings according to their scope and severity. Figure 7.33 shows a tree structure of the findings. The nodes of the tree, labeled T (Tutorial), E (Exercises), Q (Questionnaire), and V (Verbal Transcripts), represent the origins of evidence for the findings. The leaves of the tree each represent a single finding and are color coded to represent the severity of the finding represented. Each arc indicates that at least one piece of evidence for that finding originated from that segment of the test. These data clearly show that the transcripts provide the bulk of the evidence.

Although the Tutorial, Exercises, and Questionnaire contributed disproportionately to the more critical (level 1 and 2) findings, the transcripts doubled the number of level 1 findings identified by the test. Furthermore, they quadrupled the number of level 3 and level 4 findings! In fact, had I only looked at the transcripts and ignored all other analysis, I would have missed just a single finding (Finding 87). Although transcribing and analyzing the verbal protocol, even just the critical portions of it, was time consuming, the benefits are clear.

	Classification							
	Severity Level 1		Severity Level 2		Severity Level 3		Severity Level 4	
Segment	**Global**	**Local**	**Global**	**Local**	**Global**	**Local**	**Global**	**Local**
Tutorial	1	1	0	4	1	1	0	0
Exercises	1	4	1	2	0	2	0	1
Questionanire	0	1	2	1	3	1	0	5
Transcripts	1	10	4	6	15	24	3	23
Total	1	10	4	6	15	24	3	24

Figure 7.32: Findings Identified by Classification and Segment

This table shows the number of findings of each classification identified by each segment of the test and by the test as a whole. For example, using data from the Exercises alone, I would have identified one problem with severity level 1 that was global in its scope and four problems with severity level 1 that were local in their scope.

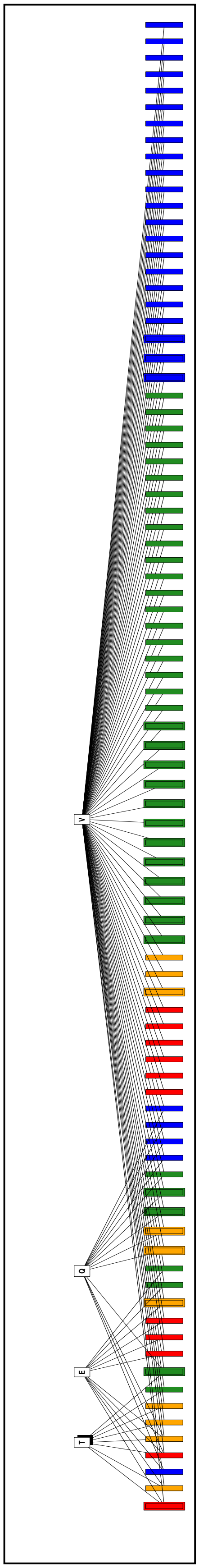


Figure 7.33: Finding Originations

This figure shows where evidence for each finding of Appendix F originated. The four top-level nodes correspond to the tutorial (T), exercises (E), questionnaire (Q), and transcripts of verbal protocol (V). Each leaf in the tree represents a single finding. The color of the leaves represents the severity rating of the given finding. Red indicates a Level 1 severity rating; yellow represents a Level 2 finding; green represents a Level 3 finding; and, blue represents a Level 4 finding. The size of the leaf represents the scope of the finding. Wide leaves correspond to global findings; narrow leaves correspond to local findings.

7.7.4 Optimal Number of Participants

Given the amount of effort needed to observe and analyze the data from each participant in a usability test, it comes as no surprise that the HCI community would like to know the optimal number of participants needed. I analyzed the evidence from each of the findings to determine which users contributed any piece of evidence, no matter how minor. This analysis is shown graphically in Figure 7.34. Each of the thirteen users is represented by a column in this matrix and each of the findings is represented by a row. Each colored box in this figure indicates that the given user contributed evidence for the given finding. More informative users appear toward the right. More frequently reported problems appear toward the top. The figure shows that the most informative user contributed to more than 50% of the findings while the least informative user contributed to fewer than 10% of the findings. The color of the box indicates the severity of the finding. This figure shows that many users found evidence for most, but not all, severe problems. In fact, six of the most severe problems were reported by just two users apiece. It also shows that most of the Level 4 problems were reported by just a few users.

Assuming that any piece of evidence would be sufficient to identify a problem, I graphed the percentage of findings uncovered as my test progressed (i.e., with my users in the order they appeared in the test). I examined the findings at each individual level, as well as the most critical problems (those with severity levels 1 or 2) and the findings as a whole. These data are shown in Figure 7.35(a-f).

An interesting characteristic of these graphs is that all but one of them is best approximated by a log-based function. The surprise is that one of them, the severity level 4 findings, is best approximated by a linear function. Evidently each additional user saw different enhancements. I hypothesize that this curve might also be logarithmic in the limit; in fact, you can see what might be a flattening trend near the end.

With just three users, I would have uncovered more than 50% of the problems at severity levels 1 and 3, but only 30% of the problems at the other levels. To uncover 80% of the problems, I would have had to run approximately six users. It is clear that beyond about six users I quickly reached the point of diminishing returns.

My next goal was to determine what, if any, effect the ordering of participants had on these results. To make this determination, I calculated the number of findings uncovered by every subset of my users. I then calculated the average number of findings reported by subsets of a given size. For instance, a set of thirteen users contains 78 subsets of size two. For each of these subsets, I calculated the number of findings they uncovered. I then averaged these values over all 78 subsets. By performing this same calculation for the other subset sizes, I could determine the average number of findings uncovered with increasing numbers of participants. On average, I find that six users were required to uncover 80% or more of the findings, but that only three users were required to uncover 50% or more of the findings.

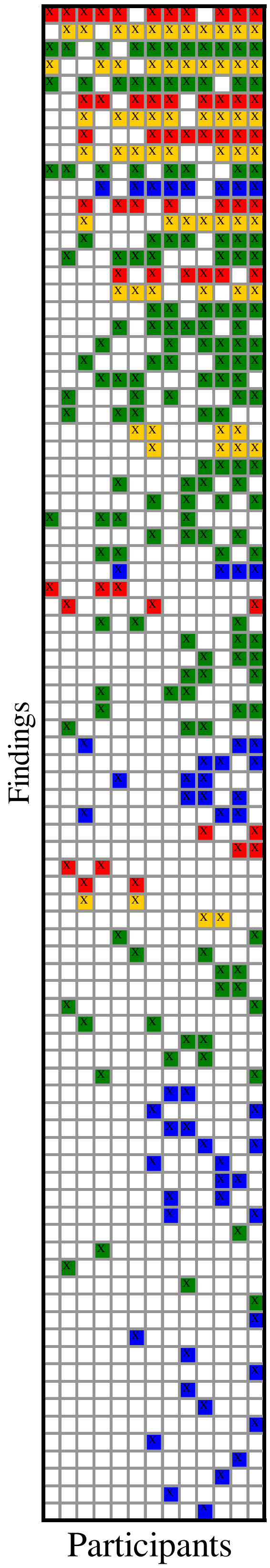
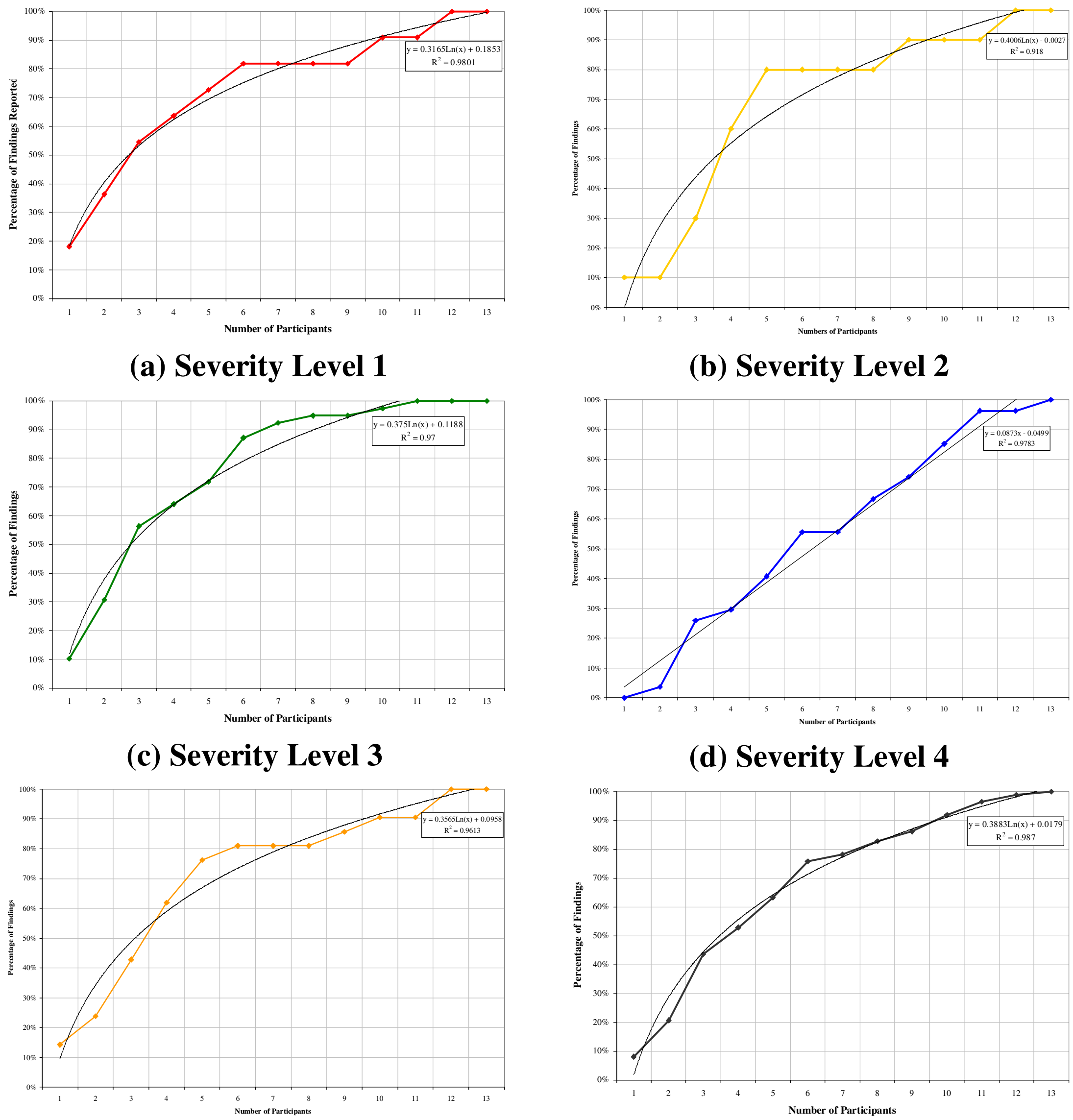


Figure 7.34: Frequency of Reporting

This matrix indicates which users contributed evidence to which findings. Each box indicates that the given user contributed evidence to the given finding. The color of the box indicates the severity of the finding: red represents level 1 problems, yellow represents level 2, green represents level 3, and blue represents level 4. The matrix has been sorted so that more productive users appear in the rightmost columns and more frequently reported problems appear towards the top. The order of users from left to right is: P2, S1, N4, P3, P5, C1, N1, E2, P4, T1, N2, E1, E3.



(a) Severity Level 1 **(b) Severity Level 2**

(c) Severity Level 3 **(d) Severity Level 4**

(e) Severity Levels 1 and 2 **(f) All Severity Levels**

Figure 7.35: Number Users Required to Cover Most Severe Findings

This figure shows the number of participants necessary to cover the usability findings. In Views (a)-(d), I show just the findings with severity levels 1-4 respectively. In View (e), I show the most critical findings, those with a severity level of 1 or 2. In View (f), I show all findings. In each view, I show the closest function that approximates the data. All but one graph is most closely approximated by a log function. View (d), however, is more closely approximated by a linear function.

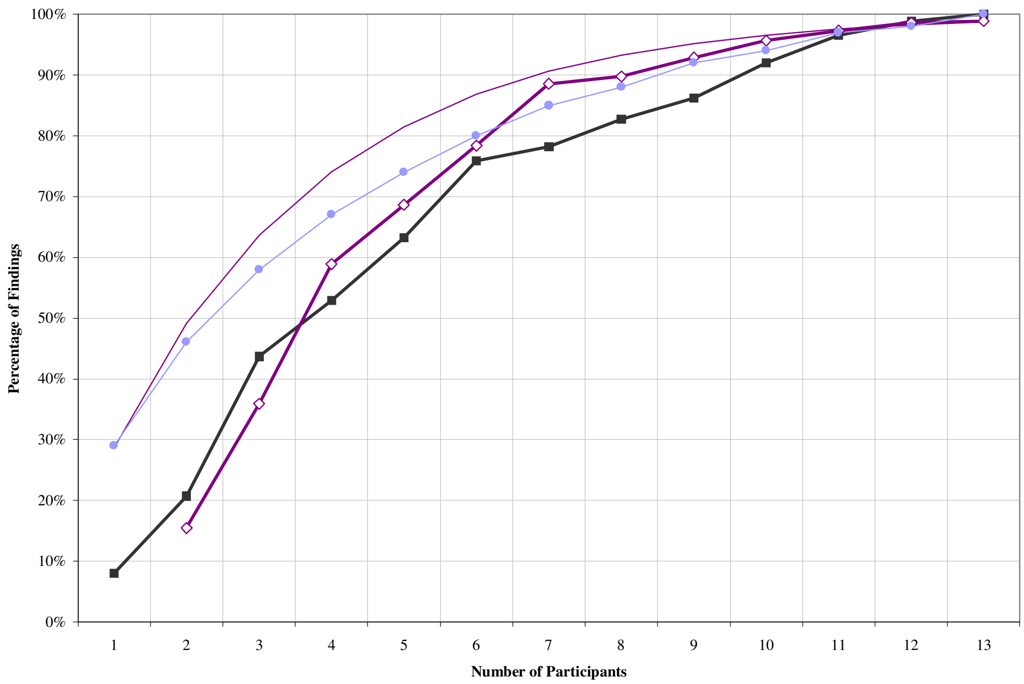


Figure 7.36: Proportion of Findings Found and Expected

This graph shows the actual proportion of findings found by the thirteen users in my test in chronological order as well as three predictions for those proportions. The line marked with filled squares represents the actual values. The curve (without markers) represents the baseline prediction using the average probability of a user uncovering a finding as calculated over all thirteen users. The line marked with open diamonds represents a modification of the baseline prediction using a running average. The line marked with filled circles represents the average number of findings uncovered by the given number of participants; it removes any effect caused by the chronological ordering of users.

The final point I wanted to explore was to examine the success of predicting the number of users required to uncover a certain percentage of findings. Virzi points out that probability theory indicates that the proportion of problems one would expect a given number of users to identify is given by the formula $1-(1-p)^n$, where p is the probability of detecting a problem (nearly 29% for this test) and n is the number of participants (13 for this test) [54]. I compared the prediction based upon this formula, which I refer to as the *baseline prediction*, with the data I collected; the results are shown in Figure 7.36.

This figure actually shows four curves. The line with filled square markers represents the actual number of findings uncovered as the test progressed (users are ordered chronologically). The curve with filled circle markers represents the average number of findings uncovered by subsets of various numbers of participants (as discussed above). The curve without markers represents the predicted values based upon the average probability of a user finding evidence supporting a finding (just under 29% for this test). The curve with open diamond markers represents a modified prediction based upon a running total average.

From these graphs, it is clear that the baseline prediction models the shape of the curve quite well, but that it overestimates the number of findings uncovered more than expected. In particular, it overestimated the expected percentage of findings by 20-30% until the number of participants reached six and by approximately 10% until the number of participants reached ten. It is also clear that the subset average predicts somewhat better, but it overestimates the number of findings uncovered by more than 20% for small numbers of participants. Neither of these models is particularly satisfying because both are very inaccurate for small values of participants, which happens to coincide with the area of the curve most interesting to experimenters trying to determine the necessity of running another experiment.

Because the baseline prediction depends heavily on the accuracy of p, I reanalyzed the data using a running average. For each point on the line with open diamond markers, I used the average probability "to date". So, to calculate the prediction for the fifth user, I averaged the probabilities collected from the first four users. The beauty of this prediction function is that it uses only data that an experimenter would have available to determine whether or not to continue running the experiment. Although I expected the running average prediction to be less accurate than the baseline prediction (because it had fewer data points to consider and was likely to be less accurate than the overall average), it actually proved to be a more accurate predictor for these data. For this reason and because it represents a more practical method for use in the field, it warrants further validation. One area of particular concern is its accuracy "on average" over multiple studies, ideally on the same studies included in [38, 54, 55].

It is interesting to note that the data from this test differ substantially from the baseline prediction, a result that apparently contradicts Virzi. Virzi claimed "In practice, this difference has been small, and neglecting it will lead to predictions within a few percentage points of those obtained" [54]. In my experience, Virzi's prediction was off by 20-30% in precisely the range of participants that would be most critical to evaluators working in the field (six or fewer users). At least part of the difference between Virzi's results and mine is simply a matter of statistics. Virzi's evaluation of the accuracy of his predictor function is based upon a Monte Carlo Procedure, examining samples of the appropriate size from the set of all users for each prediction [55]. My evaluation is based upon the chronological ordering of the users in my test. His evaluation shows us that, *on average*, the predictor function is accurate to within a few percentage points. My data show that, *in practice*, it can be quite inaccurate, particularly for small values of n. In fact, in comparing Virzi's prediction with my subset average prediction, I find that his prediction comes within approximately 7% of mine. Still, for evaluators in the field, it appears that these predictions should be used as a rough guide, and that analyzing the data "on the fly" to confirm that the test has indeed reached the point of diminishing returns continues to be the prudent course of action.

7.8 Summary

The goals of this usability test include two facets. The first was to identify areas in which the interface and the tutorial could be improved. The second was to observe a handful of people actually using the interface, looking at some specific issues regarding the ease with which these users learned the interface, their ability to understand the feedback provided by the interface, and their ability to operate the interface.

The usability test identified specific areas in which both the tutorial and interface could be improved. These findings included concerns regarding the approach, bugs in the interface, usability problems, omissions in the interface, confusing behaviors, and future enhancements. One of these findings suggests an extension to the interface that could further improve the user's ability to define tasks.

Regarding the specific issues of concern, this usability test suggests that the interface met each of its objectives. These users were able to learn how to use the interface in approximately one hour. They were able to understand problems indicated by the interface. They were able to determine what actions might resolve those problems. They were largely able to define and hoard tasks. They were able to provide advice to the system. And, finally, they were able to configure event notifications.

Perhaps the most important result of this test, however, is the fact that these novice Coda users performed almost as well as these expert Coda users. This result is extremely surprising. Although I certainly expected the interface to help novice users, I did not anticipate the extent to which it would do so. Historically, novice Coda users experience a long and arduous learning curve. A few of them have even given up before reaching the top. The interface apparently reduces this learning curve substantially, making the climb less strenuous.

Many interesting questions remain unanswered. These include questions related to the performance of the system, questions related to the effectiveness of the heuristics, as well as questions related to users' ability to operate and understand the system in the field. Although these questions are all interesting, discovering the answers to them is beyond the scope of this thesis and must be left for future work.