Appendix 1

Jacobsen, N.E., & John, B.E. (1999). A tale of two critics: Case studies in using cognitive walkthrough. Paper in review. (Submitted to *Human-Computer Interaction*)

(This page is intentionally left blank)

A Tale of Two Critics: Case Studies in Using Cognitive Walkthrough

Niels Ebbe Jacobsen[†]
Department of Psychology
University of Copenhagen
Njalsgade 88, DK2300 Copenhagen
Denmark

Bonnie E. John
Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA
USA

Abstract

Previous HCI-studies have compared usability evaluation methods quantitatively without supplementing these data with detailed qualitative data about how analysts actually learn and use methods. In contrast, we present two diary-based case studies that describe the processes of two novice analysts who learned about and applied the Cognitive Walkthrough (CW; Lewis, et al., 1990) to the specification of a multimedia authoring system. Results show that the two analysts easily learned to use CW but also that they found the technique tedious to use. Moreover, CW was neither reliable when comparing the two analysts' processes and outcomes to each other, nor accurate when comparing the analysts' problem predictions to results from usability tests applied to a running system. We examine these data in detail, searching for possible causes of the observed reliability and accuracy. Based on these analyses, we suggest three changes to CW method to improve its accuracy and two changes to improve its reliability. Further, we recommend developing a tool to reduce the tedium and integrate our suggested improvements to CW.

Keywords and phrases: *Usability evaluation methods, Cognitive Walkthrough, case studies.*

[†] Current address: Nokia Mobile Phones, Frederikskaj, DK1790 Copenhagen, Denmark, e-mail: niels@axp.psl.ku.dk

Contents

1 Introduction

In recent years there has been a growing interest in analytic usability evaluation methods (UEMs; e.g., Heuristic Evaluation, Nielsen & Molich, 1990; Cognitive Walkthrough, Lewis et al., 1990; and GOMS, Card, Moran and Newell, 1980; John & Kieras, 1996) to complement empirical methods (e.g., think-aloud usability tests or keystroke logs). Some have claimed that one advantage of analytic UEMs is that they can be applied to specifications or paper prototypes; thus, able to identify usbaility problems well before a running protoype could be implemented and empirical tests run. Several studies have compared the predictions of analytic UEMs to problems observed in usability tests (e.g., Cuomo & Bowen, 1994; Desurvire, 1994; Hammond et al., 1984; Jeffries, et al., 1991; John & Marks, 1997; Karat, et al., 1992; Nielsen & Phillips, 1993). Though the validity and reliability of some of these studies have been questioned (Gray & Salzman, 1998; Olson & Moran, 1998), the most serious problem has been that they offer only summative quantitative data (e.g., number of problems found) without supplementing these data with more detailed information about what analysts actually *do* when they are evaluating an interface. We believe the field needs process data to understand how the UEM itself leads the analyst to predict usability problems (as opposed to how the knowledge, experience, and skills of the analyst contribute to making predictions), why study results may differ, what practitioners can expect if they choose to use a specific UEM, and how UEM developers can improve their methods. Our claim is that purely quantitative studies of UEMs are akin to usability studies of a system that only provide numerical data about performance times or total number of errors; in this formative stage of UEM development, we need instead the equivalent of a series of think-aloud usability studies for assessing these methods.

To learn more about the process of using one UEM, this paper presents two diary-based case studies (Yin, 1994) that describe the processes of two novice analysts who learned about and applied the Cognitive Walkthrough (CW) technique to the specification of a multimedia authoring system. Using data from two analysts, each of whom separately conducted a CW, we compare their predictions to one another as well as to the results of usability tests on the running system. The latter enables us to assess the predictive power of the analysts' CWs (John & Marks, 1997). That is, can a CW accurately predict the kinds of problems identified in usability tests with real users? Several observations and hypotheses about the experiences and performance of the first analyst, A1, have been described elsewhere (John & Mashyna, 1997; John & Packer, 1995). This work builds on that earlier work first by studying the second analyst, A2, and then by using these new data to examine the generality of those earlier observations, test one of the hypotheses, and develop further hypotheses.

The next section of this paper includes background information about the CW method and the results of the first case study. We then describe the case-study situation, the usability-test conditions, and the database used to track information. Section 4 presents the quantitative results of the CW analyses and usability tests. Section 5 queries the database to answer many questions about how and why the two analysts differed and what impact different stages of the CW had on the outcome of the evaluation. Section 6 investigates why the results of the CW analyses diverged from usability test results. Section 7 concludes with hypotheses about the use of CW and a call for future work.

2 Background

Cognitive Walkthrough (CW) is a usability inspection method based on the psychological theory called CE+, a theory about what affects the ease with which a new interface can be learned (Polson & Lewis, 1990). Similar in rationale and execution to requirements or code walkthrough, the technique has been evolving since its introduction in 1990 (Lewis et al., 1990). The most recent description of CW (Lewis & Wharton, 1997) appeared in the *Handbook of Human-Computer Interaction* (Helander, Landauer & Prabhu, 1997). Our study examines the latest CW version available when our data were collected (Wharton et al., 1994).

CW consists of two phases: preparation and execution. In the preparation phase the analyst selects tasks to be analyzed and specifies the knowledge, experience, and skills a user can be expected to bring to the task. For each task the analyst specifies a correct action and the expected system feedback after each action. In the execution phase the analyst closely examines each action in an action sequence and asks the following four questions:

1. Will the user try to achieve the right effect?
2. Will the user notice that the correct action is available?
3. Will the user associate the correct action with the effect trying to be achieved?
4. If the correct action is performed, will the user see that progress is being made toward solution of the task? (Wharton et al., 1994, p. 112)

If, for any question at a given action, the evaluator can answer "yes", this answer should be substantiated by a *credible success story* for that question. When all four questions for a given action have been answered by credible success stories the user is expected to have no difficulty completing the action. If, for any question at a given action, the evaluator can answer "no", this answer should be substantiated by a *credible failure story* for that question indicating that a usability problem has been identified. In summary, for each question about a given action, the analyst should either identify a success or a failure story. In all cases, the story should be credible (obviously, just answering the question with "yes" or "no" is not sufficient).

The CW literature makes several other recommendations. During the execution phase, the analyst should record any additional assumptions about user knowledge requirements, e.g. information about what the user must know prior to performing the task and what the user should learn while performing the task. Side issues (such as detection of spelling errors in menus) as well as design changes are also recorded in the execution phase. An individual analyst or a group can perform the CW evaluation. Although a CW evaluation can be completed on a running system, it is especially advocated as being cost-efficient when applied to a description of a system, e.g., user-interface specification or storyboards.

John & Packer (1995) described an analyst's (A1's) process of learning and using the CW technique applied to a specification for a multimedia authoring system. That analyst experienced the CW technique as both learnable and usable for a computer designer with little psychology or HCI training. Other more specific findings in the case study were that the technique by itself does not help in selecting and setting up task scenarios. This is also known to be the case with other UEMs such as usability tests, GOMS, and Claims Analyses. As is problematic with other inspection methods, the CW technique by itself was not seen to give any guidance about frequency or severity of detected usability problems. Finally the case study showed that the actual walkthrough of the action sequences was problematic as questions 1 and 3 were hard to use correctly.

John & Mashyna (1997) used John & Packer's CW data and compared them to the outcome of usability tests of a running version of the system built from the specification. Four users participated in think-aloud usability tests with each session lasting 60 to 90 minutes. John & Mashyna found that of the 37 observed problems that could have been predicted by A1, only five percent of these were predicted precisely, and another five percent were predicted vaguely. Consequently the analyst missed at least 90 percent of the observed problems. Moreover, 27 percent of the problems detected by A1 were found to be false alarms. The lessons learned from this case were that (1) while the concepts and procedure of CW were learnable in the abstract, actually applying the technique was more difficult, (2) CW is sufficiently flexible to apply to a complex system like a multimedia authoring tool, (3) A1's use of CW was not particularly effective in predicting problems encountered in usability tests, and (4) improvements in the CW method may come from experts' tacit knowledge.

In addition to the quantitative results and possible explanations for those results, the case of A1 provided several hypotheses about learning and using CW more effectively. These included a hypothesis about creating "macros" for answering the CW questions for repetitive tasks to decrease evaluation time; a hypothesis that evaluators should set up and analyze error-recovery tasks as well as the correct action sequence for every task; and a hypothesis that people wanting to learn CW read only *The Cognitive Walkthrough Method: A Practitioner's Guide* (Wharton et al., 1994) as the earlier, more theoretical papers on CW were confusing to A1. In our second case, we follow this last suggestion to examine whether a second analyst can perform better and with less confusion and doubt after reading the *Practitioner's Guide*.

3 The Case Study Situation

3.1 The Cognitive Walkthrough Analyses

Many software development companies do not have dedicated professional usability evaluation staff (Dillon et al., 1993). A common practice for developers evaluating the usability of their system is to select an assessment technique, identifying and locating literature that describes its use, learning the technique from these materials, and applying the technique to the system design. In this study, we set up two similar situations.

3.1.1 The analysts' choice of HCI assessment technique

Five volunteer analysts were given a 30-minute lecture on HCI assessment techniques (by the second author). This lecture contained the introductory UEM tutorial material given at the ACM CHI conferences by the second author every year since 1992 (Butler, Jacob & John, 1992-1999). One week after this lecture, each analyst was asked to choose one UEM they wished to apply to a multimedia authoring system. Of the five analysts, A1 chose CW as his preferred technique. After A1 had read and applied CW to an interface he wrote a report where he strongly recommended future CW analysts read only one paper, namely *The Cognitive Walkthrough Method: A Practitioner's Guide* (Wharton et al., 1994). Based on this hypothesis a new volunteer analyst, A2, was later asked to do a cognitive walkthrough on the same interface as the one A1 evaluated, but using Wharton et al. (1994) as his only source to learn the walkthrough technique. Thus, this second case study is testing whether A2 is able to perform better and with less confusing with only Wharton et al. (1994) as his teaching material.

3.1.2 The system

The CW analysts evaluated a portion of a multimedia authoring system that was built in The Advanced Computing for Science Education (ACSE) project to teach the skills of scientific reasoning (Pane & Miller, 1993). The system, called the *Builder*, makes it possible for a user to create a running multi-media *volume* that can be used by students to learn certain parts of a science domain. The Builder resembles an advanced word processor insofar that the user can create volumes consisting of plain text, still graphics, movies, animations, and simulation code.

The two CW analysts were given two paper documents with which to complete their analyses: a user interface specification document on the Builder (Gallagher & Meter, 1993), and an example multi-media document (a volume printed on paper). The specification document was a detailed description of the user interface of the Builder. The specification document (consisting of 44 pages) was divided into three parts: an introduction, a section on how end-users should navigate and use a volume, and an explanation about how a teacher should create and modify volumes in the Builder. The specification included 37 figures of screen items, which ranged from small pictures of specialized cursor icons, to tool palettes, to full-page figures of the entire screen as seen in Figure 1.

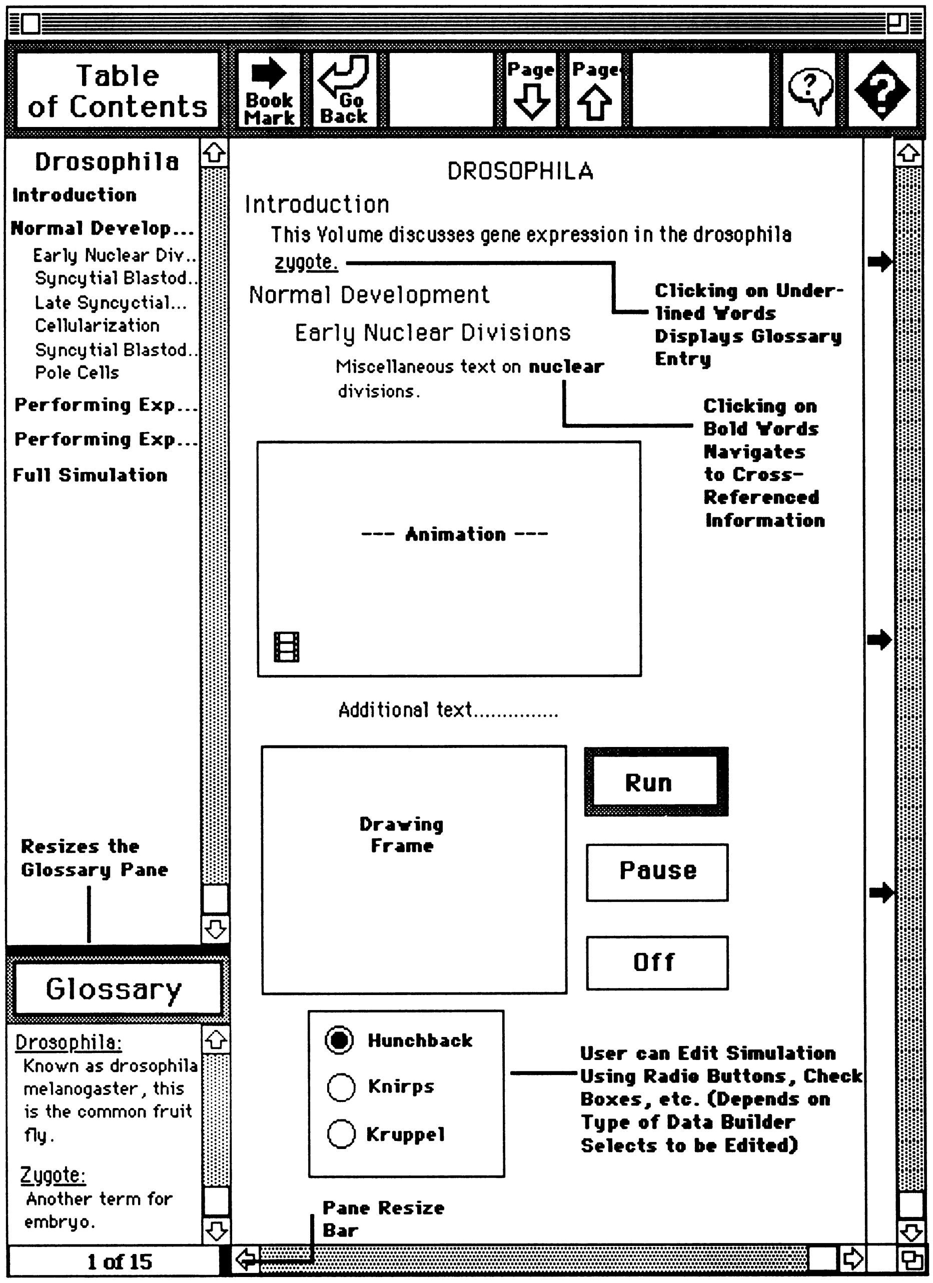


Table of Contents
Book Mark
Go Back
Page
Page
Drosophila
Introduction
Normal Develop...
Early Nuclear Div..
Syncytial Blastod..
Late Syncyctial...
Cellularization
Syncytial Blastod..
Pole Cells
Performing Exp...
Performing Exp...
Full Simulation
Resizes the Glossary Pane
Glossary
Drosophila:
Known as drosophila melanogaster, this is the common fruit fly.
Zygote:
Another term for embryo.
1 of 15
DROSOPHILA
Introduction
This Volume discusses gene expression in the drosophila zygote.
Normal Development
Early Nuclear Divisions
Miscellaneous text on nuclear divisions.
Clicking on Underlined Words Displays Glossary Entry
Clicking on Bold Words Navigates to Cross-Referenced Information
--- Animation ---
Additional text...............
Drawing Frame
Run
Pause
Off
Hunchback
Knirps
Kruppel
User can Edit Simulation Using Radio Buttons, Check Boxes, etc. (Depends on Type of Data Builder Selects to be Edited)
Pane Resize Bar

Figure 1. An example of a full-page figure of the Builder interface taken from the specification document (Gallagher & Meter, 1993, included here with permission of the authors). The Table of Contents pane and the Glossary pane reside on the left side of the Builder interface. The tool palette resides on the top of the interface. The arrows on the right side of the interface represent bookmarks. The remaining canvas is dedicated to the current volume that might contain text frames, animations, drawing frames simulations, etc.

The example multi-media document was a printed version of a volume made in the Builder. This 55-page volume included 23 high-resolution images and figures, 3 movies, 7 simulations, 10 fragments of simulation code, and 10 review questions. The multi-media example document was produced with an earlier version of the Builder that did not include certain features described in the specification document: a table of contents, a glossary, and hyperlinks. However, prior to the CW analyses the second author of this paper modified the example multi-media document to include these features. The table of contents and hyperlinks were identified in two lists included at the end of the printed volume. Figure 2 shows an example of a page in the example multi-media document, a document that was entitled "*Gradients, Gene Expression, and Pattern Formation: The Early Development of Drosophila Melongaster*".

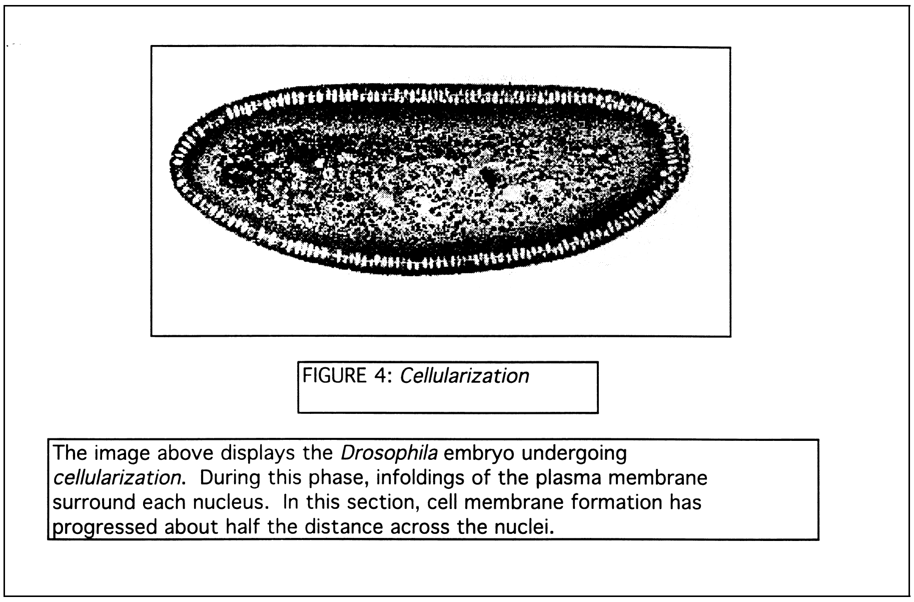


FIGURE 4: *Cellularization*

The image above displays the *Drosophila* embryo undergoing *cellularization.* During this phase, infoldings of the plasma membrane surround each nucleus. In this section, cell membrane formation has progressed about half the distance across the nuclei.

Figure 2. An example of a page in the example multi-media document. This page shows a picture frame, a frame associated to the picture (a caption), and another frame explaining the figure above.

3.1.3 The analysts

Analyst A1, a researcher in Carnegie Mellon University's School of Computer Science, had taken over a dozen courses in computer science. He considered himself fluent in two programming languages and had worked professionally as a programmer before taking part in this case study. He had taken one cognitive psychology course, but none in HCI. A1 received graduate-course credit for participating in this study. A2 was a full-time computer facilities manager for a 100-person department and a part-time student in Carnegie Mellon University's Masters of Software Engineering program. A2 did not have any formal experience in cognitive psychology, but had taken one course in HCI. A2 also received graduate-course credit for participating in this study.

3.1.4 Procedure

The analysts worked primarily on their own: A1 for an elapsed time of 10 weeks and A2 for an elapsed time of 6 weeks. They used two forms to record their work on an ongoing basis: A structured diary and a problem description form (described below). Moreover, the analysts recorded their actual evaluation in terms of task scenarios, correct action sequences, and success/failure stories in the walkthrough. Each analyst produced both a verbal and a written report of their analysis. A questionnaire assessed the analysts' educational and professional background. All of these data contributed to the case reported in this paper.

A *diary form,* adapted from Rieman (1993), allowed the analysts to record their activity for half-hour intervals through the evaluation process. For each entry, each analyst provided a short text description of the learning activity and categorized the activity into one of the following:

1. Literature search
2. Reading for "what it is"
3. Reading for "how to do it"
4. Reading/Analysis (when reading and analysis are so intertwined as to be inseparable)
5. Analysis (when the analysts know the technique well enough to analyze without references to the literature)
6. Unrelated

The diary form also included columns where different categories of information could be recorded: difficulties using the technique, insights into the technique, usability problems in the system, solutions to usability problems, and the catch-all category "other". At any time, the analyst could write a note on the form and write an extended explanation in their own words.

The *problem description report* (PDR), adapted from Jeffries et al. (1991), provided an area for describing the detected usability problem, an estimate of the severity and frequency of the problem, and an assessment of whether these judgments came from the technique itself, as a side-effect of the technique, or from some form of personal judgment. Each PDR had a reference number that also appeared in a column of the diary so that every PDR could be related to the context in which it was detected.

A1 met with the second author and four other analysts using other UEMs once a week, in a seminar setting, to discuss the process of the evaluation (as opposed to discussing the content of the analyses). That is, they discussed issues such as problems obtaining or understanding articles and problems making the techniques applicable to the Builder, but they did not discuss specific usability problems in the system. Since A2 completed his CW later, when no seminar was in session, he met once a week with the second author alone to discuss the process of the evaluation. As with A1, they discussed only the process of doing the CW, not the contents of PDRs.

Each analyst produced a written report that included a brief summary of the technique, an annotated bibliography of the articles used to learn and apply the technique, and a description of modifications made to the technique in the process of applying it to the interface. Also, areas of exceptional doubts or confidence about using the technique, suggestions for improving the technique, and the three most important problems to be fixed in the Builder were included in A1 and A2's written reports.

3.2 The Usability Tests

We conducted a series of think-aloud usability tests to see if the problems predicted by the CW analysts would show up in tests with real users. Note that we are *not* comparing the CW technique to the usability testing technique; these techniques differ greatly in the development phase during which they can be used and the resources necessary to use them. Our goal in running usability tests was to assess what John & Marks (1997) called the *predictive power* of analytic UEMs. That is, if the problems predicted from a specification weren't fixed, would they be confirmed or refuted in usability tests of the system built to that specification?

The first test, examining four users completing four tasks, was reported in John & Mashyna (1997). After comparing these user tests to A1's CW, we discovered that the tasks the users were asked to complete did not match the tasks set up by the CW analysts to a satisfying degree. To fix this oversight, we modified the assigned usability test tasks and ran four more users (see Table 1). All eight usability test sessions were videotaped for later analysis.

Task	Usability test no. 1 (4 users)	Usability test no. 2 (4 users)
Create a three-page document including text fragments, a picture, and a simulation, and add entries into the glossary pane and the table of contents pane. Mark page two so that the user of the volume can jump straight to it from anywhere in the volume using the bookmark function.	x	x
Modify document by switching pages two and three.	x	x
Delete the second page in the document.	x	
Add another page, enter a new glossary item and modify the definition of an existing entry.	x	x
Add another page with a code fragment, and make sure that students who use the volume will not be able to edit this piece of code (using the lock function).		x
Save volume in two versions: an editable version for the professor and a not-editable version for the student.		x

Table 1. Breakdown of the tasks presented to the users in the two usability tests.

In order to increase the reliability of the usability test results (Jacobsen et al., 1998) each of five evaluators analyzed a subset of the eight tapes, with each evaluator analyzing at least four tapes. Two of the evaluators were very experienced usability test evaluators, having analyzed more than 50 usability test sessions previously. The three other evaluators had less experience, having analyzed between 2 and 6 usability test sessions previously.

Nine predefined problem criteria were used by usability test evaluators to label problems encountered by the users:

1. the user articulates a goal and cannot succeed in attaining it within three minutes,
2. the user explicitly gives up,
3. the user articulates a goal and has to try three or more actions to find a solution,
4. the user produces a result different from the task given,
5. the user expresses surprise,
6. the user expresses some negative affect or says something is a problem,
7. the user makes a design suggestion,
8. the system crashes, or
9. the evaluator generalizes a group of previously detected problems into a new problem.

The evaluators were asked to report on three properties for each problem detected: (a) a free-form problem description, (b) evidence for this problem consisting of the user's action sequence and/or verbal utterances, and (c) the criteria for identifying the problem.

Based on the individual evaluators' problem lists two independent investigators matched all problems with the aim of constructing a master list of unique problems identified by the usability test. They agreed on 81% of the unique problems; disagreements were resolved through discussion and a consensus was reached (see Jacobsen et al., 1998 for further discussion of this analysis).

3.3 The Case Study Database

Based on recommendations from Yin (1994), this case study was designed to provide several types of data with which to explore the process of learning and doing a CW analysis. Materials include (1) structured diary entries, (2) free-form diary notes, (3) Problem Description Reports (PDRs), (4) the analysts' CW materials, and (5) the analysts' final reports. All of this information can be located along a timeline of activity in which the analysts engaged. Hence, it is possible to see causal links between different types of data, e.g., a link between a diary entry and a paper that an analyst read. On the usability testing side, we have usability problem reports, produced by eight users and five evaluators, against which to match to the CW analysts' PDRs.

Yin (1994) suggests that "every case study project should strive to develop a formal, presentable database, so that, in principle, other investigators can review the evidence directly and not be limited to the written reports" (p. 95). To this end, the information has been compiled and can be accessed through the relational database described in Appendix A. This database contains 21 tables with 23 relational connections between tables (see Table 8 page 149). The database contains more than 3,500 records consisting of more than 30,000 data entries. All results presented and discussed in this paper have been extracted by querying this database.

Designing and filling in data in the database has been a protracted process due to the use of an iterative systems development method and because case study hypotheses sometimes are generated while analyzing data material rather than before conducting the study. We think that other researchers interested in conducting case studies on UEMs can profit from our work and experiences, and we therefore invite researchers to use the design and contents of our database.

4 Results

In this section we describe quantitative results from the case study. First we examine the process and the outcome of the analysts' work, i.e. the time spent on different activities during the evaluation and the total number of problems detected. Second, we identify the number of problems detected in the usability tests and discuss 1) which of these problems could have been predicted by the CW analysts, 2) which were predicted correctly, and 3) which were overlooked in the CW evaluation. Based on these quantitative results the discussion sections (5 and 6) reveal qualitative, in-depth analysis of how the analysts worked, why they worked as they did, and the impact of their process in terms of the actual outcomes of their evaluation.

4.1 The process and outcome of the CW analysts' work

A1 spent a total of 45 hours on his evaluation. After starting his evaluation by searching literature and reading about a number of different evaluation techniques, he then read about the CW technique in depth. The total time spent searching the literature and reading accounted for 23 hours of his evaluation time, while he analyzed the interface with the CW technique for 22 hours. A2, who had only been asked to read Wharton et al. (1994), spent no time on a literature search and hence spent less time reading. A2 spent 32 hours on the evaluation; of these 32 hours he spent 25 hours analyzing the interface with the CW technique (see Table 2 and Figure 3 for the analysts' activities).

	A1	A2
Literature search and reading	23	7
Analysis and analysis/reading interleaved	22	25
Total hours spent on the evaluation	45	32

Table 2. Hours spent on different activities for the two analysts.

Throughout the evaluation phase the analysts recorded insight and difficulty notes in their paper diary. While A1 recorded 126 notes, A2 only recorded 42 notes (see Table 3). Compared to A2, A1 wrote many more notes while searching and reading literature partly because he found and read many more papers than A2. However, during the time spent searching and reading literature, A1 still wrote more than twice as many notes as A2. Both analysts wrote comparable number of notes on the actual evaluation process.

	Number of notes written while searching literature and reading for "how-it-is" and "how-to-do"	Number of notes written while analyzing and analyzing/reading interleaved	Total notes written
A1	84 (3.7 notes/hour)	42 (1.9 notes/hour)	126
A2	12 (1.7 notes/hour)	30 (1.2 notes/hour)	42

Table 3. Number of notes written by the two analysts distributed into two groups: literature search and reading, and analyzing and analyzing/reading interleaved.

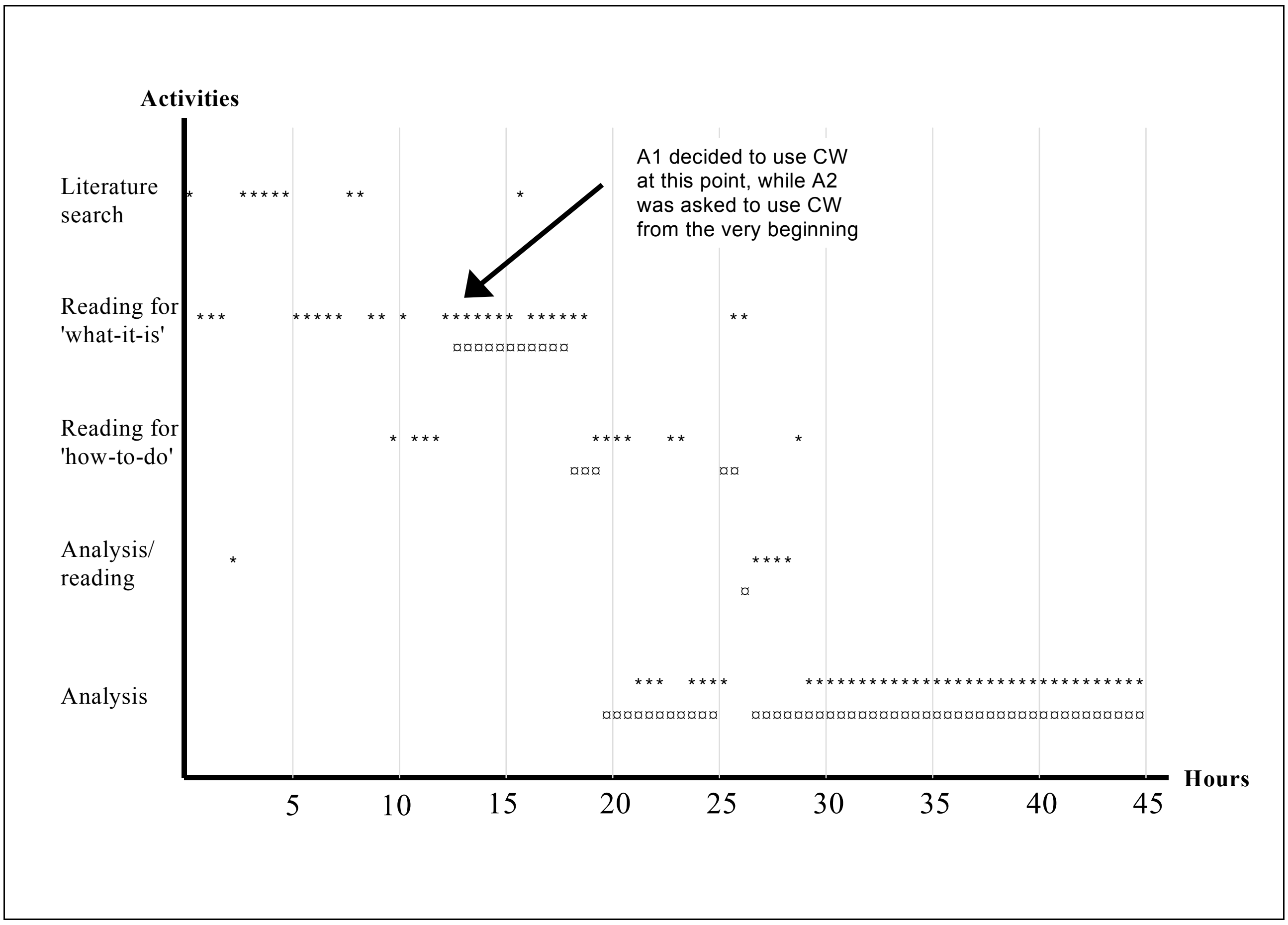


Figure 3. Through the semi-structured diary form recorded in the case study database it was possible to track the analysts' activities throughout their evaluation of the Builder. Each "*" represents one half-hour time slot for A1, while each "¤" represents one half-hour time slot for A2. A2's first time slot is adjusted to the point in A1's evaluation where he decided to use CW. Hence, A2 spent less time than A1 overall but their actual analysis times (excluding literature search and reading) were comparable.

The outcome of the CW evaluations was a number of PDRs identifying the problems that had been detected in the interface. A1 recorded 42 PDRs, while A2 recorded 9 PDRs. Since some PDRs covered more than one unique problem and others were duplicates, A1 detected 46 unique problems in total while A2 detected 7 unique problems. Some problems were detected in the preparation phase; other problems were detected in the actual walkthrough. Twenty-six of A1's 46 problems and 4 of A2's 7 problems were detected while walking through action sequences, i.e. they could be tracked back to a specific failure story for a specific action in an action sequence. A1's remaining 20 problems were detected while analyzing a test task scenario with no recordings of a walkthrough (8 problems), reading the specification document (6 problems), determining an action sequence (2 problems), filling out other PDRs (1 problem), and doing other activities not related to the actual walkthrough (3 problems). Two of A2's remaining 3 PDRs were reported while reviewing his action sequences and the last problem was reported while writing success or failure stories, though this specific PDR was not related to a specific question/action in an action sequence.

Only three problems were detected by both analysts. A1 found all three problems while walking through his action sequences, but he only credited the CW technique with helping him identify one of these problems; the other two problems were judged by A1 as being a side-effect of using the CW technique. A2 found only one problem while walking through the action sequences and he also credited this problem detection to the CW technique. The other two problems A2 identified in common with A1were found while setting up action sequences, though he gave the credit for these identifications to the reading of the specification document.

The analysts were requested to report the three most important problems to be fixed in the Builder; none of these were identical. A1 reported two general fixes, one regarded use of unambiguous labels rather than icons in the tool palette, and the other regarded reorganization of the menu structure. A1 also reported a more specific problem as glossary and table of contents panes could not be resized in all directions. A2 reported three concrete problems of which two were inconsistencies in the specification document. One inconsistency regarded frame labeling (which was not problematic in the Builder, only in the specification document), and another inconsistency regarded seemingly different ways of creating a new volume. The last important fix regarded insertion of pages, which could only be done after the user had selected a frame.

4.2 CW predictions and usability test observations

A total of 103 unique problems were observed in the usability tests, but not all of these problems could have been predicted by the CW analysts. Similarly, not all of the problems detected by the CW analysts could have been observed in the usability test. (In the following discussion, problems detected in the CW will be denoted *predicted* problems, while problems detected using the usability tests will be denoted *observed* problems). First we will analyze which of the problems predicted by the CW analysts might have been observed in the usability tests. Then we will analyze which of the observed usability test problems the CW analyst might have predicted. Finally we describe the predictive power of the CW analyses.

4.2.1 Problems predicted by CW that might have been observed

The cognitive walkthrough analyses were completed at an early stage of the development process when there existed only a specification document and no running system. Also, as the development project was a real-life process, certain functions were implemented differently than described in the specification document, often due to new and improved design decisions discovered in the implementation phase. Hence, some of the problems predicted could not have been observed as (1) those functions had not yet been implemented, (2) related functions were changed from the specification, or (3) the analysts misread the specification and therefore reported problems that actually could not occur. The remaining problems detected by the two CW analysts might have been observed in the usability tests (those problems that were identified with features implemented as described). Table 4 shows the problems predicted by the CW analysts that could or could not be observed in the usability tests.

	Reasons	A1	A2
Could not be observed	Not yet implemented in running version	18	3
	Changed in running version	13	2
	Analyst misread the specification	1	1
Could be observed	Existed in running version	14	1
Total		46	7

Table 4. The number of CW-predicted problems that could or could not be observed in the usability tests.

A1 detected 14 problems and A2 detected 1 problem that had the possibility of being predicted in the usability test. Before revealing how many of these predicted problems were actually observed, we will analyze how many of the observed problems might have been predicted by the CW analysts.

4.2.2 Problems observed in usability tests that might have been predicted

Some of the observed problems in the usability test could not be predicted because of factors we could not control (see Table 5). *Bugs and artifacts in the running system* such as slow scrolling, slow saving, and system crashes were considered to be observed problems that could not have been predicted. Similarly, observed problems that had to do with aspects of features *changed from the specification* could not have been predicted. Also, some observed problems were caused by the *user's inattention to detail* in doing the tasks they had been asked to solve, i.e. the user's end product differed from the target volume in the way the text was word-wrapped and the user made no attempt at all to try to get them to match. The CW analysts could not have predicted a user's inattention to detail.

Any written specification is typically incomplete or ambiguous. Hence, usability evaluation based on a written specification is limited by its quality. Though we regard the specification document used in this case study to be of high quality, it is not surprising that some problems observed were quite difficult to predict, often due to an incomplete or ambiguous specification. We grouped these problems into one of three categories: effect, limitation, and functionality. *Effect* problems were problems where the effect of some function was not described or not fully described in the specification (e.g., when deleting a page, the glossary entries from that page are not deleted automatically). *Limitation* problems referred to those problems where some sort of undocumented limitation in the system led to a user problem (e.g., there was an artificial limit of only having one Table of Contents item per page). And *functionality* problems were those where a certain undocumented function (e.g., auto-save function) led to user problems.

Problems observed in the usability tests relating to functions that were *explicitly described in the specification* should have been predictable with a CW analysis. Other types of problems observed in the usability test could also have been predicted. For example, if a user requested a function that was *neither described in the specification nor implemented* in the running system then it could have been predicted by the CW analysts. For example, some users mentioned that the system had no text search capability – this function was not described in the written specification nor was it implemented in the running system. However, the CW analysts could have set up a scenario where they tested for finding a piece of text in a volume and might then have predicted that a search capability was necessary.

Table 5 categorizes all observed problems into two problem types: those that could not have been predicted, and those that could actually have been predicted.

		Reason for belonging to group	Number of problems	Total for the group
Could not have been predicted	Could not be controlled	*Artifacts of the system*	4	
		Bugs in running system	3	
		Changed from the specification	23	
		User's inattention to detail	7	37
	Incomplete specification	*Effect*	5	
		Limitation	2	
		Functionality	7	14
Could have been predicted		*Explicitly described in specification*	33	
		Neither described in spec. nor implemented	19	52
Total problems observed				103

Table 5. The number of observed problems that could not have been predicted and that could have been predicted. Each of these categories contains specific reasons for why a problem could have been or could not have been predicted.

4.2.3 Summary of the match between predicted and observed problems

From Table 4 and Table 5 we know that A1 and A2 predicted 14 and 1 usability problems respectively that had the potential of being observed in the usability tests. We also know that a total of 52 of the usability problems observed in the usability tests had the potential of being predicted by the CW analysts. We studied the CW problems that might have been observed in order to match them with observed problems. The first author initially matched problems by annotating each CW problem with a reason why this problem could or could not be observed. Then both authors went through each problem and, if the second author disagreed about the first author's initial judgment, came to consensus.

Only three problems observed in the usability tests were precisely predicted by A1 (see Table 6). Three other problems predicted by A1 were related to observed problems but they can not be considered to be precise predictions[1]. A2 did not predict any problems observed in the usability tests.

Eight usability problems were predicted by A1 but were not observed in the usability tests; one such problem was predicted by A2. Three of these problems were *false alarms*. We define a false alarm as a problem predicted by a CW analyst but which, when confronted by the exact situation in the running system, causes users no identifiable problems in performing their task.[2] Having subtracted these false alarms from the total number, six problems predicted by A1 remain. These were neither confirmed as observed problems nor rejected as false alarms, as the users did not use the functions related to these problems.

[1] The precisely and vaguely predicted problems differed in the sense that precisely predicted problems pointed to the same item in the interface with an equivalent description of the problem. Vaguely predicted problems pointed to the same item in the interface but were not precisely defined with respect to what constituted the problem. For example, the CW predicted problem "User will be confused about what actions to find in the button palette and what in the (augmented) standard MAC menu" was matched vaguely to the usability test problem description "More of the important commands should appear as toolbar buttons". In comparison the CW predicted problem "Creation of frames: After choosing the location (V.V. Spec p.21, first paragraph) for a newly created frame (by clicking the mouse), the mouse button has to be kept pressed to be able to size it. From Mac experience one would expect that there are 2 separate actions: 1. place frame -> pops up in standard size with handle bars. 2. resize it by dragging on the handle bars" was matched precisely to the usability problem description "User does not know that after clicking the frame button she has to click mouse button on the Volume canvas and hold it down while moving to create a frame".

[2] The concept of a false alarm is a controversial issue in HCI. We will discuss it in more detail in section 6.2.

Of the 52 problems observed in the usability tests, A1 missed 45 (87%). All 52 observed problems were missed by A2 (100%). The analysts predicted none of the observed problems concerning lack of functions (as indicated by a user's search for a function or capacity). Table 6 shows the correspondence between the observed and the predicted problems as well as the misses[3].

	A1	A2
Observed problems that were predicted precisely	3 (6%)	0 (0%)
Observed problems that were predicted vaguely	3 (6%)	0 (0%)
False alarms (predicted problems that could have been but were not observed)	2	1
Predicted problems related to functions that were not tested in the usability tests	6	0
Misses (Observed problems that could have been but were not observed)	45 (87%)	52 (100%)

Table 6. The results of matching problems predicted by the analysts and the problems observed in the usability tests. A1 predicted six problems that were observed either precisely or vaguely. He predicted two problems that were false alarms, and he missed 45 observed problems. A2 did not predict any problems that were observed, he predicted one problem that turned out to be a false alarm, and he missed 52 problems.

5 Differences between the two analysts while learning and using the CW technique

Studies have shown that other usability evaluation methods like heuristic evaluation and think-aloud studies suffer from a substantial evaluator effect (Jacobsen et al., 1998; Molich & Nielsen, 1990). One might therefore presume that there would be a similar evaluator effect among evaluators using the CW technique. In this case, however, three factors might argue against such a presumption. First the analysts in this case had similar backgrounds, they were both highly motivated, and they followed the same procedure in applying the CW technique. Second, CW is more structured than heuristic evaluation and also more structured than the process of analyzing think-aloud sessions. Third, CW is not known to have an evaluator effect (see for example Lewis et al., 1990). Considering these arguments, we believe that the differences between the analysts in the total number of problems each detected and the small number of problems identified by both are quite surprising.

How and why were the CW analysts so different in their detection of problems? In this section we will elaborate on the analysts' work. We will reveal *how* the analysts differed, we will try to explain *why* they differed, and we will estimate the *impact* of these differences on the problems predicted. We expect these analyses will generate hypotheses on how the cognitive walkthrough can be improved to meet the demands of both utility and usability. In section 6 we will return to the usability test results to explain how the CW analyses differed from those results.

5.1 The cognitive walkthrough phases

As mentioned in section 2, CW consists of two chronological phases, preparation and execution. Their process included 6 stages—4 during preparation and 1 during execution. As seen in Figure 4 they first read about the technique and the system (the two upper ovals), then they defined the user and chose task scenarios (the oval second from the top). The last step in the preparation phase was

[3] Logically the number of correctly predicted problems (3 vaguely and 3 precisely predicted for A1; 0 for A2) and the number of misses (45 for A1 and 52 for A2) should add up to 52 problems. However, for A1 one of the predicted problems vaguely matched two observed problems. Hence, A1's total number of misses and matches consist of one less than would be expected.

to transform the task scenarios into action sequences (the oval third from the top). In the execution phase they had to walk through the action sequences and record problems (bottom circle).

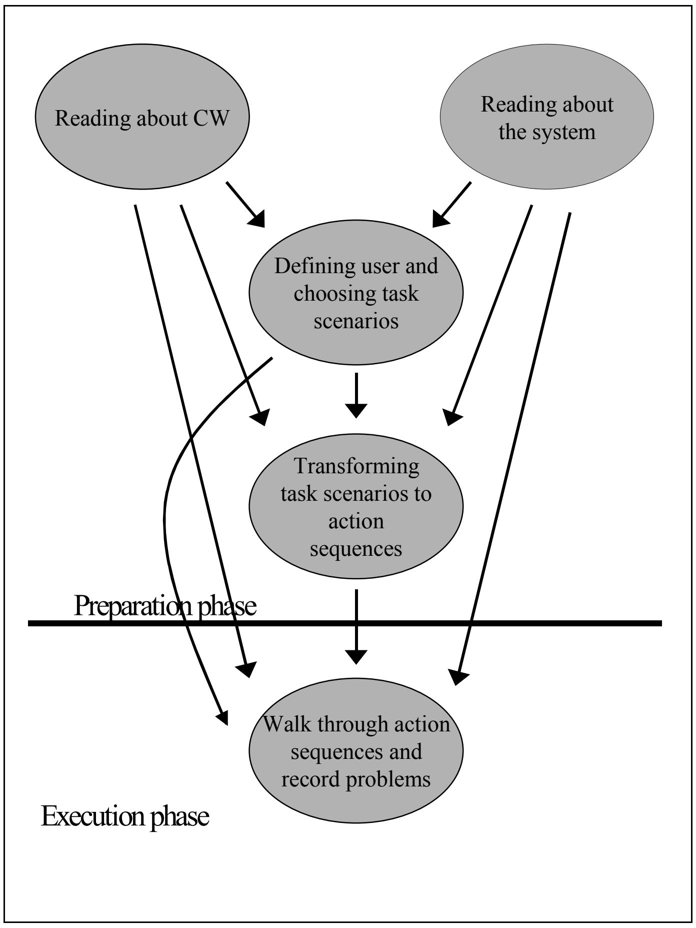

Reading about CW
Reading about the system
Defining user and choosing task scenarios
Transforming task scenarios to action sequences
Preparation phase
Walk through action sequences and record problems
Execution phase

Figure 4. The stages in a cognitive walkthrough and the dependencies between stages. The quality of the detection and recording of PDRs are directly and indirectly dependent on all stages in the evaluation.

The striking difference in the number of PDRs recorded by the two analysts might originate in any phase in the their evaluation process. For instance, poor task choice may lead to low coverage of the interface, with few detected problems as the most likely outcome. Alternatively, differences in translating the same task scenarios into action sequences might lead to different PDRs. We have organized sections 5.2 through 5.5 according to the stages in CW in order to trace the specific impact of each phase on the outcomes of the CW analyses.

5.2 The reading process

5.2.1 What did the analysts read?

Reading the specification document and about the CW technique was essential preparation in order for the analysts to properly evaluate the Builder. The case situation for A1 was to learn to use a self-selected evaluation technique by searching, finding, and reading relevant literature on usability evaluation methods so as to be able to conduct a system evaluation. The case situation for A2 was to only read Wharton et al. (1994) as background for applying the CW technique to the Builder. Table

7 shows all papers read by either one or both analysts in the evaluation phase. Given the case of A1 it is quite natural that he read several papers that had little to do with the CW technique. For example, A1 read some papers before he actually decided to use CW as his technique (e.g., about heuristic evaluation). A1 also came across non-CW papers that he thought might be useful to evaluate the Builder and read those. As noted by A1 in his diary, some papers gave him more insight and were more relevant to him than others.

A1: hour into the process	A2: hour into the process	Authors of paper (see bibliography for further details)	Paper about
1		Jeffries et al. (1991)	A comparison between four usability evaluation methods including CW, Heuristic Evaluation, Guidelines, and usability testing
2		Howes & Young (1991)	Programmable User Model and Task Action Grammars
6-7, 29	4-5	Gallagher & Meter (1993)	The specification document on the Builder
9		John et al. (1992)	A strategic plan for creating a HCI Institute at Carnegie Mellon University
10		Mack & Nielsen (1993)	Report on a workshop held at CHI'92 on usability inspection methods
11-12		Nielsen (1993) (chapter 5 and chapter 8)	Chapter 5 is about Heuristic Evaluation and Chapter 8 is about Interface Standards
11		Carroll & Rosson (1989)	The use of task scenarios as design tools in software development
14, 18		Lewis et al. (1990)	Description and evaluation of the CW technique applied to walk-up-and-use systems
15-16		Wharton et al. (1992)	Description and evaluation of the CW technique applied to more complex interfaces
17, 18		Polson & Lewis (1990)	The theory CE+ which is the foundation for CW
19		Rowley & Rhoades (1992)	Describing a fast-paced version of CW called the Cognitive Jogthrough
20		Bell et al. (1991)	A description of the programming walkthrough
20-21, 23		Wharton et al. (1993)	A practitioner's guide to CW; the tech report that was a forerunner of Wharton et al. (1994)
26		Wharton & Lewis (1993)	The role of psychological theory in usability inspection methods; a tech report that was a forerunner of Wharton & Lewis (1994)
	1-4, 6-7, 13	Wharton et al. (1994)	A practitioner's guide to CW; in content very similar to the tech report Wharton et al. (1993)

Table 7. The papers read by the analysts sorted chronologically according to A1's reading process. Note that Wharton et al. (1993) and Wharton et al. (1994) are two different versions of the same paper. The column "hour into the process" gives an idea of when in the process the papers were read and approximately how long it took the analysts to read a given paper.

Given the different case set up for A2, A1 read many more papers than A2. After his evaluation A1 suggested that CW novices should only read Wharton et al. (1993). In the meantime the tech report (Wharton et al., 1993) was included as a chapter in Nielsen & Mack (1994). Hence, A2 read this newer, published version of the paper referred to as Wharton et al. (1994). As shown in Table 7, A2 only read one other paper, the Builder specification.

5.2.2 The impact of reading the specification document

Both analysts recorded having read the specification for the Builder for several hours at a time. (Since our diary form asked for reports of activities only every half-hour, it is likely that the analysts did not record any quick skimming of the specification). A1 detected six problems while reading the specification document and another two problems when setting up action sequences based on the specification document. A2 on the other hand did not record any problems while only reading the specification document, but he recorded two problems while setting up action sequences based on the specification document. Hence, reading the specification document had an impact on the detection of problems in the interface. Of the total problems detected by the analysts A1 found 17% and A2 found 29% while reading the specification or while setting up action sequences based on the specification document. The fact that simply reading the specification can lead to PDRs is consistent with other research into UEMs (John & Marks, 1997).

5.2.3 The impact of reading papers other than the specification document

Besides the specification document A1 read several papers while A2 only read Wharton et al. (1994). Here we want to determine whether these different reading strategies influenced their evaluation outcomes.

A1 made 41 notes about papers that he read (excluding notes about the specification document), while A2 only made four such notes. A1's notes before the thirteenth hour explain his rationale for choosing to learn CW over other UEMs. He did not consider himself an HCI expert, from which he concluded that he should not use Heuristic Evaluation (Nielsen, 1993). (Note that we are not judging that his conclusion is correct in this regard, we are simply reporting his conclusion and the reason he gave). With only a specification document at hand a think-aloud study could not be arranged, and without access to the source code of the Builder he could not use Programmable User Model (Howes & Young, 1991) as his evaluation technique. Guidelines (Smith & Mosier, 1986) was a viable option, but he found it too difficult, as too much reading would have been necessary to acquire the expertise to apply guidelines to the specification document. Because of these constraints, he thought CW the most appropriate choice.

A1's later notes focused more explicitly on the CW technique itself. We were able to directly link 36 of A1's notes to a specific phrase in a specific paper (either because the note included an explicit reference or because a note was taken from or directed to a section of a paper). Ten of these notes were insights or difficulties regarding the theory CE+, on which CW was based. While reading Lewis et al. (1990) A1 wondered if he had to extend the CW technique with a memory load model, because CW initially was created for walk-up-and-use systems and he considered the Builder to be a more complex interface. Later in the process but still reading Lewis et al. (1990) he was concerned that he lacked sufficient knowledge about the theory, as Lewis et al. state "*successful execution of the walkthrough methodology would require deep knowledge of the theory*" (p. 241). It was therefore not surprising that A1 found and read the initial paper on CE+ (Polson & Lewis, 1990); he simply did not consider himself ready to do a CW without a better understanding of the theory behind CW. Reading Polson & Lewis (1990) did eliminate his fear about using the technique. However, the last note A1 wrote while reading this paper led back to an earlier question: is the Builder a walk-up-and-use system? A1 decided, or at least noted, that he wanted to check whether the basic assumptions of CE+ would apply to a more complex interface. Later in his reading process he felt confident about applying CW to the Builder, as he noted "'*The underlying assumption is that the user's selection of actions is largely guided by the interface' [Wharton et al., 1993]. This assumption holds for the [Builder], I think!*".

The theory behind CW was not the only concept that A1 attended to while reading papers on the CW technique. He was also concerned about practical problems and pitfalls when using the technique. While reading Wharton et al. (1992) he noted that "*Filling out the forms at each step interferes with the thread of thought when performing a task 'naturally'*". Reading the same paper he also noted, "*Because of the focus on individual user action there is the danger of overlooking interface problems concerning the task as a whole*".

In contrast to A1's extensive reading, A2 read Wharton et al. (1994) and he had only four comments that could be connected directly to a phrase in this paper. One of the difficulty notes, "*p. 110: What do they mean by 'how the user is expected to view the task…'*" referred to the quote from Wharton et al. (1994), page 110: "*What is the correct action sequence for each task and how it is described? For each task, there must be a description of how the user is expected to view the task before learning the interface. […]*". The three other notes, and some notes not connected to a specific page in the paper, merely summarized important aspects of the technique. Overall, it seems that A2 had no serious difficulties reading and understanding Wharton et al. (1994). He wrote no notes that identified major difficulties.

As would be expected neither of the analysts detected problems while reading CW papers. In general A1 was very informed about possible obstacles when using CW. Many insights and difficulties were recorded over an extended period of time from multiple sources, and when A1 felt insecure about a certain aspect of the technique, he located more literature that could help him better understand these aspects. A2, on the other hand, read only one paper (as requested) and wrote down only a little information about his learning process and the use of the CW technique. Though the number of papers A1 read and the number of notes he wrote strongly suggest that A1 was more introspective than A2 with respect to using the CW, this does not necessarily explain the large difference in the number of problems detected by the two analysts. By more closely analyzing the actual CW evaluation process, we can determine which aspects account for these differences.

5.3 Defining the user and choosing task scenarios

Defining the user and choosing task scenarios were complex tasks that seem to have affected end outcomes. In the first sub-section we will address how the analysts defined their fictive user, why they defined them as they did, and what impact this user definition might have had on the outcome of the evaluation. In the second sub-section we will describe the analysts' strategies for choosing task scenarios. Then we will discuss why they had different strategies. Finally we will suggest what impact user definition and task selection might have on the outcome of the evaluation.

5.3.1 How did the analysts define users, why, and what was the impact?

Wharton et al. (1994) offer two ways in which the user can be described as a part of the preparation phase – either in a general description like "*people who use existing ATM machines*" or more specifically like "*Macintosh users who have worked with MacPaint*" (p. 109). In his final report A1 made specific assumptions about users and context of use: "*[…] we assumed that the users are experienced Macintosh users, but don't have prior experience with the [Builder]*".

Ten hours into the evaluation process A2 wrote an e-mail to the second author containing the following phrase*: "Identification of users: Faculty, lecturers with knowledge of the Macintosh interface*". Four hours later he wrote the following insight note while preparing his first walkthrough: "*I'm not going to catalog information about user classes unless something comes up. I think that there is one class of user identified and that's enough*".

It is quite clear that neither of the analysts spent much time defining the user. A1 was concerned with the fictive user in relation to creating the task scenarios but he wrote nothing about defining or identifying users in his paper diary. A2 made one note about defining a user, and in his final report he mentioned user definition as one of several inputs to the walkthrough, but without amplifying this in relation to the Builder. Overall, there were no surprises in the quite general way that the analysts defined their fictive users, as both analysts followed the suggestions and examples from Wharton et al. (1993, 1994), keeping the user definitions short and general.

The necessity of identifying the user in preparation for completing a CW is closely related to the walkthrough process. For each action in an action sequence four questions are asked and answered based on the definition of the fictive user. That is, if our two analysts had identified different fictive users as the basis for their evaluations, the outcomes would certainly have differed. This is similar to when a think-aloud study with a novice user provides results unlike those that would result from a think-aloud study conducted with an expert user. This, however, does not imply that two evaluators defining the same fictive user will end up with the same problem list.

The dilemma, we believe, lies partly in the nature of the identification of the fictive user. If the description of the fictive user is both short and general, the use of this as input to the walkthrough dramatically amplifies the evaluator effect for at least two reasons: lack of guidance in answering the four questions reliably and anchoring the answers to the evaluator's own experience. This anchoring hypothesis is also suggested as one reason for the individual differences among CW novice evaluators in Hertzum & Jacobsen (1999). In discussing the walkthrough process (see section 5.5.3) we will reveal evidence from the database concerning the guidance and anchoring hypotheses to investigate if evaluators' outcomes were affected either by degree of guidance or self-anchoring.

In the next three sub-sections we will focus on another input to a CW, namely the selection of task scenarios, the method and rationale for selecting task scenarios, and the impact of this stage of the CW on the problems predicted.

5.3.2 How did the analysts choose task scenarios?

The example multi-media document given to the analysts contained the content of a sample volume: the text, graphics, table of contents, glossary items, and hyperlinks that would be found in such a volume. This document was A1's main source for generating task scenarios, although he was also inspired by the function descriptions in the specification document. He created three scenarios in an iterative manner. First he constructed a short scenario that created three pages of the example multi-media document (which included text, figures, and animations as well as adding several glossary items). After setting up correct action sequences and walking through this scenario and feeling confident of this method of choosing a scenario, he constructed a second scenario – this time covering 12 pages of the example multi-media document. The second task scenario purposely included actions that were not required in the action sequence to simply create the 12 pages of the example multi-media document. He chose to resize the glossary pane, edit a glossary item, cancel a sequence where he began setting up a cross reference, resize and move a frame, change the hierarchy of Table of Contents items, and find pages through a bookmark function. The 12 pages of the example multi-media document could have been duplicated without including these actions but A1, guided by the specification document, included them. After the second scenario was transformed into a correct action sequence and was walked through, he constructed yet another scenario where he included recovery-from-error and undo situations, noting that: "*Since, during walkthrough, the evaluator always 'stays on track' as guided by the correct sequence of actions, there is no way to evaluate how much an error-recovery is supported through the interface*". Thus, the third scenario was a method to get around one of CW's deficits, as CW does not prescribe ways to investigate recovery from error situations.

A2, with a very different perspective on creating task scenarios, based his construction of scenarios solely on the user interface specification document – a typical document describing the Builder function by function. This decision was captured in an e-mail to the second author 10 hours into the evaluation process: "*I used the Volume View Specification doc to generate a list of actions that it says a user will want to perform while using the Builder. They say (Wharton et al., 1994, p. 110) that task selection can be made based on requirements analysis as one source. I think the UI design doc can be used to extract requirements as it stands*". Hence, A2 created 11 sample tasks (see Table 8), each covering a specific feature or a special aspect of a feature in the system that could be tracked to a single section in the Builder specification.

A2's task scenario title	Portion of the Builder specification corresponding to A1's task scenarios	
	Heading	Page
Page through an existing volume	Buttons	20
Create a new volume	Creating a New Volume	22
Create a new page	Adding and Removing Volume Pages	26
Add a text frame to a page	Creating and Editing Text Frames	26
Reposition a frame	Moving a Frame	26
Add a glossary entry to a text	Adding Glossary Entries	24
Add a cross reference to a text	Setting Up a Cross Reference	31
Delete a cross reference	Deleting a Cross Reference	26
Add a Table of Contents entry	Adding Table of Contents Entries	36
Lock a frame	Locking a Frame	36
Unlock a frame	Unlocking a Frame	36

Table 8. A1's task scenario titles and the corresponding portion of the Builder specification.

Two aspects of constructing task scenarios differ between the analysts. They chose two different documents as the basis for constructing their task scenarios and they described their task scenarios differently both in terms of grain size and system coverage. The difference in grain size of task scenarios (e.g. A1's "Create p. 24-35 in multi-media example document" and A2's "Lock a frame") is quite obvious. The former scenario describes an extensive work process, which later in the evaluation was transformed into 119 actions, while the latter scenario can be performed with three mouse clicks. As the grain size of the task scenarios was so different the number of task scenarios for the two analysts cannot be directly compared. Instead we categorized the coverage of the task scenarios by annotating any given action with a feature and an aspect of feature (see Table 9).

A *feature* is a limited part of the Builder: an *aspect of a feature* is a way of making use of that particular feature. For example, the glossary *feature* has the following *feature aspects*: add glossary, delete glossary, edit glossary definition, scroll through glossary pane, find glossary item through text frame, change size of glossary pane. A total of 19 features and 74 aspects of features were identified through the specification document (see Table 9).

Feature	Aspect of feature	A1	A2
table of contents	add TOC	x	x
	delete TOC		
	change location laterally	x	
	renaming TOC		
	change size of TOC pane		
simple navigation	page up button	x	x
	page down button	x	x
	go back button	x	
	go to page	x	x
	tab through volume		
	scroll bar action	x	
	to other parts of system	x	
	to other application	x	
glossary	add new glossary item	x	x
	add second instance of glossary item	x	
	delete glossary item	x	
	edit glossary definition		
	scroll through glossary pane		
	find glossary item through text frame	x	
	change size of glossary pane		
bookmark	add bookmark	x	
	delete bookmark		
	use bookmark to navigate	x	
save	simple save	x	
	save as		
	save a copy in		
cross reference	add X-ref	x	x
	delete X-ref	x	x
	change X-ref destination	x	
	navigate via X-ref		
volume page	add page	x	x
	delete page	x	
	customizing page size		
help	use balloon help		
	use help system		
undo	undo	x	
volume	create volume	x	x
	open volume	x	
	close volume	x	

Feature	Aspect of feature	A1	A2
general frame	create frame	x	x
	delete frame		
	select frame	x	x
	move frame	x	x
	resize frame	x	
text frame	select text frame	x	x
	select text in frame	x	x
	enter in text	x	
	cut text	x	
	copy text		
	paste text		x
	format text		
code frame	select code frame		
	select code in frame		
	enter in code		
	cut code		
	copy code	x	
	paste code	x	
picture frames	select pict frame	x	
	copy pict	x	
	paste pict	x	
movie frames	select movie frame		
	copy movie		
	paste movie	x	
other frames	use drawing frame	x	
	use call stack frame		
	use runtime controls		
	use I/O frame		
runtime controls	create control	x	
	delete control		
	select control		
	change frame type of control		
lock/unlock	lock a frame		x
	unlock a frame		x
printing	print volume		

Table 9. To measure the coverage of the analysts' task scenarios we identified 19 *features* and 74 *feature aspects* based on the specification document. A1 touched on 16 features and covered 41 feature aspects, while A2 touched on 9 features and covered 17 feature aspects.

Analyzing the coverage of the task scenarios through our categorization we found that A1 touched on 16 different features (i.e. covered at least one aspect of that feature) and A2 touched on 9 different features. Similar results are found when aspects of features are compared, as A1 covered 41, while A2 only covered 17 feature aspects. Thus, A1 covered more than half of the feature aspects that could be covered in the interface, while A2 covered less than one quarter of the feature aspects of the interface.

5.3.3 Why the analysts choose task scenarios as they did

While A1 was aware of the importance of creating appropriate task scenarios as part of the CW evaluation, A2 did not seem to worry much about this issue. In fact A2 did not write a single note about task scenarios, while A1 wrote 14 notes about them (as shown in Table 10). In those notes A1 mentioned some important aspects of creating task scenarios. He wanted to talk to end users, students and instructors to investigate which scenarios to evaluate. Although he had assumed he might find explicit guidelines about how to construct scenarios by reading *Human-Computer Interaction Scenarios as a Design Representation* (Carrol & Rosson, 1989), he did not. Moreover, he worried that he had no access to an expert in walkthrough techniques who could help him set up an appropriate task scenario as described in Wharton et al. (1992). From reading this paper he was aware of the drawback that realistic task scenarios often produces long action sequences. Before initiating the preparation phase (i.e. defining user and selecting task scenarios) he concluded that one of his main problems was likely to be setting up appropriate task scenarios.

Hours into the process	Note type	Note	Activity	Reading
4	insight	Realized difference between usability inspection (Heuristic Evaluation) and Walkthrough methods are the problem scenarios	meeting	Nothing
5	difficulty	Realize that to set up good scenarios for a walkthrough technique I need to be able to talk to both end-users, students and instructors; best those who have used the system previously	reading for 'what-it-is'	Gallagher & Meter (1993)
10	insight	Cognitive walkthrough: usability heuristics (criteria) embodied in some cognitive theory; task scenario; step through task scenario and apply criteria/heuristics	reading for 'what-it-is'	Mack & Nielsen (1993)
10	difficulty	How to come up with realistic task scenarios for a cognitive walkthrough. -> must talk to users of the Volume View, students and instructors	reading for 'how-to-do'	Mack & Nielsen (1993)
11	unmarked	Seems to be promising - but turns out that the main idea is to use scenarios instead of usability heuristics. I expected some methodology on how to design good scenarios	reading for 'what-it-is'	Carroll & Rosson (1989)
12	other	Although we are supposed to focus on the Volume View as a tool for the builder/explorer, the cognitive walkthrough may require a scenario, which includes the Volume View document or both, may not be separable in the scenario	meeting	Nothing
14	insight	Authors claim that the walkthrough takes just 1 hour/ task/ interface - this number can definitely include neither learning the technique nor the time to set up the scenarios nor the time to become familiar with the interface	reading for 'what-it-is'	Lewis et al. (1990)
14	difficulty	The task (scenarios) were selected by someone who was expert in walkthroughs - how should I get access to someone like that?	reading for 'what-it-is'	Wharton et al. (1992)
14	difficulty	Task selection stated as crucial - with complex interface a tradeoff between realistic task scenario and length of walkthrough is admitted.	reading for 'what-it-is'	Wharton et al. (1992)
14	difficulty	My main problems will be: appropriate task selection	reading for 'what-it-is'	Wharton et al. (1992)
19	insight	The "doctrine" is what I need to extract from the Volume View design documents. The task I need to construct from the Drosophilia screen dumps. Overall, a poor paper, too!	reading for 'what-it-is'	Bell et al. (1991)
30	difficulty	Even in this largely augmented version of a task scenario, there are no bookmarks, animations, slide/radio buttons => have to be artificially introduced into the task scenario for the walkthrough?	analysis	Nothing
33	difficulty	Question whether I should augment our task scenario by the missing functionality of the interface	meeting	Nothing
33	insight	I need to be aware of a couple of important areas in determining future task scenarios a) different type of frames (PICT, text, code, movie adopts appropriate type automatically, but drawing, call stacks, input/output, runtime controls not) b) explore redesigning and recovery in some task scenarios, too c) what happens if certain limits are violated, e.g. # PICT / drawing/ io frames on one page -> dialog box pops?	meeting	Nothing

Table 10. A1's notes about task scenarios in chronological order. The notes were made throughout the process with the first note made four hours after he began the process, and the last note made 14 hours before he finished his evaluation.

Another reflection documented in the notes is A1's awareness of the many functions not tested even when duplicating 15 pages from the example multi-media document. Thirty hours into the process he noted: "*Even in this largely augmented version of a task scenario, there are no bookmarks, animations, slide/radio buttons => have to be artificially introduced into the task scenario for the walkthrough?*". As noted earlier A1 actually decided to introduce these functions into his second task scenario.

Why did the analysts differ so significantly in their creation of task scenarios? The difference in reading process seems to be one major explanatory factor. Many of A1's insights and difficulty notes relating to task scenarios were based on several papers he had read in the initial phase of the process (see Table 10). Both analysts read Wharton et al. (1993, 1994)[4], but neither recorded any

[4] The explanation of how to choose task scenarios was identical in both versions of this paper.

diary notes about task scenarios while reading this paper. This evidence suggests that A1 paid great attention to task scenarios since he read papers other than Wharton et al. (1993). A2, on the other hand, only read Wharton et al. (1994) and thus could not have seen the importance of this aspect of the CW technique. Not only did the analysts have different levels of awareness about the creation of task scenarios, but also their reading processes gave the analysts different potential sources from which to create those scenarios. A1 did not explicitly note why he used the multimedia document as the primary source from which to construct task scenarios, but two of three task scenarios were taken directly from this document. A2, however, was explicitly guided by one of the suggestions from Wharton et al. (1994) in that he based his evaluation on parts of the specification document.

The suggestion from A1 about only reading Wharton et al. (1994) rather than reading more papers with a variety of perspectives seems to have had at least two drawbacks in relation to the creation of task scenarios. Wharton et al. (1993, 1994) address task scenarios as one of the inputs to a CW, and though much general advice is given in this short section they fail to clarify for novice analysts the importance of task scenario selection. A1 can easily recognize the importance of task scenarios after reading more papers. A2 however had difficulty prioritizing the suggestions in Wharton et al. (1994), which led to A2's failure to adequately focus on task scenarios. The other drawback to reading only Wharton et al. (1994) for information about creating task scenarios was that A2 focused on tasks rather than task scenarios and found it reasonable to use the specification document as the only source from which to construct his task scenarios[5].

5.3.4 What is the impact on task scenario selection?

Are there any connections between the numbers of problems detected by the two analysts and the number of features and feature aspects covered by their task scenarios? Given a rather crude, quantitative measure for their task scenario coverage we know that A1 covered 41 out of 74 aspects of features described in the specification document, while A2 only covered 17 out of 74 aspects of features. A1 detected 26 problems based on the task scenarios (i.e. while walking through action sequences) for a rate of 0.63 problems detected per aspect of feature covered. A2 only detected 4 problems while walking through action sequences for a rate of 0.24 problems detected per aspect of feature covered. Extrapolating, if both analysts had covered all features described in the specification document, A1 would have detected 45 problems and A2 would have detected 17 problems. Hence, the choice of task was not the only determinant of number of problems detected.

We can look at this another way by examining the problems detected in those aspects of features the analysts had in common. In the 14 aspects of features that both analysts covered, A1 detected 8 problems, while A2 detected 4 problems. Surprisingly, only one of these problems was detected by both analysts, even though both analysts set up task scenarios covering precisely the same 14 aspects of features (these differences in problem detection for the same feature aspects will be further discussed in section 5.4.3 and section 5.5.3). In other words, if we only consider those problems detected in the actual walkthroughs A1 would not have had the potential to detect A2's problems by extending his task scenario list, since the 4 problems detected by A2 were related to feature aspects already covered by A1. On the other hand, if A2 had extended his task scenarios to cover all feature aspects evaluated by A1, A2 might have detected the remaining 18 problems found by A1.

[5] In the latest guide to CW Lewis & Wharton (1997) have devoted a short section (28 lines) to clarifying the importance of choosing realistic task scenarios (p. 721).

In summary, the differences in task scenarios between the two analysts explains only some of the differences in the outcome of the evaluation. We have to more closely examine the development of both sets of action sequences and walkthroughs to further explain outcome differences, particularly for those feature aspects both analysts evaluated.

5.4 Transforming a task scenario to an action sequence

After defining the user and setting up appropriate task scenarios the preparation phase is concluded by transforming the high-level task scenario to a concrete action sequence annotated with expected system feedback for each action. In the following we will analyze how the two analysts transformed their task scenarios into action sequences, why they did this as they did, and what impact this process had on their evaluation outcomes.

5.4.1 How did the analysts transform scenarios to action sequences?

A1 expanded his three task scenarios into three action sequences consisting of 46 actions, 119 actions, and 25 actions respectively. A2 expanded his 11 tasks to a total of 49 actions with between 3 and 9 actions for each task. According to Wharton et al. (1993, 1994), "*These actions may be simple movements, such as 'Press the RETURN key' or 'Move cursor to File menu'. Or, they may be sequences of several simple actions that a typical user could execute as a block such as, 'login to the system' for experienced UNIX user, or 'Select Save from File menu' for experienced Macintosh users"*. We counted the number of movements in A1's and A2's actions, defining a movement for example as one mouse cursor move, one down click on the mouse, one up click on the mouse, one chunk of short text entered in by keyboard, etc. Hence, copying into the clipboard an item already selected on a Macintosh system comprised 4 simple movements: move mouse cursor to File menu, down click File menu, move mouse cursor to Copy item in pull down menu, and release mouse button. Table 11 shows that A1 generally made use of actions comprising more simple movements (an average of 4.5 simple movements per action) than A2 (an average of 2.4 simple movements per action).

	A1	A2
Less than 5 simple movements per action	42%	100%
Between 5 and 9 simple movements per action	56%	0%
10 or more simple movements per action	2%	0%
Total	100%	100%

Table 11. A count of simple movements for all actions revealed that A1 generally defined actions at a higher level of abstraction than did A2. While A1 typically defined a menu selection as one action A2 typically divided a menu selection into two actions (e.g. (a) click File menu and hold down mouse button (b) select Copy and release mouse button).

To avoid reiterations during the construction of action sequences and the later walkthrough A1 created two *macros* for small sequences of repeated actions. The first macro was created and executed in the first task scenario; the other macro was created and executed in the second task scenario. The first macro covered three actions that were sufficient for adding a glossary item to a glossary pane and writing a definition and explanation for the item; this macro was executed 15 times. The second macro covered six actions that were sufficient to add a frame and copy/paste a picture or text from another application into the newly created frame; this macro was executed 12 times. By using the macros A1 avoided having to write down and walk through 117 almost identical actions.

According to Wharton et al. (1993, 1994) each action should be annotated with the expected system feedback before walking through the action sequence. A1 chose to either draw full screen shots or parts of screen shots as the expected system feedback for the first task scenario. In the second and third task scenarios he simply wrote down the expected system feedback textually (e.g. "Bookmark appears" or "Cursor changes to 'X'"). However, A1 did not record any system feedback for 72 actions. Forty-six of the 72 actions that had no system feedback were duplicated actions from earlier parts of the action sequence, for which feedback had already been recorded. Of the 26 remaining actions with no recorded feedback, 7 were duplicates, so 19 unique actions were left with no feedback recorded at all. A2 wrote prose descriptions of the feedback. Only 4 of A2's actions were not supplemented with such descriptions.

The transformation of a task scenario to an action sequence is ambiguous in the sense that there are often several correct action sequences. Optimally, all correct action sequences should be tested, but this is too cumbersome in practice. On only one occasion did either of our analysts construct alternative action sequences (A1 looked at navigating through a volume both by using the arrow buttons and by using the GoTo Page menu item).

Table 12 shows how the analysts' action sequences differed for adding a glossary entry. A2 initiated his action sequence by clicking the pointer tool button in the tool palette. This action was not part of A1's action sequence because adding the glossary entry was part of a larger task for which the pointer tool had already been selected. Thus, although their sequences differed, both A1 and A2 recorded correct action sequences. In the second example, Table 13, the sub-task was to link two frames with a cross reference. Again A1 did not have to select the pointer tool button. However, his task scenario caused him to scroll a page, as the two frames were located on two different pages in his task scenario. In contrast to A2, A1 combined two simple movements into one action ("Click on animation frame" and "Click on Xref again" in his last step in the action sequence).

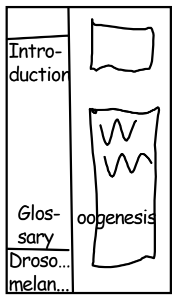
A1		A2		Investigators' explanation
Action	System feedback	Action	System feedback	
	 Intro-duction Glos-sary Droso... melan... oogenesis	Click on the pointer Tool icon	The pointer Tool icon darkens and the other tool icons are light	The pointer tool button with icon "↖" was found in the tool palette. This action ensured that a word could be selected.
Select "oogenesis"		Select some text (a word or phrase)	The text is highlighted	A1 specified the specific text to be selected, while A2 only described the type of item to be selected.
Click on "Gloss" button		Click on the glossary icon	The selected text appears underlined in the glossary window with an insertion point underneath	The glossary button with the text "**Gloss**" was found in the tool palette.
Type in text for "oogenesis"		Type the definition of the text for the glossary	The text appears at the insertion point	At the insertion point in the glossary pane the user could specify the glossary definition.

Table 12. An example of the analysts' action sequences and system feedback for a sub-task they had in common, namely adding a glossary item to a glossary pane. A1 did not need to click on the pointer tool button in order to select text, as previous actions had automatically activated the correct tool button. Apart from the wording of the actions and the fact that A1 drew diagrams to document the system feedback, the sub-tasks for the two analysts are similar.

A1		A2		Investigators' explanation
Action	System feedback	Action	System feedback	
		Click on the pointer Tool icon	The pointer Tool icon darkens and the other tool icons are light	The pointer tool button with icon "↖" was found in the tool palette. This action ensured that a frame could be selected.
Select "Figure 17"	[Left blank]	Select some text; a word or phrase	The text is highlighted	A1 specified the specific text to be selected, while A2 only described the type of item to be selected.
Click on Xref	Dialog box Fig. 24, p.28	Click on the Cross Reference icon	A dialog appears telling the user to select the destination and then click the Cross Reference icon again	The Cross Reference button with icon "**X Refs**" was found in the tool palette. A dialog box appeared with the text "*Please select the destination of this cross reference, then click the Cross Reference tool again. The destination must be a frame*" and two buttons OK and Cancel. A1 pointed to the specification of the dialog box; A2 described it.
Click OK	Cursor changes to X	Click the OK button	The dialog goes away. The Cross Reference icon flashes. The mouse cursor changes to an X	The OK button on the dialog box was activated.
Page forward	p. 27 appears			According to A1's task scenario he had to make a reference to a frame on p. 27, while A2 chose to make a reference to a frame on the same page, hence correctly omitting flipping a page.
Click on animation frame. Click on Xref again	[Left blank]	Select a frame as the Cross Reference destination	"Handles" appears on the frame	A1 combined two atomic actions into one action, while A2 had two actions for the two movements: Select a frame…
		Click on the Cross Reference icon	The Cross Reference icon stops flashing. The mouse cursor returns to a pointer. The destination frame is marked with a "tag" in the upper left corner. The source of the reference (text) will appear in bold type	…and click on the Cross Reference button

Table 13. Another example of the analysts' action sequences and system feedback for a sub-task they had in common, here linking two frames as cross references. A1 did not need to click on the pointer tool button in order to select text, as previous actions had automatically activated the correct tool button. As the two frames to be linked appeared on two different pages in A1's task scenario he had to scroll pages in the middle of selecting frames.

5.4.2 Why did the analysts transform scenarios into action sequences as they did?

The process of transforming a task scenario into an action sequence concerned A1 a great deal. While reading Lewis et al. (1990) he asked himself *"[...] how likely is it that an actual user deviates from the direct path?*". This question is followed by more specific questions about who should specify the correct action sequence. For instance, one early note reads: *"The sequence(s) of actions which successfully perform a scenario are to be specified by the designer*". This reads very much like a sentence in Lewis et al. (1990), p. 238: "*Next, the sequence of user actions that will successfully perform a given task is specified by the designer*", which A1 in fact read while writing his own note. Much later in the process A1 read Wharton et al. (1993) and wrote another note related both to this paper and to the question of specifying action sequences: *"So they claim that the focus of the walkthrough is on learning by exploration. But how can this be captured if the sequence of actions is to be determined by the developer? - Therefore, my idea to have the evaluator determine the sequences of actions seems better - also confirmed in personal communications with C. Wharton*". However, two more notes recorded one hour and three hours later reveal his worry about specifying the action sequences correctly. He wrote: "*[...] before I do the walkthrough I must have the sequence of actions validated by the designer and the corresponding feedback of the system, otherwise my evaluation will depend on how well I understood the interface out of this incomplete spec*" and "*Realize need that designer checks both my action sequence and the intuitive idea I have about the interface's feedback*". A1 finally decided to construct his action sequences on his own.

After the problem of who should transform task scenarios into action sequences had been resolved, another problem emerged. Before A1 decided to use CW as his technique of choice, he wrote this note: "*Realize that many things are not documented in the specification and it will be hard to get a deep understanding of the system*". This note was then followed by 27 difficulty notes with specific questions on how the system worked in various situations. He noted for example: "*Couldn't figure out how to start the Volume with an empty volume, in order to create task A (Spec p. 4)*", "*Was unsure what the feedback is when the FRAME button is clicked (Spec p. 18, "Frames Tool")*", and "*What exactly happens if builder wants to save the just created V.V. What's the difference between 'Save As' and 'Save a Copy in' (Spec p. 4)*". A1 met with one of the designers of the Builder twice in his evaluation process, and many of the 27 questions were answered in these meetings. His many notes show that transforming a high-level task scenario into action sequences based on a written specification document is quite complicated.

A1's notes also explain his decision to construct macros as a way of handling repetitive actions. Twenty-five hours into the process A1 noted: "*How to handle frequently recurring sequences of actions? Introduce MACRO's for'em?*". At first blush this seems to be a good extension of the CW described in Wharton et al. (1994). However, looking closely at the small sequences of actions embedded in the macros, it strikes us that what A1 finds tedious and tiresome might also be what actual users will find tedious and tiresome. Why, for example, is it necessary to create a frame every time any object is to be copied into a volume in the Builder? Today frame-based applications are rare, perhaps because frame-based applications introduce non-productive actions (in our case creation of a dummy frame for entering in text, figures, animations, movies, etc. in a volume). It seems that when a CW analyst finds actions repetitive in a real-life task scenario it is also likely that it is repetitive (and typically non-productive) for the user. In this respect identifying a macro in order to save the CW analyst's time might instead be interpreted as detecting a potential usability problem.

A2 wrote fewer notes than A1 during the transformation phase, so we have fewer insights into why he transformed his action sequences as he did. First he wrote two notes summarizing the purpose of the task scenarios, including one in the first hour of the process ("*Critical features: provide links between user task description and the correct action, provide feedback indicating the previous actions advanced progress*"), and another written seven hours into the process with reference to Wharton et al. (1994) ("*OK. Page 110 describes the action sequence part. On page 111 they refer to 'actions in the solution path'. This must be the action sequence for a task*"). A2 also decided who should create the action sequences: "*From the experience I've had with [Software Engineering] techniques I think that the action sequences are better if they are assembled by someone besides the designer. Designers tend to gloss over undocumented details because they have an intimate view of their objectives. I think that action sequences will better reflect the documents they are based on if the assembly is mechanical, more objective*". A2 had no specific questions about how the interface worked in certain situations, or at least wrote no notes about this issue.

Clearly, their choice of task influenced their action sequences, as shown in the example tables above. A1's tasks were large and sub-tasks flowed from one to the other, adding specific context to their intial conditions (e.g., what tool was selected, how many pages away a target was). A2's tasks were small and he usually began with a tool selection.

The difference in grain size of actions may be attributed to Wharton et al. (1993,1994), whose example actions included some actions consisting of a few simple movements (like A2 used) and others comprised of a series of simple movements (like A1 used). Wharton et al. (1994, p. 110) offer the following rationale for including actions of different granularity: "*The decision as to what level of action granularity is appropriate depends primarily on the level of expertise of the expected users".* Although A1 and A2 both defined their expected users as experienced Mac users, evidently their interpretations of "experienced" influenced their choice of action granularity differently.

5.4.3 What is the impact of the transformation phase?

From section 5.3.2 we know that the analysts' task scenarios covered different interface features. In order to analyze the impact of the transformation phase in this section we will examine only those problems detected by A1 and A2 that were associated with walking through the 14 aspects of features they both covered. (In A1's case, this includes 8 problems; in A2's case, 4). The question is whether differences in the number and type of problems can be related to differences in specifying correct action sequences for the same aspects of features.

Table 14 shows the seven feature aspects that both analysts identified and that led to at least one problem detection. Both A1 and A2's action sequences for the seven feature aspects are revealed in the table, where an action in bold indicates both the point at which the analyst reported a failure story as well as what the problem was.

Analyzing the content of Table 14 we find that the two analysts transformed six of the seven aspects of features into equivalent action sequences ("add volume page", "add new glossary item", "add cross reference", "add table of contents", "simple navigation", and "delete cross reference"). Equivalent action sequences are defined as the same actions in the same order, though they are neither necessarily the same number of simple actions nor necessarily using the exact same words. Of the six aspects of features that the analysts specified in a similar manner, only one of the detected problems was the same (similar problem descriptions for aspect "add cross reference" in Table 14 are shadowed). In three cases ("add volume page", "add new glossary item", and "add cross reference") both analysts detected problems although *different* problems; in three cases ("add table of contents", "simple navigation", and "delete cross reference"), only A1 reported problems. One of the seven common feature aspects was transformed into different action sequences by the two analysts ("create volume"). A2 actually set up a wrong action sequence for this feature aspect, as his action sequence made him create a new environment rather than a new volume (a volume is created within an environment, which might in fact contain several volumes). Hence, A1's reported problem on creating a volume might have been detected by A2 had he set up a correct action sequence for this feature aspect.

Aspect of feature	A1		A2	
	Actions	Problem(s) detected	Actions	Problem(s) detected
Add volume page	**1. Select Edit - Add Volume Page**	Add Volume Page is misplaced under the "Edit" menu	**1. Select a frame** 2. Pull down the Edit menu 3. Highlight the "Add Volume Page" menu item and release the mouse button	To create a new page you need to create a frame first, select it and then create the page. Why can't the user create an empty page
Add new glossary item	1. Select "Drosophila Melongaster" **2. Click on "Gloss" button** 3. Type in text for "Drosophila Melongaster"	Glossary button "Gloss" doesn't support label-following by user	1. Select some text; a word or phrase 2. Click on the glossary icon **3. Type the definition of the text for the glossary**	No explicitly feedback is given after adding a glossary entry
Add Cross Reference	1. Select "Fig. 17" 2. Press Xref 3. OK 4. Find - Goto page 29 5. Create Fig. 18 **6. Click on Xref** 1. Select "Figure 17" 2. Click on Xref 3. Click OK 4. Page forward **5. Click on animation frame b) Click on Xref again**	In the Specification Document the tag symbol is located on the same spot as where the lock/unlock symbol is said to belong Necessity to click on Xref button a second time, after having selected the destination, will create problems	1. Select some text; a word or phrase 2. Click on the Cross Reference icon 3. Click the OK button **4. Select a frame as the cross reference destination** **5. Click on the Cross Reference icon**	The user is left to figure out what to do after clicking the Cross Reference button. A novice user will not know to click on a frame next. There's an extra step to setting a cross reference: you have to click the cross - reference button again. This seems poor. The icon gives some feedback but it's not obvious. See #9 about a dialog with instructions
Add Table of Contents	1. Select "Introduction" **2. Click on TOC Button**	User will have difficulties associating TOC with Table of Contents - no label following is possible	1. Select a text frame 2. Click on the TOC icon	(No problem detection)
Simple navigation – go to page	**1. Find - Goto Page 1**	User will not find the most efficient command "GoTo page" under the right menu	1. Pull down the File Menu 2. Highlight the "Go To Page..." menu item and release the mouse button 3. Enter a page number which is within the valid range of page numbers	(No problem detection)
Delete cross references	1. Select "Fig. 17" **2. Press Xref Button** 3. Click on "Delete"	Deleting Xref: Mac experience would lead to assume that Edit - Delete performs this action, but not Select - XRef !	1. Select a cross reference 2. Click on the Cross Reference icon 3. Click the delete button	(No problem detection)
Create volume	1. Click on Drosophila Icon **2. Choose "Show Volume" from the "Windows" menu**	Starting a volume in builder mode: Necessity to choose Windows - Show Volume to get into builder mode with the current volume, contradicts Mac experience where you get into the correct application just by clicking on a file created by this application	1. Pull down the File menu 2. Highlight the "New" menu item 3. Select a size with the mouse, or leave the default selected 4. Click the OK button with the mouse or press "Enter"	(No problem detection)

Table 14. Aspects of features that the two analysts had in common and that led to a problem detection by at least one of the analysts. For most aspects of the features the analysts transformed their task scenarios into equivalent action sequences, but in one case A1 set up a wrong action sequence ("create volume"). Bold actions triggered the analysts to answer "no" to at least one of the four questions leading to a problem detection. Of the six aspects of features that the analysts specified in a similar manner, only one of the detected problems was the same (shadowed).

From this analysis we have sufficient evidence to conclude that for these analysts the transformation phase from setting up an action sequence to specifying an appropriate and correct action sequence had no systematic impact on problem detection. For the feature aspects that they had in common, most were similarly transformed into action sequences. Action granularity did not seem to matter because A1 seemed to simply delve into the details of his simple-movement sequences to identify problems. Interestingly, only one of the equivalent action sequences led to the detection of the same problem by both analysts ("add cross reference"). Their agreement about how to set up action sequences and their lack of agreement about reported problems for similar action sequences suggests that the differences between the two analysts cannot be attributed to problems in the transformation phase.

5.5 Walking through the action sequences

According to Wharton et al. (1994) it is advisable to imagine a walkthrough of each action in an action sequence by answering four questions:

1. Will the user try to achieve the right effect?
2. Will the user notice that the correct action is available?
3. Will the user associate the correct action with the effect trying to be achieved?
4. If the correct action is performed, will the user see that progress is being made toward solution of the task?

The questions are said to be "loose guidelines" (Wharton et al., p. 112) to support the inspection of the interface. For each question the analyst should come up with a credible story of either success or failure. If, for a given action, the analyst comes up with success stories for all four questions, these judgments imply no problem detection. If the analyst comes up with a failure story for any of the questions this judgment implies that the analyst has detected a potential problem.

In this section we will 1) reveal how the analysts walked through their action sequences, 2) explain why they walked through their action sequences as they did, and 3) discuss the impact of this phase on the evaluation outcome.

5.5.1 How did the analysts walk through their action sequences?

Rather than using the preexisting set of four questions suggested by Wharton et al. (1994) as loose guidelines, A1 created the following checklist:

1. What effect to achieve (part of original task/experience using the system/system prompted)
2. Whether an action is available (experience/device - e.g. buttons visible)
3. Whether an action is appropriate (experience/label following possible/all other actions look wrong)
4. That things are OK? (experience/recognize connection between system response and lesser goal)

The four points are almost identical to points in the section "*Common Features of Success*" in Wharton et al. (1994), p. 115-116.

A1 used a clear structure for his walkthrough (see Figure 5). He had consecutively numbered his actions and referred to these numbers in his walkthrough process. Hence, a walkthrough of a particular action consisted of an action number followed by the question numbers (i.e. 1, 2, 3, or 4); for each question number he briefly described (in handwriting) the success or failure story. If any of the four questions for a given action were answered by a failure story he summarized that action with the text "failure"; otherwise he summarized the action with the text "success". Although this walkthrough pattern was the most common for A1 he used different types of shorthand notation (see below), probably to remedy the fairly slow process of recording his walkthrough.

A1 recorded 190 actions in his three task scenarios. With four questions for each action he should have recorded (190 x 4=) 760 success or failure stories. In fact he only recorded 206 success or failure stories with full text. Three types of shorthand notation, however, can to some extent be considered acceptable, recorded walkthroughs, namely 1) macros, 2) explicit references to previous walkthroughs, and 3) implicit references to previous walkthroughs. As noted earlier A1 created macros for repeated actions while setting up action sequences. These macros saved him time during the transformation phase because he could refer to small sequences of consecutive actions in a single step. In the walkthrough process he saved even more time by creating macros, as he only needed to walk through these actions once; later in the process he simply referred to these macros as "previously walked through." A total of 80 success or failure stories were designated as macros.

Some basic actions were repeated in his action sequences several times without being part of a macro (for example, typing in text). A1 sometimes chose to refer to repeated actions in his walkthrough process either explicitly, i.e. by numbering the action/walkthrough that had been repeated (which covered 28 success or failure stories) or implicitly by writing, "previously walked through" (which covered 117 success or failure stories). Hence, out of the 760 success or failure stories that could have been recorded 431 (206 full text + 80 macros + 28 explicit references + 117 implicit references) were recorded either fully or by referring to an earlier walkthrough.

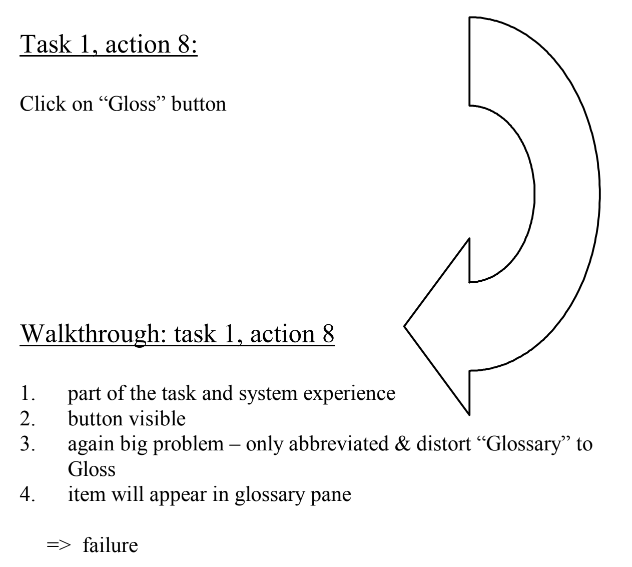
For the remaining walkthroughs, A1 did one of the following: constructed stories not associated with any specific of the four questions (5 actions = 20 stories), recorded either a "success" or "failure" but without giving a credible story (201 stories), or did not record anything (108 stories).

Summarizing these results, roughly 60% of A1's walkthrough process was recorded with success or failure stories or references to previous walkthroughs, 25% was walked through with no recordings of the story but only with words "success" or "failure", and 15% was not recorded at all.

Walking through action sequences allowed A1 to identify only 26 of his 46 detected problems. Seven problems were detected while walking through the first task scenario, 14 problems while walking through the second task scenario, and 5 problems while walking through the third task scenario. Compared to the number of actions in each task scenario A1 detected increasingly more problems for each task scenario: 9%, 12%, and 20% of the actions in task scenario one, two and three received failure stories.

The problems were not equally distributed among the four questions. Questions number 1 (user's goal) generated 4 problems. Question 2 (availability of action) also generated 4 problems. Question 3 (associating the action with the goal) generated the most problems, 16. Question 4 (adequate feedback) led to 2 problems.

A2 wrote down the questions from Wharton et al. (1994), p. 112, but had difficulty remembering examples of credible stories for each question. As a result, he decided to create what he called a cheat-sheet – a sheet of paper with the questions and the most common features of success (Wharton et al., 1994, pp. 115-116). In some sense A2's walkthroughs were easier to read than A1's (see Figure 6). In particular, A2 used a word processing system as a tool to record his walkthroughs, while A1 used pen and pencil (in the next section we will discuss how the tool had an impact on the walkthrough process). Moreover A2 not only referred to his action sequences by number, but he also copied the action sequence name as a heading for each walkthrough. Underneath the heading (the textual action) he wrote a success or failure story and annotated each story with his explanation, as guided by his cheat-sheet. That is, if A2 judged an action to be successful he defended this standpoint with credible success stories. On the contrary, when he judged an action to be a failure story, he defended this standpoint based on his response to one of the questions. A2 listed his credible stories on bullets rather than numbers without referring directly to any one of the four questions. However, categorizing A2's stories into one of the four questions was straightforward because he used success stories copied directly from Wharton et al. (1994). Figure 6 shows an example of A2's recording of his walkthrough of an action.



Task 1, action 8:

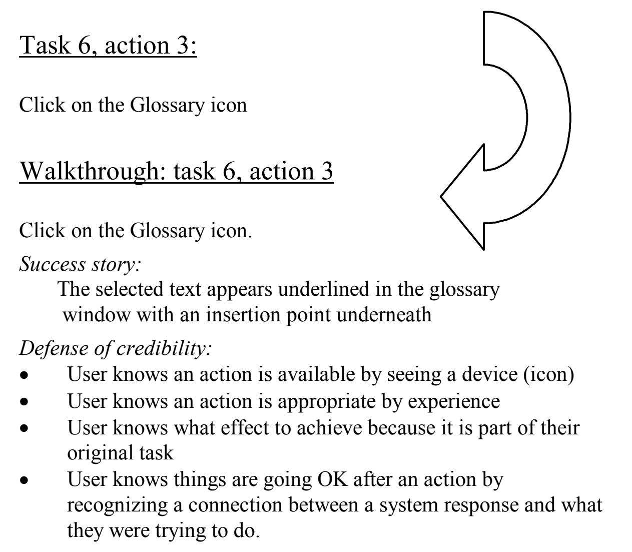
Click on "Gloss" button

Walkthrough: task 1, action 8

1. part of the task and system experience
2. button visible
3. again big problem – only abbreviated & distort "Glossary" to Gloss
4. item will appear in glossary pane

=> failure

Figure 5. An example of A1's recording of a walkthrough. The numbers 1-4 refer to the four questions described in Wharton et al. (1993).



Task 6, action 3:

Click on the Glossary icon

Walkthrough: task 6, action 3

Click on the Glossary icon.

Success story:
The selected text appears underlined in the glossary window with an insertion point underneath

Defense of credibility:
- User knows an action is available by seeing a device (icon)
- User knows an action is appropriate by experience
- User knows what effect to achieve because it is part of their original task
- User knows things are going OK after an action by recognizing a connection between a system response and what they were trying to do.

Figure 6. An example of A2's recording of a walkthrough. The bulleted items refer to the four questions described in Wharton et al. (1994) in the order 2, 3, 1, 4.

As noted earlier, A2's 11 task scenarios were transformed into 49 actions. Asking four questions for each action, he could have recorded 196 success or failure stories. In fact A2 only recorded 128 success or failure stories, with the remaining 68 credible stories missing. This was because A2 did not completely answer the sets of questions. In fact, while A2 did not miss a single action from his action sequence, more often than not he recorded stories partly by answering one, two or three rather than all four questions for a given action. For 10% of the actions (5 actions), the analyst recorded only one credible story (in 4 of these 5 cases, failures were recorded). For 41% (20 actions), two credible stories were recorded; for 25% (12 actions), three; and for 25% (12 actions), four. One striking aspect of A2's credible stories is that they generally are missing concrete facts regarding the action being analyzed (e.g., A2 often recorded the following credible success story for question number 1 "User knows what effect to achieve because it is part of their original task"). It is quite clear that A2 must have used the copy and paste function when recording his walkthrough. This is supported by the fact that all 37 success stories related to question number 4 are exactly identical (see the last bulleted item in Figure 6 for the story reported for question number 4).

A2 recorded seven problems in total. Four of these problems were detected while walking through his action sequences. Thus, A2 detected problems in 8% of the 49 actions in his 11 task scenarios. One of the problems detected related to a failure story for question number 1 – user goal; the remaining three problems were detected while walking through the actions required by question number 3 – user associates action with goal.

There are several strategic differences between A1 and A2 in their walkthrough processes. First A1 used pen and pencil while A2 used a word processor. Second, A1 recorded concrete stories related to the action in question, while A2 recorded abstract and general stories that could be used walking through any action. Third, although both analysts recorded around two-thirds of their walkthroughs with full stories (or references to full stories) their success and failure stories were not distributed equally. A1 either answered all four questions for a given action or recorded no credible stories for a given action, while A2 only answered few of the actions with all four questions leaving the remaining actions recorded only partly. Fourth, A1 used reasonable techniques to save himself from walking through the same actions over and over again either by referring to macros or by implicitly or explicitly referring to previous actions walked through. A2 had very few repeated actions due to his focused, function-by-function task scenarios and hence did not need such shorthand notation in his walkthrough process. A2 however used copy/paste functions in recording his walkthrough, saving him from rewriting similar credible stories. As will be discussed in section 5.5.3, these process choices clearly affected their outcomes.

5.5.2 Why did the analysts walk through the action sequences as they did?

In the walkthrough process both analysts relied on guidelines and examples from Wharton et al. (1993, 1994). The guidelines included the use of the four questions (p. 112), but in their walkthroughs both analysts were more guided by the section "Success and Failure Stories" (pp. 114-118) in which Wharton et al. give examples of credible success and failure stories. While reading Wharton et al. (1993), A1 noted, "*Ah, now they introduce this concept of 'credible and failure' stories. - Is this to address the problem of exploratory learning of the interface?*" The notes also support A2's use of his so-called cheat-sheet. He noted while reading Wharton et al. (1994), "*I keep flipping through the Wharton paper to check the criteria for success & failure. I think I need to go back and recheck things. [I have to] write a cheat-sheet with the 'common features of success'*". Both analysts chose to summarize Wharton et al.'s examples in a check-list or "cheat-sheet". A1's check-list guided him to be specific and context dependent in his walkthrough, while A2's cheat-sheet guided him to answer the four questions generically. While A1's context driven walkthrough forced him to come up with a reasonable story specific to the situation in which the action appeared, A2's generic walkthrough documentation seemingly restricted him to select one of the 11 reasons offered by Wharton et al. (1994, pp. 115-116). Thus, identical teaching materials were interpreted and subsequently used differently by the two analysts, leading to large discrepancies in problem identification.

Neither of the analysts recorded answers to all four questions for all actions in their action sequences. Certainly, it is clear from the CW literature that every action should have at least one success or failure story. However, it is unclear whether an analyst should continue a walkthrough of an action when one of the first three questions leads to a failure story. If, for example, question number 2 leads to a failure story the interface item related to the failure story for that action should be fixed, and hence continuing to walk through questions 3 and 4 may seem unproductive. On the other hand one can argue that even for the same action each question has the potential to detect completely different problems, supporting the view that all four questions should be answered irrespective of the recording of a failure story. A1 seemed to hold the latter viewpoint because in roughly 90% of the actions where A1 recorded a failure story he continued answering the remaining questions for that action. In fact, this procedure led to nine actions having more than one problem report associated with them. A2 seemed to take the former viewpoint. He only reported four failure stories in his walkthrough process, but in all four cases he skipped over the remaining questions for that given action and instead jumped to the next action in the action sequence.

Occasionally A1 recorded no stories at all for some task actions. He was aware that there was a potential problem with the mechanical recording process, noting "*Filling out the forms at each step interferes with the thread of thought when performing a task 'naturally'*". This note was written while reading Wharton et al. (1992) and was directly quoted from that paper. A1's failure to record success or failure stories might be because completing the stories interfered with his natural performance of the walkthrough. However, the data point to another explanation. First, A1's missing walkthroughs were not equally distributed over the action sequences, possibly indicating that A1 was tired of recording notes or found the process tedious. There was a sharp drop-off over time in terms of completion rates. In the first third of the action sequences for each of his three tasks, the analyst walked through the sequences leaving few actions blank (11% on average for the first third of the three task scenarios). In the second third of the action sequence, slightly more actions were left blank (26% on average for the second third of the three task scenarios). In contrast, in the last third of the action sequence many more actions were left blank (63% on average for the last third of the three task scenarios). This tendency became even more pronounced in the last task scenario. Thus, drop-offs occurred both within individual walkthroughs and across the series of walkthroughs; we believe A1 simply got tired of recording stories.

A2 also wanted to find ways to improve the speed with which he could answer the questions. For example, after creating his cheat-sheet he noted: "*With the cheat-sheet I'm moving faster and the process is more mechanical. I think that's good. Now I can look through the criteria and make a quick decision about the result of an action rather than a unique approach to every single result*". As indicated by the note and as explained earlier, A2 was heavily guided by the 11 examples of reasons for credible stories in Wharton et al. (1994). In the vast majority of cases, rather than answering a question with words specific to the task action, A2 answered the questions with words directly copied from Wharton et al.'s. However, for two of the four problems detected in the walkthrough of his actions he recorded context specific failure stories; for the two other actions leading to problem reports he copied example questions from Wharton et al. (1994), rather than answering these question with context specific failure stories. A2 also left many questions blank as he walked through his action sequences, but in success cases there was no clear pattern to the omissions. Though question number one accounted for the most of the unrecorded stories (24) this number does not differ dramatically for any of the other questions: questions number two, three and four included 17, 14, and 12 omissions respectively. A2 may have felt confident about a given walkthrough after giving only one or two credible stories rather than answering all four stories, because the examples in Wharton et al. (1994), pp. 118-122 sometimes left questions unanswered with no explanation for the omissions.

5.5.3 What was the impact of the walkthrough phase on problem detection?

From section 5.4.3 we know that setting up action sequences for identical aspects of features had no systematic impact on differences in problem detection. Moreover, we know that despite similar action sequences for some aspects of features the two evaluators detected different numbers and types of problems. One question remains: what is the impact of the walkthrough phase on the outcomes?

To offer a fair comparison of the two analysts' work regarding their walkthrough processes we have reduced the scope of interest to those aspects of features that were transformed to equivalent action sequences by both analysts and resulted in a problem detection for at least one of the analysts (six action sequences). From this reduced set of actions, six problems were detected by A1 while only one of these was detected by A2. However, A2 detected three additional problems, undetected by A1.

Looking in more detail at two examples taken from the six action sequences they had in common, Table 15 and Table 16 show actions, walkthroughs, and the problems detected by the two analysts for identical feature aspects. Both tables depict situations in which the analysts set up equivalent action sequences without detecting the same problems. In Table 15 A1 transformed the feature aspect "Add volume page" into one action. A2, on the other hand, used the same simple movements to add a volume page, but used three actions to explain the movement in contrast to A1's single action. A1 believed that the user would not associate the "Add Page" menu item with the menu "Edit", so he identified this as a credible failure story for question 3. A2 believed that the user would pull down the Edit menu because Mac experience would suggest that goal (question 1) and the user would see the menu (question 2), but he did not record a story for question 3 (or question 4 – as described earlier probably because he by default skipped the remaining questions for an action after recording a failure story). Hence, A2 did not detect this problem.

On the other hand, with the problem that A2 detected – that a user would not know to select a frame before adding a volume page – A1 and A2 seemed to have a difference of opinion about what the user would want to do. A2 detected this problem (reporting a credible failure) saying that the user would not have as a goal selecting a frame (question 1) while A1 reported a success story for this question, saying the entire complex action, including selecting a frame, was part of the original task. However, when A1 used the "original task" as evidence for the user wanting to produce the right effect (i.e., have the right goal), it is unclear whether he was referring to adding a page (the effect of his whole complex action), or having a frame selected (the effect of the first simple movement). Our bet would be the former, making this discrepancy between the analysts a granularity of action issue. In both cases, then, the analysts brought their own knowledge, skills, experience, and intuitions to bear on their answers to the questions.

	A1	A2
	Action, walkthrough, and problem	Action, walkthrough, and problem
Add volume page	**Action**: Select☐ Edit - Add Volume Page **Walkthrough:** 1. Part of original task 2. Menu visible *3. Big problem here: Why should one expect a "Add Page" operation under the "Edit" Menu. This is contrary to all experience, and no label following for inexperienced users either.* 4. New page will appear **Problem:** Add Volume Page is misplaced under the "Edit" menu	**Action:** Select a frame **Walkthrough:** *1. Will the user be trying to achieve the right effect? What would lead a user to think that they need to select a frame in order to create a page to put it on?* 2. [Left blank] 3. [Left blank] 4. [Left blank] **Problem:** To create a new page you need to create a frame first, select it and then create the page. Why can't the user create an empty page
		Action: Pull down the Edit menu **Walkthrough:** 1. User knows what effect to achieve because they have experience using a system (Mac) 2. User knows an action is available by seeing a representation of an action (Menu) 3. [Left blank] 4. [Left blank] **Problem:** (No problems detected)
		Action: Highlight the "Add Volume Page" menu item and release the mouse button **Walkthrough:** 1. [Left blank] 2. User knows that an action is available by seeing a representation of an action (menu item) 3. [Left blank] 4. [Left blank] **Problem:** (No problems detected)

Table 15. The action sequences, walkthroughs, and problems detected by the two analysts for the feature aspect "Add volume page". A1 decided to transform this feature aspect into one action, which led him to detect one problem. A2 transformed the feature aspect in a similar way, but he divided the feature aspect into three actions, which led him to detect one problem. The two problems were different. A1 recorded a success story for question 1, while A2 reported a failure story for question 1 in his first action. A2's question 3 was left blank for action 2 – the exact question that enabled A1 to detect a problem.

Drawing from a different example, Table 16 shows the analysts transforming the feature aspect "Add glossary item" into identical action sequences. A1 detected a problem ("label 'Gloss' on glossary button does not support label-following") on the second action while answering question 3; A2 reported that the user would be able to understand this label because of previous experience. Similarly, A2 reported a failure due to the blinking cursor for action 3, question 3, and A1 reported a success story credited to user experience. Clearly, these analysts offer different judgments about what the fictive user will know.

Examining all six feature aspects that the analysts had in common and transformed into equivalent action sequences, the analysts agreed on a problem only once. Each analyst failed to report a problem that the other analyst reported because a particular action/question (in the following 'action/question' refers to any one of four questions for any action) in the walkthrough was left blank (a total of four problems all missed by A2). The analysts failed to reach agreement about five problems because they reported different credible stories (two success stories by A2 and three success stories by A1). Thus, omitting answers to some of the four questions and reporting different answers to similar questions were the explanatory factor for the different outcomes for those problems that related to same feature aspects.

	A1	A2
	Action, walkthrough, and problem	Action, walkthrough, and problem
Add glos-sary item	**Action:** Select "Drosophila Melongaster" **Walkthrough:** 1. Experience with Mac 2. Experience 3. Experience 4. "Drosophila Melongaster" will become highlighted **Problem:** (No problems detected)	**Action:** Select some text; a word or phrase **Walkthrough:** 1. [Left blank] 2. User knows an action is available by experience *3.* User knows an action is appropriate by experience 4. User knows things are going OK after an action by recognizing a connection between a system response and what they were trying to do **Problem**: (No problems detected)
	Action: Click on "Gloss" button **Walkthrough:** 1. Part of task & system experience 2. Button visible *3. Again big problem - why abbreviate & distort "Glossary" to Gloss* 4. Item will appear in glossary pane **Problem:** Glossary button "Gloss" doesn't support label-following by user	**Action:** Click on the glossary icon **Walkthrough:** 1. User knows what effect to achieve because it is a part of their original task 2. User knows an action is available by seeing a device (icon) 3. User knows an action is appropriate by experience 4. User knows things are going OK after an action by recognizing a connection between a system response and what they were trying to do **Problem**: (No problems detected)
	Action: Type in text for "Drosophila Melongaster" **Walkthrough:** 1. Part of original task 2. Insertion point visible & experience 3. Experience 4. Text will appear in glossary pane **Problem:** (No problems detected)	**Action:** Type the definition of the text for the glossary **Walkthrough:** 1. [Left blank] 2. [Left blank] *3. Will the user know an action if the action is appropriate? There is no feedback from the system other than the blinking insertion point* 4. [Left blank] **Problem:** No explicitly feedback is given after adding a glossary entry [Note: A2 is referring to the blinking cursor in the glossary frame being the only clue that the next step is to type in the definition at the beginning of this action. He is not referring to the feedback *after* the typing-in action, which is that the text will appear in the glossary pane, just as A1 described it.]

Table 16. The action sequences, walkthroughs and problems detected by the two analysts for the feature aspect "Add glossary item". Both analysts transformed this feature aspect into identical actions. A1 detected a problem on his second action and associated this detection with question 3 (shown in italics); A2 reported a success story for this particular action/question. A2, however, reported a problem with the third action while answering question 3 (shown in italics); for this particular action/question A1 reported a success story.

The substantial deviation between the two analysts' problem identifications indicates that the walkthrough process itself, i.e., answering the questions, has a strong effect on the final result of a CW. Our two analysts who were evaluating the same feature aspects with the same actions using the same questions identified only a single problem. Thus, roughly a fifth of the total number of problems detected by the analysts (including those not found directly while answering the walkthrough question) were not detected by both analysts because they differed in their walkthrough processes.

5.6 Summary of the differences between A1 and A2

In an ideal world, a usability evaluation method should guide evaluators to reliably detect usability problems. Specifically, the CW – a process more structured than some other evaluation methods – had the potential of being more reliable than other UEMs like heuristic evaluation and think-aloud usability tests. Our detailed analyses of the process of learning and using the CW for two analysts show that the ideal world is far from the real world. Despite honest and persistent effort from the analysts – an effort in which direct errors or misunderstandings have not been observed – the substantial differences between the two analysts speak for themselves. Not only have we seen that the outcome from the two analysts' evaluations differ dramatically, but it is evident that every stage in the CW opens up possibilities for the analysts to diverge from the seemingly structured process.

A1 detected 46 problems of which 20 problems (43%) were not associated with any specific action/question in his walkthrough; 18 problems (43%) were detected by A1 but not A2 because A1 covered the interface more thoroughly; and 8 problems (19%) were detected only by A1 because A1's walkthrough differed from that of A2. A2 detected 7 problems of which 3 problems (43%) were not associated with any specific action/question in his walkthrough; 4 problems (43%) were only detected by A2 because A1 recorded credible success stories when A2 recorded failure stories; 1 problem (14%) was only detected by A2 because A1 did not record a credible story for that action.

A striking qualitative difference between the two walkthroughs is that A1 wrote Builder-specific success and failure stories and A2 primarily copied Wharton et al.'s words from the 11 credible story examples presented in the 1994 version. A2's procedure for recording general credible stories might have caused him to miss identifying problems, while A1's process of recording context specific credible stories might have caused him to identify more problems, since this procedure forced A1 to think more carefully about the interaction between user and system.

Both analysts' user descriptions for the fictive user were similar in form and content (following the suggestions by Wharton et al., 1993 & 1994), even to the extent of being rather underspecified and insufficient as inputs to the walkthrough process. The analysts often differed in what they assumed the user would know. A1 assumed the user would not associate "Gloss" with the goal to "add a glossary entry" while A2 assumed the user would. A1 assumed the user would not associate the "Edit" menu with the goal to "add a volume page" and A2 assumed the user would. An underspecified description of the user may cause the analyst compensate by anchoring his own experience to the process of answering questions (see Hertzum & Jacobsen, 1999; see also Tversky and Kahneman, 1974, who have described the anchoring effect in other contexts). More precisely, when the analyst is in the midst of the walkthrough process and lacks sufficient detail about the fictive user to answer one of the four questions, he may think of his own likely goals and behaviors in the situation in question. If the analyst believes that he would have a problem related to the four questions he may report this problem as a failure story. Similarly, if the analyst believes that he is likely to have the right goal, notice the proper effect, make a correct association, and retrieve sufficient feedback, he may report a success story.

The anchoring hypothesis is to some extent supported by data in the database. Of the 26 problems associated with A1's actual walkthrough process 6 problems were described in notes as problematic well before being reported in the walkthrough. For example, four hours before A1 walked through an action testing cascades of undo commands he wrote the following note "*Assume: Undo command refers to the last key click/mouse click. How to undo sequences of actions? -> Not possible*". Four hours later, in the walkthrough of the undo action he reported the following problem: "*Undo of sequences of actions: Not possible, only last mouse/key click*". Twelve hours into the evaluation process A2 wrote one note identifying a problem with poor feedback when adding a glossary entry; 15 hours later he reported the same problem, this time associated with the actual walkthrough process. We believe this foreshadowing of predicted usability problems may indicate a personal bias that finds expression when the opportunity arises in the walkthrough process, rather than a problem derived from the CW process itself.

6 Why the CW results did not predict usability test results

In the results section (section 4) we revealed how the CW analysts' problem predictions differed substantially from the problems observed in usability tests. In this section our aim is to analyze the matches, false alarms and the large number of misses in the CW analysts' problem reports and to compare them to the results from the usability tests. As mentioned earlier, this analysis is not a comparison between the two techniques in order to determine which technique is best at detecting usability problems. Rather, we want to investigate the predictive power of CW conducted in the specification stage of design, when measured against the problems observed in a usability test conducted after a running prototype has been implemented. CW has been advocated as a technique that is especially valuable before a running version exists (Lewis & Wharton, 1997, p. 718), while usability tests are often conducted late in the design cycle (see for example Gardner, 1999). It is obviously not reasonable to compare the benefits of each technique when the techniques are designed for use in different stages of the design cycle. Finding a problem early in the design cycle can save significantly more effort than finding the same problem late in the design cycle; hence, a real comparison is inappropriate when measuring CW problem predictions against usability test observations. However, in order to improve CW in the future, it is valuable to assess its predictive power, to understand its relative strengths and to identify its weaknesses.

In our case, A1 precisely predicted three problems and vaguely predicted three problems that were later observed, predicted two problems that were false alarms, and missed 45 problems observed in the usability tests. In contrast, A2 did not predict any problems that were observed in the usability tests, predicted one problem that was a false alarm, and missed 52 problems that were observed in the usability tests. In the sections below most effort has been devoted to explaining the large number of misses due to the greater proportion of problems of this type. But we first examine the predicted problems that were either matched to observed problems or explicitly labeled as false alarms.

6.1 The CW analysts' matches

The problems precisely and vaguely predicted by A1 are identified in Table 17.

Analyst	Prediction type	Problem description
A1	Predicted precisely	User will not find the most efficient command "GoTo page" under the right menu
A1	Predicted precisely	It is not MAC standard to keep pressing mouse button down when creating a new object (here a frame) and resizing object. This should be done in two steps. Creating and resizing.
A1	Predicted precisely	Rename menu "Windows" to "Show" (label following!) [Menu item Window is not appropriate for its content]
A1	Predicted vaguely	User will be confused what actions to find in button palette and what in the (augmented) standard MAC menu.
A1	Predicted vaguely	Save Volume View document dialog box is confusing. Text: "Execute Only" is not necessary.
A1	Predicted vaguely	Save Volume View document dialog box is confusing. Text: "Text Only" is not necessary.

Table 17. Problems predicted by A1 and A2 that were either precisely or vaguely predicted, all compared to the problems observed in usability tests. (None of A2's predicted problems were observed in the usability tests).

These correctly predicted problems can be characterized as minor problems in three respects. First, none of the six correctly predicted problems caused major annoyance for users in the usability tests. This judgment about problem severity was based on reports from the usability test evaluators. As described earlier the usability test evaluators had to report which of nine predefined usability criteria were used for each problem they reported (see section 3.2 for the full list of usability criteria). After all usability test sessions were analyzed we defined three of these nine criteria as *severe* because the interface problem prevented the user from completing the task: (1) the user articulates a goal and cannot succeed in attaining it within three minutes, (2) the user explicitly gives up, and (3) the system crashes. (These criteria were also used to judge severity in Jacobsen et al., 1998). If an evaluator reported that a problem violated any one of these three criteria then we defined the problem as severe. None of the problems predicted by the CW analysts were reported as violating any of the three criteria.

Another way to characterize the predicted problems in terms of severity is to see to what extent the interface would have to be changed in order to correct or eliminate the problems. The three problems related to menu placement, menu naming and consistency between menu content and the tool bar did not essentially change the interface. Rather, these fixes could be easily understood and immediately corrected in most current programming environments. The two problems related to check-boxes in a save dialog box could also be easily fixed by renaming that check-box labels and perhaps moving the check-boxes to within the dialog box. The remaining problem concerns a violation of a Macintosh standard. When a user creates a frame, the procedure is to click on a frame button in the tool bar, move the cursor to the canvas, and then click and hold down the mouse button until the frame is of an appropriate size. The CW evaluator suggests that the user should be able to create a frame by clicking once on the tool bar button followed by one click on the canvas. Then a resizing mechanism should enable the user to resize the frame to an appropriate size. Fixing this problem does not essentially change the application, and the fix itself is quite easy.

Early problem detection has been promoted as highly cost-beneficial. Therefore CW has been especially promising as it claims applicability early in the development cycle. If it is possible for evaluators to find serious design problems through the use of the CW technique, these problems will not be implemented, thus reducing overall development process costs. However, in our case we have found that problems that were correctly predicted by the CW analysts were not severe when measured against user breakdown situations in usability tests. In fact, they did not dramatically change the interface compared to the overall functionality of the application under evaluation. In the end they were not of a problem type that could have reduced the overall costs of the development project. In fact the problems could easily have been detected and fixed late in the process when a running prototype was available.

6.2 The CW analysts' false alarms

In section 4.2.3, we defined a false alarm as a problem predicted by a CW analyst but which, when confronted by the exact situation in the running system, does not present a problem for users when performing their task. The actual concept of a false alarm is controversial in HCI. One definition of a usability problem is a tautology: a usability problem exists if an evaluator detects a usability problem using any method. If we followed this definition, we should claim that problems not confirmed in a usability test were usability problems that had failed to be detected through usability tests (and hence, not false alarms). That is, we might simply suspect that we had not included a sufficient number of users and evaluators in the usability tests to actually validate that the CW-predicted problems were indeed problems for users. Although we admit that we do not have the philosopher's stone on this issue we have chosen to label the problems not validated by usability tests as false alarms because we believe that the tautological definition of a usability problem is unmanageably broad and pragmatically flawed. Why expend effort to fix a "problem" that a reasonable number of users have been unable to detect when the number of usability problems detected often outstrips the resources available to fix them?[6] Furthermore, "fixing" any problem

[6] We are aware that the usability test method has not been validated in real world settings. Hence, our way of comparing CW predictions to observed problems in a usability test is limited to the fact that the usability test itself is an experimental setting that should not bear comparison to real users in real world settings. However, as long as we are short of validation studies of usability tests we believe that the technique that most closely depicts what users might experience when using a product for the first time is the usability test.

runs the risk of introducing new usability problems, which may be even more serious than wasting time detecting and fixing a false alarm.

The three problems included as false alarms are shown in Table 18. In each of these cases, when the precise situation arose for the users in the usability test none of the users behaved in a way that caused an evaluator to record a PDR.

Analyst	Prediction type	Problem description
A1	False alarm	When copying a PICT to the clipboard both from Builder and from another application, no particular feedback is given in the menu of Edit. Hence the user doesn't know what kind of item exists in the clipboard at a given time
A1	False alarm	The cursor used on code frames is a cross. Why not the regular text editing mouse point over code (|).
A2	False alarm	No explicit feedback is given after adding a glossary entry

Table 18. Problems predicted by A1 and A2 that were explicitly found to be false alarms, i.e. none of the problems were detected in usability tests even though several users used the functions in question.

On a quantitative scale A1 literally went two steps forward and one step backward, predicting six usability problems correctly and then predicting three problems that did not trouble any of the users. A2 did even more poorly as he did not predict any problems correctly, but predicted one problem that was a false alarm. But qualitatively how serious were these false alarms? As with the analysts' correctly predicted problems, their false alarms seem to be of minor importance. Obviously we do not have evidence from the usability tests to measure the severity of these false alarms, as they were never detected. However, one measure of the importance of false alarms is to imagine how developers would fix them. The first false alarm (regarding lack of feedback after using a copy function) can easily be fixed; the repair situation is part of the problem description as the evaluator suggests that the menu item "paste" should contain an indication of which type of item is in the clipboard. If, for example, the user has copied a picture, the paste menu label should read "paste picture", rather than just "paste". This change does not seem to bring about new usability problems and the change is minor. The second false alarm concerns an alleged lack of cursor change when the cursor enters a programming code frame. An editor cursor (|) is more logical than a cross cursor (+) in a programming code frame, as was suggested by the evaluator. The problem is partly a matter of taste and partly a violation of a standard; it is easy to fix and the change will most likely not disturb the user at all. Lastly, one false alarm regards lack of feedback when adding a glossary item. Changing this aspect of the glossary function is easy and is a minor change, e.g. by causing a dialog box to appear after the user has added a glossary item.

Overall the false alarms are of minor importance, and if the interface had been changed because of these reported "problems" this would not have taken up much project development time. Moreover, we do not see changes in the interface bringing about new usability problems. The false alarms resemble the correctly predicted problems insofar as they concern superficial aspects of the interface and so will cause neither huge improvement nor dramatic deterioration.

6.3 The CW analysts' misses

In this section we will explain why in their action sequence walkthroughs the CW analysts missed a great number of problems (in comparison with problems identified in the usability test). A1 missed 45 problems and A2 missed 52 problems that were observed in the usability tests (see also section 4.2.3 on page 108). Based on a binary decision tree (see Figure 5) we analyzed each of the usability problem descriptions and correlated them to the analysts' actions and walkthroughs. More precisely we examined each missed problem and judged whether each analyst had set up an action that could have revealed the missed problem. We then continued to refine our analysis by walking through the decision tree until we found a reasonable explanation for why the analyst missed the problem in question. The results from this analysis can be seen in Table 19.

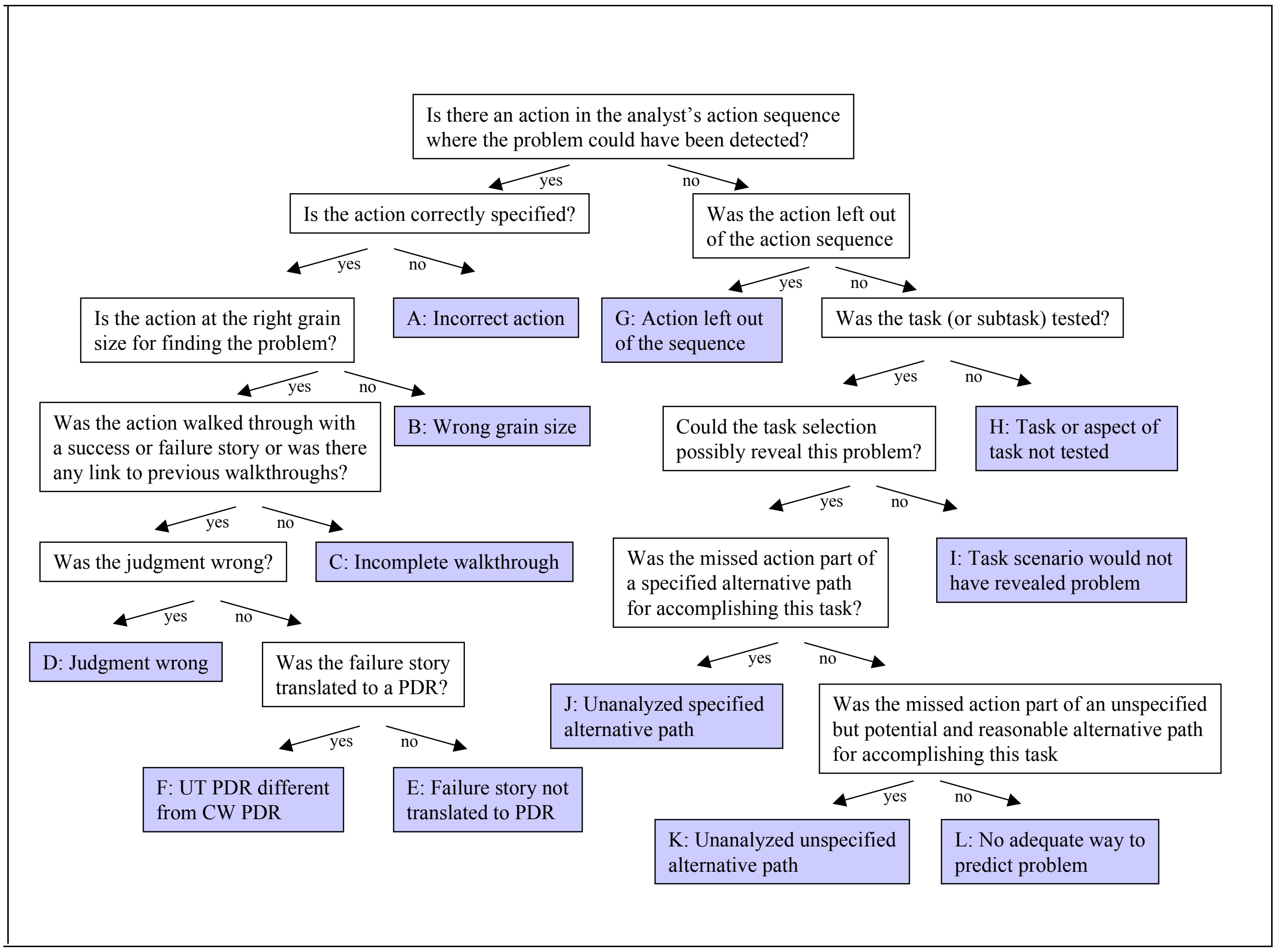

Is there an action in the analyst's action sequence where the problem could have been detected?
yes
no
Is the action correctly specified?
Was the action left out of the action sequence
yes
no
Is the action at the right grain size for finding the problem?
A: Incorrect action
G: Action left out of the sequence
Was the task (or subtask) tested?
Was the action walked through with a success or failure story or was there any link to previous walkthroughs?
B: Wrong grain size
Could the task selection possibly reveal this problem?
H: Task or aspect of task not tested
Was the judgment wrong?
C: Incomplete walkthrough
Was the missed action part of a specified alternative path for accomplishing this task?
I: Task scenario would not have revealed problem
D: Judgment wrong
Was the failure story translated to a PDR?
J: Unanalyzed specified alternative path
Was the missed action part of an unspecified but potential and reasonable alternative path for accomplishing this task
F: UT PDR different from CW PDR
E: Failure story not translated to PDR
K: Unanalyzed unspecified alternative path
L: No adequate way to predict problem

Figure 7. The decision tree from which the analysts' missed problems were categorized. Leaves B, E, J, and L in the tree could explain none of the misses so they are not included in the result table below. (UT = usability test).

	Walkthrough and action existed for this problem				No walkthrough and action existed for this problem				Total
	C: Incomplete walkthrough	F: UT PDR different from CW PDR	D: Judgment wrong	A: Incorrect action	H: Task or aspect of task not tested	I: Task scenario would not have revealed problem	G: Action left out of the sequence	K: Unanalyzed unspecified alternative path	
A1	25 21.9%	33 28.9%	22 19.3%	4 3.5%	20 17.5%	7 6.1%	2 1.8%	1 0.9%	114 100%
A2	7 13.2%	2 3.8%	5 9.4%	0 0%	24 45.3%	10 18.9%	3 5.7%	2 3.8%	53 100%
A1 and A2	17.6%	16.4%	14.4%	1.8%	31.4%	12.5%	3.8%	2.4%	100%

Table 19. Actions related to observed problems missed by A1 and A2 distributed according to the decision tree. The first four columns consist of those missed problems that could be related to a walkthrough (totaling 50.8%). The last four columns consist of those missed problems that cannot be related to a certain action or walkthrough (the remaining 49.2%).

For the missed problems that were not related to an action (right half of the tree and table) there is a one-to-one mapping between observed problems and explanations. Since actions could be repeated in action sequences, some missed problems associated with a walkthrough (left half of the tree and table) were categorized by more than one explanation. For example, one missed problem concerned locking a frame to prevent the user from editing the text in the frame. Based on the usability tests one evaluator reported that the lock/unlock functions were not obvious to the user. A2 tested both lock and unlock functions. Hence, he had two opportunities to predict the problem observed in the usability tests. Both occurrences appear in Table 19 as incomplete walkthroughs because although A2 set up actions appropriate to detect the problem, he wrote no explicit success or failure story for question 1, but judged the action to be a success.

Roughly 75% of A1's and 25% of A2's missed problems were related to actions found in their action sequences (left half of the tree and table). Conversely, roughly 25% of A1's and 75% of A2's missed problems can be explained by incomplete task selection or inappropriate transformation of task scenarios to action sequences (right half of tree and table). Since coverage of feature aspects varied enormously between the two analysts (discussed in detail in section 5.3.2 on page 114) it is no surprise that A1 had the potential to detect, and actually did detect, more problems than A2.

It is more surprising to notice that A1, despite his thorough coverage of the interface, missed so many problems. For those problems that A1 could have detected given his task selection, more than one fifth can be explained by omissions in the recording of his walkthrough process (C: incomplete walkthrough). In contrast, only 13% of A2's missed problems were attributed to incomplete walkthroughs. This probably reflects the difference we noted in section 5.5.1 when we examined the patterns of missing stories. All of A2's actions had at least one story associated with them, while A1 left 15% of his actions totally blank. Of A1's missed problems caused by omissions in the recording of his walkthrough process three problems could potentially have been detected if he had just walked through that action, rather than leaving the walkthrough totally blank.

Another 19% of A1's and 9% of A2's missed problems can be explained by incorrect judgment (category D), i.e. recording success stories when observed users actually had problems completing the action in question. We know that one user is not enough when evaluating an interface with usability tests (Virzi, 1992; Lewis, 1994) and even that one evaluator is not enough (Jacobsen et al., 1998). Given these results, in retrospect it is difficult to believe that one CW evaluator analyzing a user interface with only one user description (no matter how complete) would be sufficient to generate a trustworthy evaluation. In addition, we believe an underspecified user description, like the ones these analysts used, will lead an analyst to insert his own introspections into the role of the fictive user. Thus, the situation may be even worse than a single-user single-evaluator usability test because the fictive user and the evaluator are essentially the same person.

For 35 missed problems (33 for A1 and 2 for A2, category F), the CW analyst had actually detected a problem, but that problem differed in content from the observed problem. One example of a predicted problem that differed from an observed problem regards the action "Select Frame Edit - Add Volume Page" and the third question "Will the user associate the correct action with the effect trying to be achieved?". A1 reported a failure story writing "No user will find 'Add Volume Page' under the 'Edit' menu". The observed problem regards a user who is annoyed about not having a shortcut key for adding a volume page. For some reason CW analysts tend to behave as if there can be no more than one problem detected for each action/question. However, the data in this study belie this assertion; often different usability problems were associated with a single action/question. This is also supported by the studies of Hertzum & Jacobsen (1999) in which different CW analysts reported different problems for the same action/question.

A1 and A2 respectively missed 7 and 10 problems that would have been difficult to detect even if they had set up task scenarios that covered these aspects of observed features (category I). The explanation for this is that some of the problems observed in the usability tests were of such general character that they would not match any single action/question related to the walkthrough process. One example of a general problem description was "*The system does not support all Mac shortcut conventions*". This problem was observed by three usability test evaluators analyzing a video on a certain proficient Macintosh user. The CW analysts might have set up actions that could reveal that a *certain* Macintosh shortcut was not implemented in the system, but generalizing this problem to lack of shortcuts for more functions in the system would not be part of the procedure in the CW technique. Of course the CW evaluators could have stepped back from the detailed walkthrough process, but the evidence suggests that they did not generalize many of the specific problems. A1 notes that "*Because of the focus on individual user actions there is the danger of overlooking interface problems concerning the task as a whole*". And, in spite of one analyst's awareness of this particular danger, this is exactly what has happened for both analysts[7].

In summary, 92% of the missed problems are attributable to five causes. Almost a third of the problems missed by the analysts could not have been detected because the analysts did not select a suite of task scenarios that covered the interface appropriately (category H). Eighteen percent resulted from incomplete walkthroughs, where analysts failed to record stories (category C). Sixteen percent resulted when problems in the usability tests differed from the predicted problems (category F), indicating that more than one problem arose at given action/question. Fourteen percent were caused by incorrect judgments about what a user would know (category D). More than 12% of the explanation for missing problems can be found in the difficulty of generalizing problems (category I). The remaining 8% can be blamed on the transformation from tasks to action sequences: listing an incorrect action in an action sequence, leaving an action out of a sequence, or neglecting to analyze an alternative path.

7 Conclusions

An ideal UEM should be easy to learn and quick to use; the evaluation it produces should be reliable and accurate. CW fairs well on these first two attributes, but falls far short of the ideal with respect to the last two. Although our data are limited by the background of the analysts, the type of application and specification documents, and the specific procedures used in the usability tests (see also John & Mashyna, 1997, for a more detailed discussion of the limitations of A1's case), the detailed process data provided by these two cases can lead to recommendations for improving the technique. We present our recommendations and then discuss what sort of tool support would help streamline the enhanced CW process.

7.1 Recommendations for changes in CW procedures

CW is easy to learn – but we recommend reading more than just Wharton et al. (1994). Both analysts spent roughly the same amount of time learning to use CW (ranging from 7 to 11 hours – with A1's time counted from the point in his learning process where he chose to apply CW to the Builder). However, A1 displayed much more concern about issues like the selection of tasks and the problems that CW is known to miss, which were brought up in papers that A1 read but A2 did not. Therefore, we reject the hypothesis that reading only Wharton et al. (1994) as teaching material for learning to use CW is sufficient. (Perhaps the newer Wharton et. al 1997 solves some of these problems, but we have no data to make a judgement on that point.)

[7] The problem of CW missing general aspects of an interface has been reported earlier by Wharton et al. (1992).

CW is easy to use, but tedious. The analysts complained about the tedium of doing the analysis, and they failed to record a number of the answers to the walkthrough, perhaps because of its repetitiveness. A1 devised macros to help relieve this burden; A2 used copy-and-paste of generic success stories. In the next section we will propose functional specifications for a tool for CW which will address this issue and others.

CW was not accurate for our analysts compared to the usability tests. Of the 52 problems observed in the usability tests that could have been predicted by the CW analysts, A1 missed 45 and A2 missed all 52 problems. Based on the process data in our case study, we hypothesize that three procedural changes to the CW technique would increase its predictive power.

1. *Analysts should deliberately select task scenarios that cover the functionality of the system.* Our case studies revealed that selection of appropriate task scenarios was a necessary but not sufficient requirement for accurately predicting usability test results. Because A2 covered so little of the Builder's feature aspects he had no opportunity to identify 45% of the problems observed in usability tests. Although A1 covered roughly three times more of the system's feature aspects he still missed a large number of problems observed in usability tests (demonstrating the insufficiency of good task selection). Because A1's large task scenarios based on a real task provided him better coverage than A2's specification-based approach, we suggest that analysts first chose realistic task scenarios, e.g. based on user observations, (in line with Lewis & Wharton, 1997). After this initial task selection, the analysts should identify all features and feature aspects available in the system using the specification (see Table 9 page 116 for examples of features and feature aspects), and explicitly judge which feature aspects are included in the selected task scenario. If the coverage based on realistic task scenarios seems unsatisfactory, the analyst could modify or extend the task scenarios to include more feature aspects or simply add more scenarios. This procedure combines the coverage and face-validity of realistic tasks with systematic coverage available from a specification. Of course, this step will take more time than simply making up a few task scenarios, but we believe that low coverage is a major problem with CW (and other UEMs as well, e.g., our first set of usability tests).
2. *The fictive user description should describe sets of users, rather than an individual user.* To overcome the effect of individual differences empirical studies are typically conducted with more users. The CW technique, however, is typically based on a description of *one* user. Our analysts missed 30% of the observed problems because they reported a failure story different from the one observed or because they reported a success story for a question on a given action that was observed to be a problem by at least one of the users in the usability tests. Perhaps if the description of the fictive user consisted of a set of users rather than an individual user, the analysts would consider several points of view and uncover problems more accurately. This recommendation is in line with the procedure called PAVE (Programmed Amplification of Valuable Experts) explored at NYNEX in the early 1990s (Desurvire & Thomas, 1993). To do this, each question for a given action would be walked through as many times as there are defined fictive users. On the positive side, the potential of the recommendation is that more problems might be identified and also that more problems might be identified for the very same question/action. Negatively, the recommendation increases the risk of tedium of the CW technique (but see the next section for how tool-support might help).
3. *A generalization step should be added to the CW technique.* A third problem with the accuracy of CW is that it is more successful identifying specific rather than general problems (Jeffries et

al., 1991; Wharton et al., 1992). Our data support this critique. One-eighth of the problems missed by the analysts where attributed to CW not finding general problems in an interface. Therefore we recommend that evaluators reflect on each problem, either after an individual problem has been identified or after completing the full walkthrough, with respect to whether the problem generalizes to other parts of the system. Further, the analyst should offer a judgement about the appropriate level of abstraction at which to describe a given problem. For example, the most concrete level describes details about the interface (e.g., a menu item name, a button label), which can be combined into a lesser number of feature aspects (e.g., adding a cross reference or navigating forward using the GoTo function), which again can be combined into a lesser number of general features (e.g., cross reference, navigation, and save). It is advisable to describe a problem at as abstract a level possible in order to prevent fixing local problems while ignoring global ones. Note, however, that the recommendation of generalizing problems should not be misunderstood as a recommendation to report problem *types* rather than problem *instances*. An example of a problem type is "buttons in the tool palette are difficult to use", while an example of a problem instance is "buttons in the tool palette do not provide any feedback that they have been clicked (i.e., they don't highlight or indent)". This problem instance could have been generalized from a specific problem description: "The Glossary button in the tool palette does not highlight after its been clicked". Problem types are not specific enough for designers to fix the problem, whereas problem instances are (John & Mashyna, 1997).

CW was not reliable when comparing our analysts to each other. One significant reason for disagreement between the two analysts was their different coverage of the feature aspects of the Builder (which is addressed in recommendation 1, above). However, even when they walked through the same task using the same sequence of actions, they differed on their success and failure stories. We hypothesize, as do Hertzum & Jacobsen from their data (1999), that individual evaluators anchor their answers to the four questions to their own experience rather than to the description of the fictive user. We recommend two changes to CW to address this problem.

4. *The description of the fictive user should be more extensive than what is suggested in Wharton et al. (1994).* It should be more specific with regards to the fictive users' domain experiences, operating system experiences, and experiences with other similar systems. When user descriptions become as detailed as characters in a novel, the characters tend to be separated from ourselves – they become another person – which might reduce the anchoring problem. The previous recommendation to include several users in a fictive user set further reduces the anchoring effect because not all of the users in the set can act as the analyst him- or herself would act – it again takes the analyst outside him- or herself.
5. *Evaluators with different backgrounds should walk through action sequences in groups.* Other reports have recommended that CW is best done in groups (Karat et al., 1992; Karat, 1994; Rowley & Rhoades, 1992) and our cases support this wisdom. A group process holds assumptions up to scrutiny that an individual process misses. It is likely to counter-act the anchoring effect when different people's opinions are voiced and discussed.

Having recommended these procedural changes, we now offer further recommendations for tool development to support these procedures.

7.2 Recommended tool support

Just as interface builders have made the coding of prototypes, and even shippable systems, easier and faster to do, we propose that a computer-based tool would support CW analysts in their work. We are not the first to suggest a tool for performing CW. In fact, a CW tool was implemented in HyperCard very early on (Rieman et al., 1991). This tool, however, was built for the first version of CW, which differed from the version described by Wharton et al. (1994) in many respects (e.g., by having many more questions to be answered for each action in an action sequence), and to our knowledge updates to the tool have not kept pace with updates to the technique. Our data suggest several requirements for a CW tool, which we believe, may help alleviate the tedium and improve the accuracy and reliability of the technique.

All preparatory materials should be integrated into the tool. For example, there should be means to record the initial realistic task scenarios and feature and aspects of feature of the system. It should support bookkeeping, like allowing the analysts to check-off the aspect of features that each task scenario exercises, and highlight any gaps. Multiple fictive user descriptions should be included, with room for extensive definitions. All such information should be displayable on command, editable at any time during the walkthrough (e.g., to record additional detail about a fictive user), and will be used by the tool as needed in its support of the analysts.

Answering the CW questions should be mandatory but very easy to do. The two analysts in our cases were not accurate in their walkthrough process partly because they failed to record credible stories for 35-40% of the questions. A1 seemed to be affected by exhaustion, particularly towards the end of walking through his long action sequences, while A2 more or less randomly only recorded one, two, or three credible stories for each action. To avoid missed questions, a CW tool should guide an analyst through each step, asking each question for each step. To make this process fast, it should allow analysts to pick standardized general success and failure stories from a list (as used by A2 in his walkthroughs), which also makes credible story types easy to remember. However, it should also prompt analysts to add specific information about their interface to each credible story. If multiple fictive users have been defined in the preparation phase the system should ask the same question for *each* fictive user (perhaps by displaying the user description at the same time that the credible story for that particular fictive user is recorded).

When realistic task scenarios are transformed into action sequences some actions are often continuously repeated (e.g., "create frame" was repeated in the scenarios for testing the Builder). Our evidence suggests that not only were some of these repeated actions annoying to the analyst but they also had the potential to be annoying to real users of the system. A tool to support CW should identify repeated actions and prompt the analyst to judge whether a repeated action is likely to be unproductive and bothersome for the user of the application at this point (i.e., the tool ought to suggest the reporting of a problem based on a repeated action). If the analyst judges a repeated action to be unproblematic for the user, the tool should ask if the walkthrough of this particular action should refer back to the previous walkthrough thereby saving the analyst time in the walkthrough phase.

The tool should provide support for generalizing problems. As reported above, one eighth of the problems missed by the CW analysts were attributable to under-generalization of specific problems. To improve identification of general problems, each time a failure story is been recorded the tool could prompt the analyst to consider generalizing the problem. This could be a simple question about whether the problems applies to other and more general parts of the system, together with a cross-referencing capability to link several related problems to each other and their generalizations. In an advanced version, the tool might recognize words (like "button", "menu", etc.) in the problem description and offer more precise suggestions like "should this problem be generalized to all buttons in the button palette?".

A CW tool might be integrated into a prototyping tool. Our data provide some evidence that task transformation from high-level task scenarios to detailed action sequences was difficult especially when based on a written specification. One in 12 problems that could have been detected by both analysts were not detected because of differences in granularity in the task transformation, and another 8% of the problems observed in usability tests were not detected by the analysts because they listed incorrect action sequences, left actions out of sequences or neglected to analyze an alternative path. An automated transformation of task scenarios to action sequences would increase the reliability and accuracy of CW. This argument has also been made for another UEM (GOMS), where data show that novice analysts leave out whole steps in their action sequences (John, 1994). To alleviate that problem in GOMS modeling, Hudson et al. (1999) have recently integrated GOMS-modeling into the Subartic programming environment in a tool called CRITIQUE (Convenient, Rapid, Interactive Tool for Integrating Quick Usability Evaluation). The analyst need only demonstrate a task using a prototype built in Subartic and the actions are recorded for further analysis. CRITIQUE currently produces a Keystroke-Level Model of the task automatically, but it is easy to see how such a recording could be used for several UEMs. For instance, it could present the steps for CW, guaranteeing that they are sufficient to accomplish the task (or the demo wouldn't work). It could also alert the analyst if areas of functionality have not been exercised in any task scenario, helping the coverage problem.

CW is a fairly young usability evaluation method. The developers of the method have already modified CW at least three times based on their experience using and teaching CW (the first version was described in Lewis et al. (1990), the second version in Polson et al. (1992), and the third version in Wharton et al. (1994)). While the most recent paper (Lewis & Wharton, 1997) does offer some new recommendations, fundamentally the technique has not changed since the third version. Our recommendations for change, both procedural and for tool-support, come from careful study of detailed process data which we hope will inspired CW developers and users to improve the technique.

Acknowledgments

The Danish Research Councils (DRC) sponsored Niels Ebbe Jacobsen's participation in this research. Bonnie John's participation in this research was sponsored by the National Science Foundation (NSF), Award #IRI-9457628. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DRC, NSF, or the U.S. Government.

References

Bell, B., Rieman, J., & Lewis, C. (1991). Usability Testing of a Graphical Programming System: Things We Missed in a Programming Walkthrough. In S. P. Robertson, G. M. Olson, & J. S. Olson (Eds.), Proceedings of CHI'91: New York: ACM.

Butler, K, Jacob, R. J. K., & John, B. E. (1992-1999). Introduction and Overview of Human-Computer Interaction. Tutorial Materials, presented at CHIs, 1992-1999 ACM, New York.

Card, S.K., Moran, T.P., & Newell, A. (1980). The keystroke-level model for user performance time with interactive systems. Communications of the ACM, 23(7) (July), 396-410.

Carroll, J.M., & Rosson, M.B. Human-Computer Interaction Scenarios as a Design Representation. (1989). T.J. Watson Research Center, Yorktown Heights, NY 10598: IBM Research Division. IBMC 15033.

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

Desurvire, H.W. (1994). Faster, Cheaper!! Are Usability Inspection Methods as Effective as Empirical Testing? In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods. (pp. 173-201). Wiley.

Desurvire, H.W., & Thomas, J. C. (1993). Enhancing the performance of interface evaluators using non-empirical usability methods. In Proceedings of the Human Factors and Ergonomic Society 37th Annual Meeting (Seattle, WA, October 11-15), 1132-1136.

Dillon, A., Sweeney, M., & Maguire, M. (1993). A Survey of Usability Engineering Within the European IT Industry - Current Practise and Needs. In J. L. Alty, D. Diaper, & S. Guest (Eds.), People and Computers VIII: Cambridge: Cambridge University Press.

Gallagher, S., & Meter, S. (1993). Volume View Interface Design. School of Computer Science, Carnegie Mellon University.

Gardner, J. (1999). Strengthening the focus on users' working practices; Getting beyond traditional usability testing. Communications of the ACM, 42(5), 79-82.

Hammond, N., Hinton, G., Barnard, P., MacLean, A., Long, J., & Whitefield, A. (1984). Evaluating the interface of a document processor: A comparison of expert judgment and user observation. In B. Shackel (Ed.), Human-Computer Interaction - INTERACT'84: North-Holland: Elsevier Science Publishers B.V.

Helander, M.G, Landauer, T.K., & Prabhu, P. (1997). Handbook of Human-Computer Interaction, (Second Edition ed.). Elsevier Science Ltd.

Hertzum, M., & Jacobsen, N.E. (1999). The Evaluator Effect during First-Time Use of the Cognitive Walkthrough Technique. To appear in Proceedings of the Eight International Conference on Human-Computer Interaction, (HCI International ´99), Munich, Germany, August 22-27, 1999.

Howes, A.,& Young, R.M. (1991). Predicting the Learnability of Task-Action Mappings. In S. P. Robertson, G. M. Olson, & J. S. Olson (Eds.), Proceedings of CHI'91: New York: ACM.

Hudson, S.E, John, B.E., Knudsen, K., & Byrne, M.D. (1999). A Tool for Creating Predictive Performance Models from User Interface Demonstrations. To appear in UIST'99.

Jacobsen, N.E., Hertzum, M., & John, B.E. (1998). The evaluator effect in usability studies: problem detection and severity judgments. In Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting: Santa Monica, CA: Human Factors and Ergonomics Society.

Jeffries, R., Miller, J.R., Wharton, C., & Uyeda, K.M. (1991). User Interface Evaluation in the Real World: A Comparison of Four Techniques. In S. P. Robertson, G. M. Olson, & J. S. Olson (Eds.), Human Factors in Computing Systems CHI'91: New York: ACM.

John, B. E. (1994). Toward a deeper comparison of methods: A reaction to Nielsen & Phillips and new data. In Proceedings Companion of CHI, 1994 (Boston, MA, April 24-28, 1994) ACM, New York, 1994, 285-286.

John, B. E., Miller, P., Myers, B. A., Neuwirth, C. M., & Shafer, S. (eds.) (1993) Human-Computer Interaction in the School of Computer Science: A report of the Human-Computer Interaction Faculty in the School of Computer Science, Carnegie Mellon University. Carnegie Mellon University, School of Computer Science, Technical Report No. CMU-CS-92-193.

John, B.E., & Packer, H. (1995). Learning and Using the Cognitive Walkthrough Method: A case study Approach. In I. Katz, R. Mack, L. Marks, M. B. Rosson, & J. Nielsen (Eds.), Proceedings of CHI'95: New York: ACM.

John, B.E., & Kieras, D.E. (1996). Using GOMS for User Interface Design and Evaluation: Which Technique? ACM Transactions on Human-Computer Interaction, 3(4), 287-319.

John, B.E., & Marks, S.J. (1997). Tracking the effectiveness of usability evaluation methods. Behaviour & Information Technology, 16(4/5), 188-202.

John, B.E., & Mashyna, M.M. (1997). Evaluating a Multimedia Authoring Tool. Journal of the American Society for Information Science, 48(9), 1004-1022.

Karat, C.-M. (1994). A Comparison of User Interface Evaluation Methods. In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods. (pp. 203-233). Wiley.

Karat, C.-M., Campbell, R., & Fiegel, T. (1992). Comparison of empirical testing and walkthrough methods in user interface evaluation. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), Proceedings of CHI'92: New York: ACM.

Lewis, C., Polson, P.G., Wharton, C., & Rieman, J. (1990). Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces. In J. C. Chew & J. Whiteside (Eds.), Proceedings of CHI'90: New York: ACM.

Lewis, C., & Wharton, C. (1997). Cognitive Walkthrough. In M. Helander, T. K. Landauer, & P. Prabhu (Eds.), Handbook of Human-Computer Interaction. (pp. 717-732). Elsevier Science B.V.

Lewis, J.R. (1994). Sample Sizes for Usability Studies: Additional Considerations. Human Factors, 36(2), 368-378.

Mack, R., & Nielsen, J. (1993). Usability Inspection Methods: Report on a Workshop Held at CHI'92, Monterey, CA, May 3-4, 1992. SIGCHI Bulletin, 25(1), 28-33.

Molich, R., & Nielsen, J. (1990). Improving a Human-Computer Dialogue. Communications of the ACM, 33(3), 338-348.

Nielsen, J. (1993). Usability Engineering. Boston: Academic Press Inc.

Nielsen, J.,& Molich, R. (1990). Heuristic evaluation of user interfaces. In J. C. Chew & J. Whiteside (Eds.), Proceedings of CHI'90: New York: Addison-Wesley.

Nielsen, J.,& Phillips, V. (1993). Estimating the relative Usability of Two Interfaces: Heuristic, Formal, and Empirical Methods Compared. In S. Ashlund, A. Henderson, E. Hollnagel, & T. White (Eds.), Proceedings of InterCHI'93: New York: ACM.

Nielsen, J., & Mack, R.L. (1994). Usability Inspection Methods. Wiley.

Olson, G.M., & Moran, T.P. (1998). Commentary on “Damaged Merchandise”. Human-Computer Interaction, 13(3), 263-323.

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

Polson, P.G., & Lewis, C. (1990). Theory-Based Design for Easily Learned Interfaces. Human-Computer Interaction, 5, 191-220.

Polson, P.G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthrough: a method for theory-based evaluation of user interfaces. International Journal of Man-Machine Studies, 36, 741-773.

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

Rieman, J., Davies, S., Hair, D.C., Esemplare, M., Polson, P.G., & Lewis, C. (1991). An Automated Cognitive Walkthrough. In S. P. Robertson, G. M. Olson, & J. S. Olson (Eds.), Proceedings of CHI'91: New York: ACM.

Rowley, D.E.,& Rhoades, D.G. (1992). The Cognitive Jogthrough: A Fast-Paced User Interface Evaluation Procedure. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), Proceedings of CHI'92: New York: ACM.

Smith, S.L., & Mosier, J.N. (1986). *Guidelines for Designing User Interface Software*. ESD-TR-86-278. Bedford, MA: MITRE Corporation.

Tversky, A., & Kahneman, D. (1974). Judgment under Uncertainty: Heuristics and Biases. Science, 185, 1124-1131.

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

Wharton, C., Bradford, J., Jeffries, R., & Franzke, M. (1992). Applying cognitive walkthroughs to more complex user interfaces: Experiences, issues, and recommendations. In P. Bauersfeld, J. Bennett, & G. Lynch (Eds.), Procedings of CHI'92: New York: ACM.

Wharton, C., & Lewis, C. The Role of Psychological Theory in Usability Inspection Methods. (1993). Tech report. Colorado: University of Colorado at Boulder. CU-ICS-93-06

Wharton, C., Rieman, J., Lewis, C., & Polson, P.G. The Cognitive Walkthrough Method: A Practitioner's Guide. (1993). Institute of Cognitive Science, University of Colorado, Boulder, Colorado: #CU-ICS-93-07.

Wharton, C., & Lewis, C. (1994). The Role of Psychological Theory in Usability Inspection Methods. In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods. (pp. 341-350). New York: John Wiley & Sons, Inc.

Wharton, C., Rieman, J., Lewis, C., & Polson, P.G. (1994). The Cognitive Walkthrough Method: A Practitioner's Guide. In J. Nielsen & R. Mack (Eds.), Usability Inspection Methods. (pp. 105-140). Wiley.

Yin, R.K. (1994). Case Study Research - Design and Methods. (Second Edition ed.). California: SAGE Publications.

Appendix A: The Structure of the Case Study Database

Figure 8 shows the structure diagram of the case study database. Four areas in the structure diagram are shadowed polygons representing four logical entities. Each white rectangle in Figure 8 represents a table: the first capitalized text line indicates the table name, and the subsequent text lines in each table indicate fields in that table. A single underlined field in the table represents a unique key; if more than one field within the table is underlined, a combined key ensures the uniqueness of that table. A simple line between tables represents a one-to-zero or one-to-one relation between the two tables. A line going from table X to table Y where the line ends in a forked shape on table Y indicates a one-to-many relation between table X and table Y. That is, for each record in table X there can exist zero, one or more related records in table Y.

Table ANALYST and table TIMESLOT are seen in the center of the structure diagram in Figure 8. Table ANALYST shows the two analysts A1 and A2 in the case study. Table TIMESLOT shows the analysts' activities relative to time.

Tables related to the analysts' reading processes and their production of insight and difficulty notes are seen in the upper left shadowed area of the figure. In the upper left corner table NOTE holds information about a note taken by an analyst, the note type, and its contents. From the diary notes we know which papers the analysts read to learn about the CW technique. Hence, table PAPER (upper mid part) contains data on all articles read by the analysts in the case study. The link between table PAPER and table TIMESLOT goes through table PAPER READ. To get an idea of the relation between notes taken and papers read, we read all papers thoroughly and analyzed all notes to match paper paragraphs with notes taken. Table PAPER EXTRACT contains pointers to all paper paragraphs that were related to any note, while the actual relation between a given note and a given paper paragraph is identified in table NOTE REFERS TO.

In order to track the analysts' preparation and execution phases in the actual CW evaluation we created a table for each CW step. These tables are seen in the right shadowed area of the figure. Table TASK CHOSEN (upper right corner) contains a short description of each task scenario. Table ACTION contains the action sequence and system feedback for each task scenario. Table WALKTHROUGH contains failure or success stories for each action in an action sequence. It is necessary to connect TASK CHOSEN with ACTION and ACTION with WALKTHROUGH with a one-to-many relation as a task scenario consists of several actions, and an action leads to four success/failure stories in response to the four questions described in section 2. We did not ask the analysts to time stamp creation of task scenarios or to time stamp each action in an action sequence as this would have been too cumbersome for the analysts. Therefore table TASK CHOSEN, table ACTION and table WALKTHROUGH are not connected to table TIME SLOT. However, through the notes we do have a gross estimate of when the analysts picked a new task, when they constructed action sequences, and when they walked through those sequences.

The main aim of using CW is to evaluate a user interface to detect usability problems. The analysts were asked to record all detected problems in a Problem Description Report (PDR) located in table CW PDR (lower right corner). The one-to-many relation between table TIME SLOT (in the center of the figure) and table CW PDR indicates that all problem detections were related to a time slot in the diary. Moreover, a PDR would typically be filled out when a failure story was detected, as shown in the one-to-many relation between WALKTHROUGH and CW PDR. Sometimes, however, an analyst would detect a usability problem while reading the specification document, or during some other activity unrelated to the actual walkthrough. Hence, by introducing the relation between WALKTHROUGH and CW PDR, we can easily distinguish problem detections related to the CW technique itself (manifested by the link) and those unrelated to the technique (those without any link). In order to enable comparison of unique problem tokens from the usability test with unique problem tokens from the CW evaluation, we filtered the records in table CW PDR to unique problem tokens in table CW-P. A given PDR often relates to one unique problem token, but can potentially be related to more unique problem tokens. Similarly, a unique problem is created when at least one PDR points out a problem token. Hence, we have a one-to-many relation between table CW PDR and CW PDR/P MATCH; similarly we have a one-to-many relation between CW-P and CW PDR/P MATCH. In this way we represent a naturally many-to-many relation between CW PDR and CW-P through the constructed table CW PDR/P MATCH.

Tables related to the usability test are shown in the shadowed area in the lower part of the structure diagram. Both a user and an observer usually participate in a usability test. Information about those parties is given in table USER and OBSERVER. A usability problem appears when an observer judges a user's activity to violate any given usability criteria. This information is stored in table UT PDR. In order to transform problem description reports from the usability test to unique problem tokens, we created table UT PDR/P MATCH and UP-P that works just as CW PDR/P MATCH and CW-P described above.

Problem matches between unique problem tokens from CW and the usability test are stored in table CW-P UT-P MATCH seen in the lower part of the structure diagram. The analysts missed some problems detected in the usability tests. Each of these misses exists in table MISSED UT-P, which is linked to a specific usability problem in UT-P and to a certain walkthrough of an action in table WALKTHROUGH.

For the sake of completeness we have included table ANALYST REPORT in the structure diagram although it does not contain any data in the database. When we needed to refer to the analysts' reports we used only the paper based version.

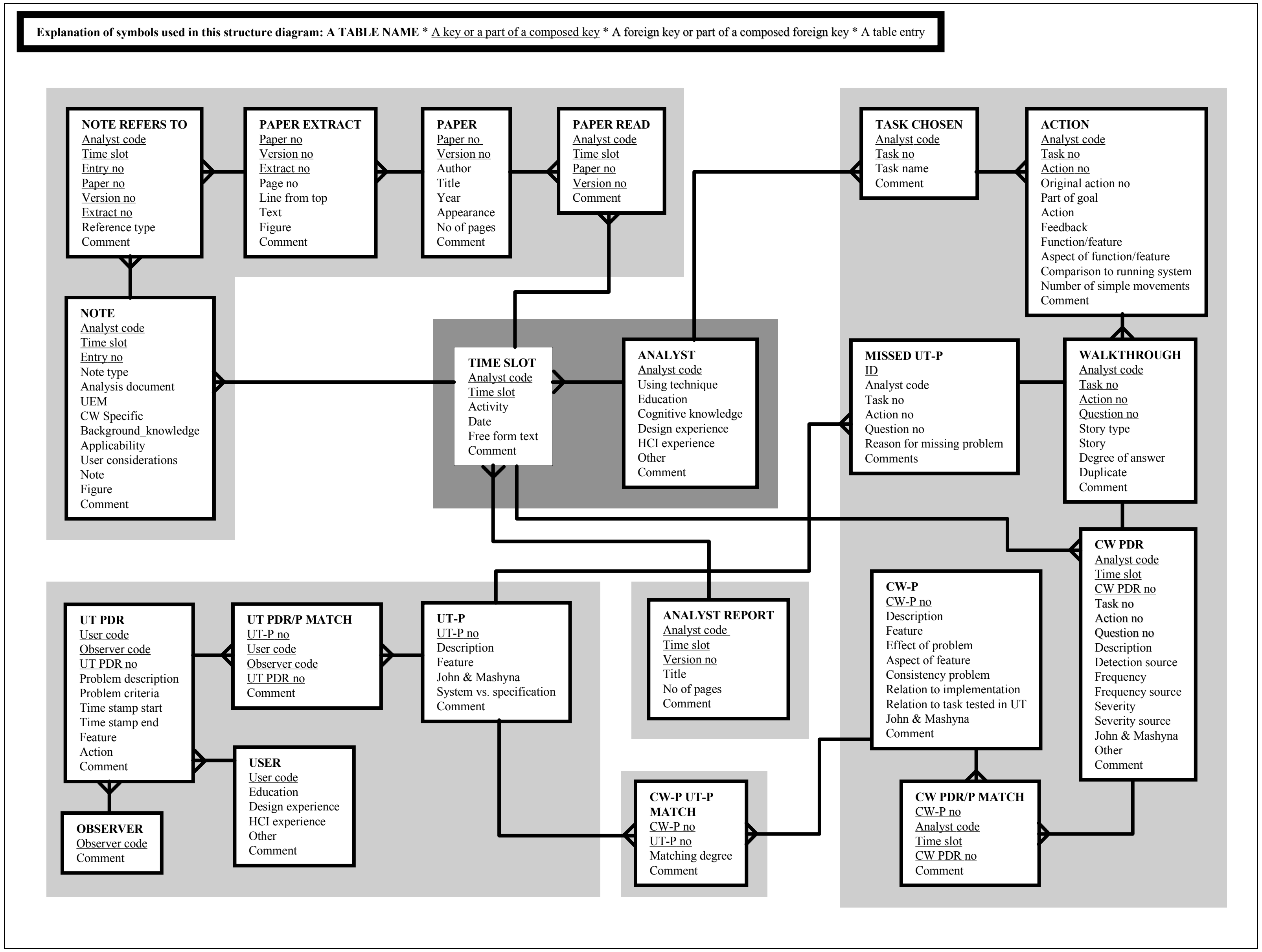


Figure 8. The structure diagram of the case study database.

Appendix 2

Hertzum, M., & Jacobsen, N.E. (1998). Individual Differences in Evaluator Performance during First-Time Use of the Cognitive Walkthrough Technique. Unpublished Manuscript.

(This page is intentionally left blank)

Individual Differences in Evaluator Performance during First-Time Use of the Cognitive Walkthrough Technique

Morten Hertzum
Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen, Denmark
+45 35 32 14 26
hertzum@diku.dk

Niels Ebbe Jacobsen
Psychological Laboratory
University of Copenhagen
Njalsgade 88
DK-2300 Copenhagen, Denmark
+45 35 32 87 59
niels@axp.psl.ku.dk

ABSTRACT

The cognitive walkthrough (CW) technique has been devised to provide practising system developers with a robust, easy-to-use usability evaluation method. The aim of this study was to compare the performance of 11 evaluators during their first-time use of the CW technique. To focus on the walkthrough, rather than the task selection, the evaluators were provided with set tasks. The study shows that detection of usability problems with CW is subject to substantial individual differences. None of the 33 detected problems was detected by all evaluators. A single evaluator detected on average 18% of the problems. The findings resemble those for other usability evaluation methods and indicate that the detailed prescription of how to perform CW has not made it a robust technique. In a certain sense an individual evaluator conducting a CW resembles one evaluator conducting a thinking-aloud study with one user.

Keywords
Cognitive walkthrough, evaluator performance, individual differences, usability, usability evaluation methods

INTRODUCTION

Practising system developers without a human factors background need robust, easy-to-use usability evaluation methods. The cognitive walkthrough (CW) technique has been devised to provide such a method and is particularly suited to evaluate designs before testing with users becomes feasible and as a supplement to user testing in situations where users are difficult or expensive to recruit.

The CW technique describes in detail what the evaluator has to do and when a problem has been detected. While several studies have evaluated how well CW predicts the problems encountered in thinking-aloud studies [5, 6, 8], only Lewis et al. [8] have assessed to what extent the detailed prescription of the evaluator's activities leads different evaluators to duplicate, rather than supplement, each other. Studies of other usability evaluation methods have shown that heuristic evaluation [13] and thinking-aloud studies [3] are subject to a substantial evaluator effect. Hence, as CW has evolved considerably since the study of Lewis et al. [8], and as an evaluator effect seems to exist for other usability evaluation methods, we find it interesting to study this effect for CW.

This paper investigates to what extent evaluators who perform a CW of the same tasks detect the same problems in the evaluated interface. While acknowledging the importance of choosing the right tasks in a usability evaluation, we have decided to focus on the actual walkthrough process primarily to make it possible to compare the evaluators' walkthroughs, secondary because the guides to the CW technique fail to describe the process of task selection. The achieved level of agreement among the evaluators' problem detection will allow an assessment of the robustness of the CW analysis. If the agreement is high, CW provides even novice evaluators such as those in this study with firm guidance in their analysis process. In case of substantial individual differences, a CW analysis is similar to other usability evaluation methods.

THE CW TECHNIQUE

CW is a usability inspection method that enables an evaluator to detect usability problems in a user interface based on a detailed specification document, screen mock-ups, or a running system. The technique is based on a cognitive theory of exploratory learning called CE+ [17, 18]. CW enables evaluators with no or, preferably, a basic understanding of CE+ to do a detailed simulation of a user's interaction with a system. It is sometimes recommended that CW should be performed by groups of co-operating evaluators [18, 20, 21], but all previous studies maintain that CW can also be performed by evaluators working individually [see, e.g., 9, 21].

Our study regards evaluators working individually and is based on the third version of CW described in Wharton et al. [21]. CW consists of a preparation phase and an execution phase. In the preparation phase the evaluator describes a typical user, chooses the tasks to be evaluated, and constructs a correct action sequence for each task. When this is done, the execution phase can begin. For each action in the action sequences the evaluator asks four questions: (1) Will the user try to achieve the right effect? (2) Will the user notice that the correct action is available? (3) Will the user associate the correct action with the effect trying to be achieved? (4) If the correct action is performed, will the user see that progress is being made toward solution of the task? With the description of the user in mind the evaluator decides whether each question leads to success or failure. In case of failure a usability problem has been detected.

PREVIOUS STUDIES

Individual differences in human performance have been studied extensively the past 100 years, but according to Dillon & Watson [1] the field of Human-Computer

Interaction (HCI) has largely ignored this research. Landauer [7] notes, however, that we all tend to greatly underestimate variability in performance and preferences. The only previous study of the level of agreement among CW evaluators is reported by Lewis et al. [8]. In addition to that Hilary Johnson has generously given us access to the data set from Dutt, Johnson, & Johnson [2]. Both these studies are based on the first version of CW, which used a single-page form with nine general questions and several sub-questions to evaluate each action, rather than the four questions in the version of CW used in our study. For that reason caution should be exercised in comparing results from those previous studies with our results.

Lewis et al. [8] had four evaluators individually perform a walkthrough of a simple mail system and found that they collectively detected almost 50% of the problems revealed by an empirical evaluation. The consistency across the four evaluators was fair in that more than half of the problems were detected by three evaluators but only one problem by all four evaluators, see Table 1. The applicability of these results is however difficult to assess since three of the evaluators were co-authors of the article, familiar with CE+ theory, and had discussed the general trends of the data from the empirical evaluation prior to their walkthroughs [8, pp. 238-239].

Dutt et al. [2] had a student without HCI experience, a student with HCI experience, and a person with HCI research experience individually perform a CW of a personnel recruitment system. Half of the reported problems were detected by all three evaluators, see Table 2, and a single evaluator found on average 73% of the total set of problems from the three walkthroughs. The authors noted that '[t]he number of problems found is relatively low given the quality of the interface.' This could indicate that the evaluators did not find all the problems in the interface or that the evaluators applied a rather high threshold for the amount of difficulty or inconvenience inflicted on the user before they reported a problem.

On the surface the detection rate of a single evaluator using CW is higher than that of single evaluators using heuristic evaluation and thinking-aloud studies. Nielsen [11] reports that a novice evaluator conducting a heuristic evaluation detected on average 22% of all problems in a simple telephone-based banking system. Nielsen [12] reports that a novice evaluator conducting a thinking-aloud study with one subject detected on average 29% of all problems in a word processor and an outliner. By choosing a thinking-aloud study with one subject we compare one CW session with one thinking-aloud session.

Detected by exactly	No. of problems	
1 evaluator	2	(10%)
2 evaluators	5	(25%)
3 evaluators	12	(60%)
4 evaluators	1	(5%)
Total	20	(100%)

Table 1. The detected problems broken down by the number of evaluators detecting them, study by Lewis et al. [8].

Detected by exactly	No. of problems	
1 evaluator	11	(34%)
2 evaluators	4	(13%)
3 evaluators	17	(53%)
Total	32	(100%)

Table 2. The detected problems broken down by the number of evaluators detecting them, study by Dutt et al. [2].

METHOD

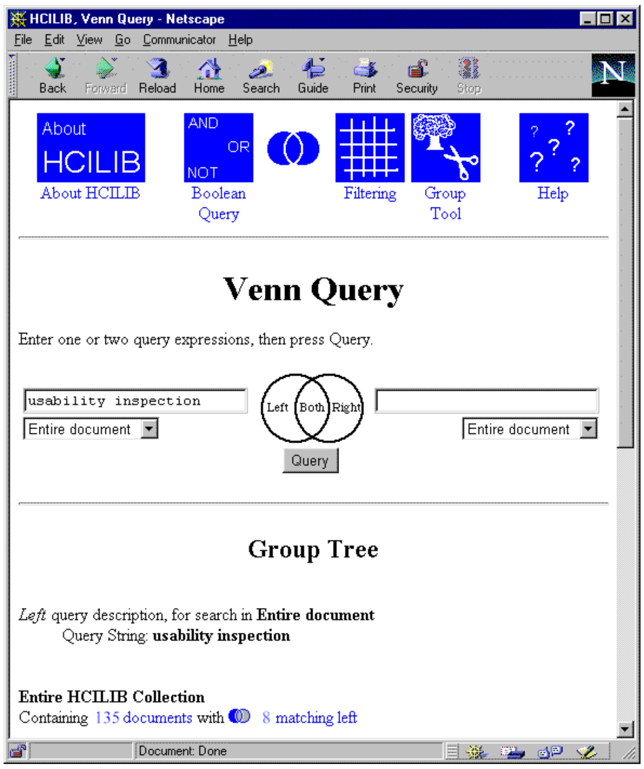
Eleven graduate students evaluated a prototype of a World Wide Web-based system against three set tasks using CW.

Evaluators

The evaluators were 11 graduate students in computer science taking an introductory course in human-computer interaction. Apart from their undergraduate degree in computer science the academic backgrounds of the evaluators included undergraduate degrees in business administration, chemistry, education, English, linguistics, physics, and psychology. Half of the evaluators also worked part time as system developers in the industry. The evaluators were all male and their age ranged from 23 to 32 years with a mean of 25.5 years. The evaluators were experienced World Wide Web (WWW) users but they had no prior knowledge of the system to be evaluated, and they had no or very shallow knowledge of CW.

System

The evaluated system, called HCILIB [16], was a prototype of a WWW-based library giving access to a collection of scientific articles on human-computer interaction, see Figure 1. HCILIB integrates Boolean search with a scatter-gather inspired technique to display a browsable structure of the collection. Boolean searches can be expressed as conventional Boolean queries or by means of a Venn diagram which relieves the user from direct interaction with ANDs and ORs in face of the simple queries that form the majority of cases. HCILIB is a quite small system intended for users with basic knowledge of human-computer interaction and the World Wide Web. The evaluated version of HCILIB gave access to a collection of 135 articles from the CHI conferences in 1995 and 1996.


HCILIB, Venn Query - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Guide Print Security Stop
About HCILIB
About HCILIB
AND OR NOT
Boolean Query
Filtering
Group Tool
Help
Left Both Right
Entire document
Query
Document: Done

Venn Query

Enter one or two query expressions, then press Query.

usability inspection

Group Tree

Left query description, for search in **Entire document**
Query String: **usability inspection**

Entire HCILIB Collection
Containing 135 documents with 8 matching left

Figure 1. HCILIB's Venn Query page. The user enters a query phrase in the text boxes and, possibly, limits the search to certain parts of an article by selecting for example title or abstract in the combo box, rather than 'Entire document'. The search is performed when the user clicks the Query button under the Venn diagram. The results of the search are displayed in the lower part of the page (reproduced with permission).

Tasks

Three tasks were used in the evaluation. The two first tasks concerned simple and typical uses of HCILIB; the third task required a more thorough understanding of how the search facilities can be used to express more complex queries. The tasks appear below along with sample action sequences for solving them (the evaluators were to construct the action sequences themselves):

Task 1. Use HCILIB's Venn Query facility to find the articles concerning how the usability of computer systems can be evaluated without involving users. How many such articles are there? (In general, evaluation of the usability of computer systems is known as 'usability evaluation'. Evaluation that does not involve users is often referred to as 'usability inspection', while evaluation with users is called 'usability testing'.)

Sample action sequence:

1. Click on the 'Enter the library' link
2. Click on the 'Venn Query' button
3. Move cursor to the left input field
4. Type 'usability inspection'
5. Click on the 'Query' button
6. Click on the link to the hit list, labelled '8 matching left'

Task 2. One of the articles found in task 1 is entitled 'Making A Difference - The Impact of Inspections'[8]. Bring this article up for closer study and make an electronic copy of it.

Sample action sequence:

1. Click on the 'Document location' link of the article
2. Open the 'File' menu of the browser
3. Select 'Save as'
4. Enter a file name
5. Select a destination directory
6. Complete the save dialog by pressing the 'Save' button

Task 3. The article in task 2 refers to an article on heuristic evaluation (a certain usability inspection method) authored by J. Nielsen & R. Molich[9]. Find all the articles that refer to this article on heuristic evaluation. How many such articles are there?

Sample action sequence:

1. Click on the 'Venn Query' button
2. Move cursor to the left input field
3. Type 'Nielsen Molich heuristic evaluation'
4. Click on the combo box below the input field
5. Select 'References' to restrict searching to the references at the end of the articles
6. Click on the 'Query' button
7. Click on the link to the hit list, labelled '2 matching left'

Procedure

The experiment was embedded in a grade-giving assignment where the students were asked to construct action sequences and do a cognitive walkthrough of the three tasks given above. Just before the assignment was handed out the first author gave the evaluators two hours of instruction in the CW technique. This instruction consisted of a presentation of the practitioner's guide to CW [21], a discussion of this guide, and an exercise where the evaluators got some hands-on experience and instant feedback. The evaluators had to perform their cognitive walkthroughs individually and to document them in a problem list describing each detected problem and the CW question that uncovered it. As a rough estimate each evaluator spent 2-3 hours completing his CW. In addition, the evaluators produced a report describing their planning and execution of their walkthroughs, and their opinions on using the technique.

Based on the eleven problem lists the two authors independently constructed a master list of unique problem tokens. We agreed on 80% of the problem tokens; disagreements were resolved through discussion and a consensus was reached.

[8] Sawyer, P., Flanders, A. & Wixon, D. Making a difference - the impact of inspections, in Proceeding of the ACM CHI'96 Conference (Vancouver, Canada), ACM Press, 376-382.

[9] Nielsen, J. & Molich, R. Heuristic Evaluation of User Interfaces, in Proceeding of the ACM CHI'90 Conference, ACM Press, 249-256.

RESULTS

The eleven evaluators reported a total of 74 problem instances from their CWs, including a number of cases where an evaluator reported the same problem several times. Excluding these duplicates, 65 problem instances remained. As some problem instances were detected by more than one evaluator, the 65 problem instances made up 33 unique problem tokens (in the following just termed problems).

The evaluators differed with respect to which and how many problems they detected, see Figure 2. The largest number of problems found by a single evaluator was 13, whereas the lowest was 2. Looking at evaluators with less extreme performance, the ratio between the third quartile score and the first quartile score[10] is 2:1. The standard deviation is 3.20. Thus, the individual differences are substantial compared to Dutt et al. [2] and Lewis et al. [8].

As much as 58% of the problems were detected by only a single evaluator, and no single problem was detected by all evaluators, see Table 3. Studies of heuristic evaluation have found that no evaluator was consistently good or bad, since the evaluators finding few problems found some problems overlooked by most evaluators, and the evaluators finding many problems overlooked some that were found by most other evaluators [15]. In our study, the most glaring finding is the diversity in the evaluators' problem detection. In fact 30 of the 33 problems were detected by at most three evaluators, suggesting a great deal of misses—or false alarms—in the performance of single evaluators.

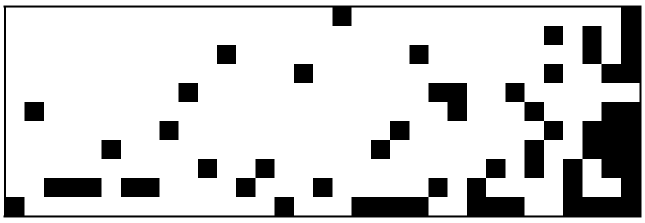


Figure 2. Matrix showing who found which problems. Each row represents an evaluator, each column a problem, and each black square that the evaluator detected the problem.

Detected by exactly	No. of problems	
1 evaluator	19	(58%)
2 evaluators	8	(24%)
3 evaluators	3	(9%)
4 evaluators	0	(0%)
5 evaluators	1	(3%)
6 evaluators	1	(3%)
7 evaluators	0	(0%)
8 evaluators	0	(0%)
9 evaluators	0	(0%)
10 evaluators	1	(3%)
11 evaluators	0	(0%)
Total	33	(100%)

Table 3. The detected problems broken down by the number of evaluators detecting them.

We were curious to know how many individual evaluators were needed to detect the majority of the problems in the interface. Figure 3 shows the average number of problems that would be found by aggregating the sets of problems found by several evaluators. The aggregates were formed by calculating the average number of problems found across all combinations of a given number of evaluators. For each combination a given problem was considered found if it was found by at least one of the selected evaluators. A single evaluator found on average 18% of the 33 known problems.

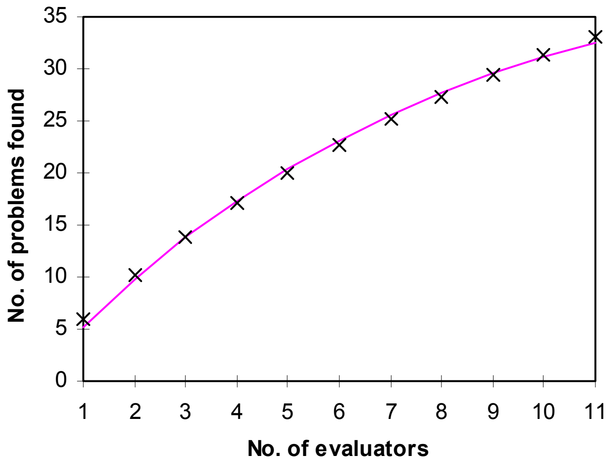


Figure 3. The number of problems detected shown as a function of the number of evaluators. The data points are the aggregated values from the experiment. The curve plots an estimate given by the formula $f(k) = 43(1 - (1 - 0.121)^k)$.

Probability theory indicates that the number of problems one would expect to find at a given group size is given by the formula $f(k) = n(1 - (1 - p)^k)$, where n is the total number of problems in the interface, p is the probability of detecting a problem in one walkthrough, and k is the number of independent walkthroughs, i.e. the number of evaluators [10, 14]. By transforming the formula and applying linear regression n and p can be estimated to 43 and 0.121, respectively. With these estimates the fit between the formula and our data is highly significant (squared correlation coefficient (R^2) = 0.998, standard error for n estimate = 1.4%, standard error for p estimate

[10] The third quartile score is the value for which 75% of the evaluators found fewer problems and 25% found more problems. The first quartile score is the value for which 25% of the evaluators found fewer problems and 75% found more problems.

= 0.20%, p<0.001). Hence, according to the formula HCILIB has a total of 43 problems in the tasks evaluated, the 11 evaluators have collectively detected 76% of these problems, and the estimated average detection rate of a single evaluator is 12%. Given the three set tasks, our study shows that five first-time evaluators must walk through the tasks to detect half of the estimated number of problems in the interface.

Robustness of results

The average percentage of problems found by a single evaluator was several times larger in the data set from Dutt et al. [2] than in our study. While several factors such as the nature of the evaluated system, the tasks, and the version of CW may contribute to this, it does raise the question of the robustness of our results.

Problem Severity

It could be hypothesised that the individual differences in problem detection were caused largely by cosmetic problems and were thus essentially a matter of taste. If CW evaluators detect critical usability problems with a much higher probability than cosmetic problems, then the individual differences in problem detection would be less grave. To investigate this hypothesis we extended our study with an assessment of problem severity.

A week after the assignment six of the evaluators, those present at the course lecture, received a list with short descriptions of all the encountered problem tokens. The evaluators were told that HCILIB would soon be released for public use and they then spent approximately 15 minutes individually answering the question: 'Which of the detected problems must be solved before HCILIB is released?' Their answers provide six fast and cheap severity assessments of the entire set of problems and constitute a straightforward extension of the CW technique in cases where several evaluators perform individual walkthroughs. Nielsen [13] recommends this extra phase as an extension of heuristic evaluation.

Two reservations have to be made regarding the validity of the evaluators' severity assessments: (1) The evaluators may to some extent have been biased toward the problems they originally detected themselves. However, in real-life settings it seems very likely that severity will be assessed by people who were also involved in detecting the problems. So this situation is of considerable practical interest. (2) Following their CWs the evaluators conducted a thinking-aloud session using the three tasks mentioned earlier. Since thinking-aloud sessions tend to be very convincing they may have biased the evaluators' severity assessments toward the problems identified during these sessions. The problem list handed out to the evaluators included the number of times a problem was encountered (by an evaluator or a subject), but without linking problems to usability evaluation methods. The potential bias introduced by the thinking-aloud sessions is therefore restricted to the problems the evaluators' originally detected themselves.

A total of 20 problems were selected as severe by the six evaluators who individually selected 4, 5, 7, 7, 10, and 13 problems. Before we investigate the relationship between detection rate and problem severity, it is worth noting how much the evaluators differ in their assessment of problem severity. The sheer numbers of problems selected display some individual differences. Looking at the actual problems selected by the evaluators the individual differences are substantial, see Table 4. As much as 35% of the problems were selected by only a single evaluator and another 35% by two evaluators. Not a single problem was selected by all six evaluators. Thus, the cognitive walkthroughs did not give rise to a common agreement as to what constitute the central usability issues in the interface. This result could be ascribed to the aforementioned biases or to the evaluators' lack of experience but other usability evaluation methods studies with usability specialists as evaluators have reported similar results [3, 13].

In order to investigate whether the moderate proportion of problems found by a single evaluator was caused by less severe problems, two sets of presumably severe problems were defined: The first set simply consisted of the 20 problems that were included in any of the six subsets with problems to be fixed before release of the system. The second set consisted of the 13 problems that two or more evaluators had included in their subset. Figure 4 shows the proportion of problems detected for different numbers of evaluators. The bottom curve is identical to the data points in Figure 3 and is shown for reference, the curve in the middle corresponds to the set with 20 problems, and the topmost curve corresponds to the set with 13 problems. For the three levels of severity a single evaluator found on average 18%, 21%, and 26% of the problems. The curves are quite close to each other, and a Kolgomorov-Smirnov test shows that there is absolutely no basis for assuming that the evaluators were better at detecting severe problems than at detecting problems in general (standard score (Z) = 0.426, p=0.99). For thinking-aloud, Lewis [10] got a similar result, but other studies have found that the number of evaluators who detect a problem increases with the severity of the problem [for example, 19].

Selected by exactly	No. of problems	
1 evaluator	7	(35%)
2 evaluators	7	(35%)
3 evaluators	2	(10%)
4 evaluators	1	(5%)
5 evaluators	3	(15%)
6 evaluators	0	(0%)
Total	20	(100%)

Table 4. The severe problems broken down by the number of evaluators who considered them severe, i.e. considered that they should be fixed before release of the system. Six evaluators performed this severity assessment.

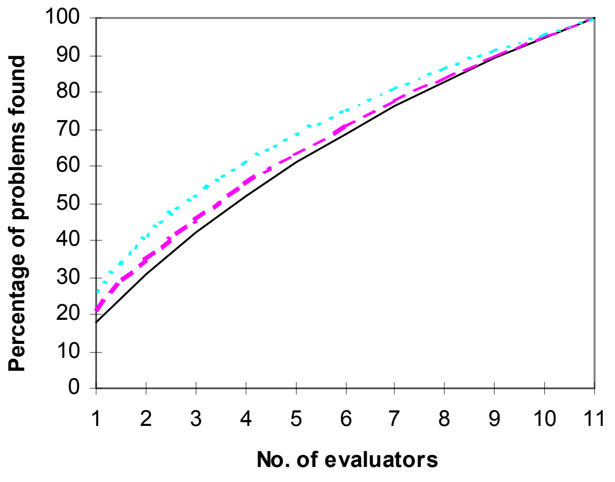


Figure 4. The percentage of problems found shown at different levels of problem severity and for varying numbers of evaluators. The bottom curve corresponds to all 33 problems. The curve in the middle corresponds to the 20 problems that at least one evaluator wanted fixed before release of the system. The topmost curve corresponds to the 13 problems that two or more evaluators wanted fixed before release. The difference between the curves is not significant.

The Grouping of the Problem Instances

It could be hypothesised that the average proportion of problems detected by a single evaluator changes considerably with minor changes in the way the problem instances are grouped into problem tokens. If this is the case then our results, and those of other studies, may to a large extent be an artefact of the way the master list is constructed.

In our data set the grouping of the problem instances is based on two candidate master lists produced independently by the two authors. The final master list was constructed by discussing the disagreements one by one and agreeing on whether one or the other author's grouping was the more appropriate. An alternative approach is to handle all cases of disagreement by merging the problem tokens that constitute each disagreement into one problem token. This approach is based on the reasoning that if one evaluator has joined problem instance a and b and the other evaluator has joined a and c then a, b, and c probably all belong to the same problem token. The result of consistently applying this approach is fewer problem tokens, which contain more problem instances. A single evaluator detected on average 21% of the 26 problem tokens defined by this approach. This is almost the same as the 18%, which is the result of calculating the average detection rate on the basis of the 33 problem tokens on the original master list.

In Dutt et al. [2] the grouping of the problem instances is split into two distinct steps: the matching of the problems reported by the individual evaluators into an intermediate master list and the generalisation of common problems into the problem types of the final master list. A single evaluator detected on average 65% of the 45 problems on the intermediate master list and the aforementioned 73% of the 32 problems on the final master list. Thus, the results do not seem to be particularly sensitive to changes in the grouping of the problem instances.

The Number of Evaluators

The number of evaluators needed to find all problems in an interface depends on the detection rate of the individual evaluators. However, the accuracy of the calculation of the detection rate depends, in turn, on whether the evaluators collectively detect all problems in the interface. To illustrate the relationship between the number of evaluators and the calculated detection rate our experiment can be scaled down to experiments with fewer evaluators. This is done by selecting, say, three of the eleven evaluators, counting the number of problems they have collectively detected, and calculating how many of these problems a single one of these evaluators on average detects. Table 5 shows the average obtained by repeating this process for all possible selections of three evaluators; as well as for any other number of evaluators. Thus, scaling our experiment down to three evaluators we find that they detect 13.91 problems and that a single evaluator detects on average 43% of them.

The seeming increase in the evaluators' performance as the number of evaluators in the experiment decreases is an artefact of the assumption that the number of problems found collectively by the evaluators is the same as the total number of problems in the interface. In our study a group of three evaluators is neither enough to find the majority of problems nor to reliably predict how large a proportion of the problems in the interface a single evaluator detects. In general, detection rates from different studies are not comparable without an argument that they are based on the, if necessary estimated, total number of problems in the interface.

No. of evaluators	Total no. of problems found	Percentage of problems found by one evaluator
1	5.91	100%
2	10.24	59%
3	13.91	43%
4	17.16	35%
5	20.08	30%
6	22.73	26%
7	25.15	24%
8	27.36	22%
9	29.40	20%
10	31.27	19%
11	33.00	18%

Table 5. Scaling our experiment down to fewer evaluators. For each number of evaluators the table shows the number of problems they collectively find and the percentage of these problems that one evaluator on average detects.

The Evaluators' Background and Experience

Since the evaluators were first-time users of CW it is possible that more experienced evaluators would achieve a higher detection rate and display less individual differences. While this seems likely there is no empirical data to prove it. Compared to the evaluators in our experiment, two of the evaluators in Dutt et al. [2] had more background knowledge of CW but they too were first-time users of the technique. This distinction between background knowledge and use experience is noteworthy because CW was intended to be a usability evaluation method for people with little or no background knowledge of cognitive psychology or terminology. Future research must show to what extent evaluators experienced in the use of CW achieve higher detection rates and better agreement on the severity of the detected problems.

DISCUSSION

A series of studies have found substantial individual differences in evaluators' performance of CW, heuristic evaluation, and thinking-aloud studies [3, 8, 11, 14]. On that background it seems probable that these differences originate partly in the general activity of usability evaluation, rather than solely in the specific evaluation methods. The studies suggest that, irrespective of usability evaluation method, evaluators are constrained by numerous factors such as their general expertise, experience in using the method, and knowledge of the system evaluated, as well as by the usability problem criteria used, time constraints, and motivational factors. These factors are likely to vary more in real-life settings than in the experimental studies of the evaluator effect where an effort has been made to control such factors.

In our study a single evaluator using CW detects 12% of the estimated total number of problems. This is somewhat less than but still comparable to the detection rates of 22% for novice evaluators using heuristic evaluation [11] and 29% for novice evaluators conducting a single thinking-aloud session [12][11]. Thus, the detailed prescription of the evaluator's activities has not made CW more robust than heuristic evaluation, which is a much more informal evaluation method. With respect to the robustness of CW, Jacobsen & John [4] find that the CW evaluator in their study report only few false alarms, i.e. problems predicted by the CW evaluator but not observed in an accompanying thinking-aloud study. This result suggests that a major reason for the individual differences in our study is misses rather than false alarms.

It is a critical aspect of the CW technique to what extent posing the four questions supports the evaluator in reaching the answers. To answer the questions accurately the evaluator needs to know quite a lot about how the target users will react to different user interface properties and facilities. It is not yet clarified to what extent moderate familiarity with CE+ theory or a sample of actual target users provides the evaluator with the ability to infer these reactions. In case of insufficient knowledge of how the users will use the interface the walkthrough becomes inaccurate for at least two reasons: (1) Anchoring, i.e. despite the evaluator's efforts the walkthrough will end up evaluating the system against a user who is much too similar to the evaluator to be representative of the actual users. (2) Stereotyping, i.e. the walkthrough will end up reflecting a view of the users that is much too homogeneous to accommodate the diversity of the actual users.

In the preparation phase of CW the evaluator describes the users and this easily leads to stereotyping, for example Wharton et al. [21, p. 109] provide the following two examples of user descriptions: '*people who use existing ATM machines*' and '*Macintosh users who have worked with MacPaint*'. Slightly caricatured, stereotyping means that the walkthrough is done with one user in mind. Thus, a single evaluator using CW resembles a thinking-aloud study with one evaluator and one user, i.e. a single thinking-aloud session. Jacobsen et al. [3], who had four experienced evaluators individually analyse the same four thinking-aloud sessions, find that the number of problems detected increases substantially with both the number of evaluators and the number of users. Though a single thinking-aloud session is better than doing no evaluation work, it is commonly regarded as insufficient because any single session will miss a great many of the problems in the interface. Similarly, the majority of problems will be missed when one evaluator inspects an interface using CW.

CONCLUSION

To detect the usability problems in a small web-based system 11 evaluators, who were first-time users of the CW technique, individually performed cognitive walkthroughs of three tasks. The evaluators differed tremendously with respect to which problems they detected. Collectively the evaluators detected 33 problems but not a single problem was detected by all 11 evaluators and a single evaluator detected on average a mere 18% of the 33 problems. Linear regression indicates that the interface contains another ten problems not detected by any of the evaluators. Six of the evaluators also assessed the severity of the problems by identifying the problems they considered it necessary to fix before release of the system. The detection rate for these presumably severe problems was only marginally higher than for the entire set of problems.

The average performance of individual evaluators is probably not acceptable for practical use of the CW technique. We believe that CW evaluators run a large risk of greatly underestimating the variability of the target users of the evaluated system. If a CW is based on one homogeneous picture of the user, the evaluation bears some resemblance to a thinking-aloud study with one user. While Wharton et al. [21] state that CWs can be performed by individual evaluators as well as by groups of evaluators this study strongly suggests that a group of evaluators is necessary to get reasonably reliable results, at least when the evaluators are inexperienced.

Additional studies are required to learn how more experienced evaluators perform and whether the best results are obtained by a group of co-operating evaluators or by evaluators who work individually and then aggregate their results. Moreover, this study has only reported quantitative data on the evaluator effect in CW analyses. There is however no doubt about the need of

[11] In these studies of heuristic evaluation and thinking aloud the number of problems found collectively by all the evaluators equals the linear regression estimate of the total number of problems in the interface [14]. Therefore we compare those studies with our evaluators' estimated detection rate of 12% rather than the 18% of the 33 known problems.

studying *why* we see these individual differences among evaluators.

ACKNOWLEDGEMENTS

The Danish Research Councils have supported this work. We are grateful to Hilary Johnson for giving us access to the data set from Dutt, Johnson, & Johnson [2]. We also wish to thank Ejlif Jacobsen for his help on the statistical treatment of our data. Finally, we wish to thank the evaluators who participated in the experiment.

REFERENCES

1. Dillon, A., & Watson, C. User analysis in HCI—the historical lessons from individual differences research. International Journal of Human-Computer Studies 45 (1996), 619-637.
2. Dutt, A., Johnson, H., & Johnson, P. Evaluating evaluation methods, in G. Cockton, S.W. Draper, & G.R.S. Weir (eds.), People and Computers IX (1994), Cambridge University Press, Cambridge, 109-121.
3. Jacobsen, N.E., Hertzum, M., & John, B.E. The evaluator effect in usability studies: problem detection and severity judgments, to appear in Human Factors and Ergonomics Society 42nd Annual Meeting (Chicago, October 1998), HFES.
4. Jacobsen, N.E., & John, B.E. In preparation.
5. Jeffries, R., Miller, J.R., Wharton, C., & Uyeda, K.M. User interface evaluation in the real world: a comparison of four techniques, in Proceedings of the ACM CHI'91 Conference (New Orleans, April-May 1991), ACM Press, 119-124.
6. John, B.E., & Mashyna, M.M. Evaluating a multimedia authoring tool. Journal of the American Society for Information Science 48, 11 (1997), 1004-1022.
7. Landauer, T.K. Behavioral research methods in human-computer interaction, in M. Helander, T.K. Landauer, & P. Prabhu (eds.), Handbook of Human-Computer Interaction. Second, completely revised edition. Elsevier, Amsterdam, 203-227.
8. Lewis, C., Polson, P., Wharton, C., & Rieman, J. Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces, in Proceedings of the ACM CHI'90 Conference (Seattle, WA, April 1990), ACM Press, 235-242.
9. Lewis, C., & Wharton, C. Cognitive walkthroughs, in M. Helander, T.K. Landauer, & P. Prabhu (eds.), Handbook of Human-Computer Interaction. Second, completely revised edition. Elsevier, Amsterdam, 717-732.
10. Lewis, J.R. Sample sizes for usability studies: additional considerations. Human Factors 36, 2 (1994), 368-378.
11. Nielsen, J. Finding usability problems through heuristic evaluation, in Proceedings of the ACM CHI'92 Conference (Monterey, CA, May 1992), ACM Press, 373-380.
12. Nielsen, J. Estimating the number of subjects needed for a thinking aloud test. International Journal of Human-Computer Studies 41 (1994), 385-397.
13. Nielsen, J. Heuristic evaluation, in J. Nielsen & R.L. Mack (eds.), Usability evaluation methods, John Wiley, New York, 1994, 25-62.
14. Nielsen, J., & Landauer, T.K. A mathematical model of the finding of usability problems, in Proceedings of the INTERCHI'93 Conference (Amsterdam, April 1993), ACM Press, 206-213.
15. Nielsen, J., & Molich, R. Heuristic evaluation of user interfaces, in Proceedings of the ACM CHI'90 Conference (Seattle, WA, April 1990), ACM Press, 249-256.
16. Perstrup, K., Froekjaer, E., Konstantinovitz, M., Konstantinovitz, T., Soerensen, F.S., & Varming, J. A World Wide Web-based HCI-library designed for interaction studies, in The Third ERCIM User Interfaces for All Workshop (Obernai, France, November 1997).
17. Polson, P., & Lewis, C. Theory-based design for easily learned interfaces. Human-Computer Interaction 5, 2&3 (1990), 191-220.
18. Polson, P., Lewis, C., Rieman, J., & Wharton, C. Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. International Journal of Man-Machine Studies 36 (1992), 741-773.
19. Virzi, R.A. Refining the test phase of usability evaluation: how many subjects is enough? Human Factors 34, 4 (1992), 457-468.
20. Wharton, C., Bradford, J., Jeffries, R., & Franzke, M. Applying cognitive walkthroughs to more complex user interfaces: experiences, issues, and recommendations, in Proceedings of the ACM CHI'92 Conference (Monterey, CA, May 1992), ACM Press, 381-388.
21. Wharton, C., Rieman, J., Lewis, C., & Polson, P. The cognitive walkthrough method: a practitioner's guide, in J. Nielsen & R.L. Mack (eds.), Usability Inspection Methods, John Wiley, New York, 1994, 105-140.

Appendix 3

Hertzum, M.,& Jacobsen, N.E. (1999). The Evaluator Effect during First-Time Use of the Cognitive Walkthrough Technique. In H. Bullinger & J. Ziegler (Eds.), Human-Computer Interaction – Ergonomics and User Interfaces. Vol. 1., 1063-1067.

(This page is intentionally left blank)

The Evaluator Effect during First-Time Use of the Cognitive Walkthrough Technique†

Morten Hertzum
Centre for Human-Machine Interaction
Risø National Laboratory, Denmark

Niels Ebbe Jacobsen
Department of Psychology
University of Copenhagen, Denmark

1 Introduction

Practising system developers without a human factors background need robust, easy-to-use usability evaluation methods. The cognitive walkthrough (CW) technique (Lewis et al. 1990, Wharton et al. 1994) has been devised to provide such a method and is particularly suited to evaluate designs before testing with users becomes feasible and as a supplement to user testing.

While several studies have evaluated how well CW predicts the problems encountered in thinking-aloud studies (e.g. John and Mashyna 1997, Lewis et al. 1990), only Lewis et al. have assessed to what extent different evaluators obtain the same results when evaluating the same interface. Data from Lewis et al. suggests that the variability in performance among evaluators using CW is much lower than that of evaluators using heuristic evaluation or thinking-aloud studies (Jacobsen et al. 1998, Nielsen 1994). One reason for this seemingly higher robustness of CW might be that it is a quite structured process. CW has however evolved considerably since the study of Lewis et al. Moreover, their data was limited in sample size and applicability to actual CW evaluators.

To inform practitioners and methods developers about the robustness of CW this paper investigates to what extent novice evaluators who perform a CW of the same tasks detect the same problems in the evaluated interface. While acknowledging the importance of choosing the right tasks in a CW, we have decided to focus on the actual walkthrough process.

† This work was supported by grants from the Danish National Research Foundation and the Danish Research Councils. We wish to thank the evaluators for their time and effort.

2 The CW Technique

Our study is based on the version of CW described in Wharton et al. (1994). CW consists of a preparation phase and an execution phase. In the preparation phase the evaluator describes a typical user, chooses the tasks to be evaluated, and constructs a correct action sequence for each task. In the execution phase the evaluator asks four questions for each action in the action sequences: (1) Will the user try to achieve the right effect? (2) Will the user notice that the correct action is available? (3) Will the user associate the correct action with the effect trying to be achieved? (4) If the correct action is performed, will the user see that progress is being made toward solution of the task? With the description of the user in mind the evaluator decides whether each question leads to a success or failure story. In case of a failure story a usability problem has been detected.

3 Method

Eleven graduate students in computer science evaluated a prototype of a Web-based system against set tasks. Half of the evaluators had design experience from industry, but they had no prior knowledge of the system to be evaluated. The evaluated system, called HCILIB, was a prototype of a Web-based library giving access to a collection of scientific articles on human-computer interaction. HCILIB (Perstrup et al. 1997) integrates Boolean search with a scatter-gather inspired technique to display a browsable structure of the collection. Boolean searches can be expressed as conventional Boolean queries (using ANDs and ORs) or by means of a Venn diagram metaphor. The Venn diagram metaphor relieves the user from direct interaction with logical expressions. Instead, query terms are entered into two search boxes, A and B, and the search results are automatically sorted into three disjunctive sets corresponding to $A–B$, $A\cap B$, and $B–A$.

The experiment was embedded in a grade-giving assignment where the students were asked to construct action sequences and do a cognitive walkthrough of three set tasks. Just before the assignment was handed out the evaluators received two hours of instructions in CW based on a lecture on the practitioner's guide to CW (Wharton et al. 1994). The instructions also offered the evaluators some hands-on experience followed by instant feedback. The evaluators documented their cognitive walkthroughs in a problem list describing each detected problem. As a rough estimate each evaluator spent 2-3 hours completing his/her CW. Based on the problem lists from the 11 evaluators the two authors independently constructed a master list of unique problem tokens. Combining these master lists we had an inter rater reliability of 80%; disagreements were resolved through discussion and a consensus was reached.

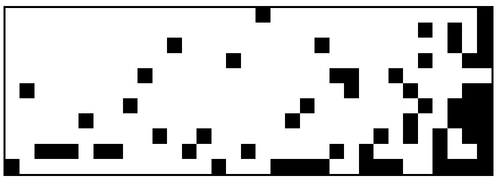


Figure 1. Matrix showing who found which problems. Each row represents an evaluator, each column a problem, and each black square that the evaluator detected the problem.

4 Results

The eleven evaluators reported a total of 74 problem instances from their CWs. These problem instances made up 33 unique problem tokens (in the following just termed problems). As much as 58% of the problems were detected by only a single evaluator, and no single problem was detected by all evaluators, see Figure 1. A single evaluator found on average 18% of the 33 known problems.

We were curious to know how groups of evaluators performed compared to single evaluators. Figure 2 shows the average number of problems that would be found by aggregating the sets of problems found by different groups of evaluators. For each group a given problem was considered found if it was found by at least one of the evaluators in the group. The results suggest a great deal of misses – or false alarms – in the performance of single evaluators. An analysis of problem severity could not explain this evaluator effect, as the detection rate for

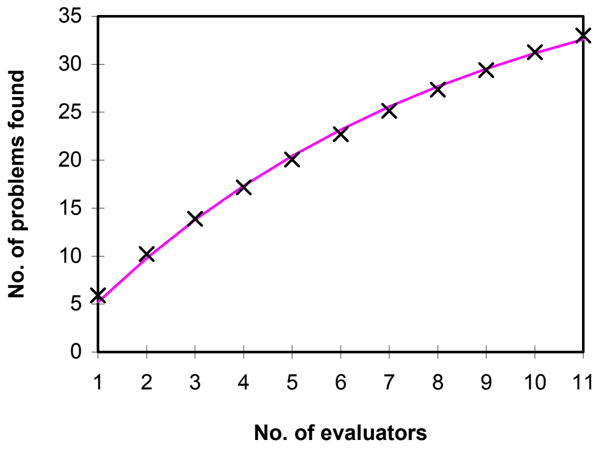


Figure 2. The number of problems detected shown as a function of the number of evaluators. The data points are the aggregated values from the experiment. The curve plots an estimate given by the formula $f(k) = n(1 - (1 - p)^k)$, for $n = 43$ and $p = 0.121$.

severe problems was only marginally higher than for the entire set of problems.

5 Discussion

As for other usability evaluation methods it is crucial to CW that the walkthrough leads to a reliable problem list. Studies on heuristic evaluation and usability studies have found substantial individual differences in the evaluators' performance (Jacobsen et al. 1998, Nielsen 1994). This suggests that our results are partly attributable to usability evaluation in general, rather than solely to CW. We believe, however, that the CW procedure falls short of providing the evaluators with a feel for the users and thus becomes inaccurate for two reasons: (1) Anchoring, i.e. despite the evaluator's efforts the walkthrough will end up evaluating the system against a user who is much too similar to the evaluator to be representative of the actual users. (2) Stereotyping, i.e. the walkthrough will end up reflecting a user that is much too homogeneous to accommodate the diversity of the actual users of the system evaluated.

We investigated the anchoring and stereotyping hypotheses by looking closer on how the evaluators answered the four questions on identical actions. Though the evaluators constructed their action sequences from the same three tasks only 4 out of an average of 15 actions were identical across all evaluators. One of these actions is to execute a query by activating a Query button. In evaluating this action three evaluators reported success stories on all four questions, while eight evaluators reported a total of five different problems: Three evaluators reported that the user will click a Venn pictogram situated above the Query button, rather than the button itself. Three evaluators reported that there is weak feedback from the system after clicking the Query button. Two evaluators reported that the Enter key does not execute the query, i.e. the user has to use a pointing device. One evaluator reported that the caption on the button should be changed. And finally, one evaluator reported that the user will forget to activate the Query button. It seems quite reasonable that all problems would actually happen for some users in a real situation, just as some users might experience no troubles using the Query button, as suggested by three evaluators. Though all evaluators' use of the four questions on the analysed action seems reasonable, the outcome is very different across evaluators. The same pattern was found for the three other actions that were identical across the evaluators.

The evaluators' descriptions of the target user in the preparation phase are similar in content, and they generally provide a broad description of a large, homogeneous group of users. The descriptions are in many respects similar to the descriptions of users given as examples by Wharton et al. (1994). Despite the formal description of the user, or perhaps because of the generality of these descriptions, the evaluators might not fully realise the heterogeneity of the user group or their walkthrough might be anchored to their own experience with the system. Each of the four questions drives the evaluator to think of the user's behaviour in a certain situation. When the fictive user description becomes too fuzzy or lacks details to judge the user's behaviour, the evaluator unintentionally substitutes the description with a particular user much like herself/himself. Thus, evaluators tend to produce success stories if they imagine themselves having no troubles using the feature in question, and they report problems when they imagine themselves having troubles in the particular situation. In this sense a single evaluator using CW resembles an evaluator performing a thinking-aloud study with one user, namely himself/herself.

Wharton et al. (1994) state that CWs can be performed by individual evaluators as well as by groups of co-operating evaluators. For inexperienced CW evaluators our study strongly indicates that several evaluators are necessary to achieve a performance that is acceptable for practical use of the CW technique. Additional studies are required to learn how more experienced evaluators perform and to study more closely *why* we see these individual differences.

References

Jacobsen, N. E., Hertzum, M. & John, B. E. (1998). The evaluator effect in usability studies: problem detection and severity judgments. In *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting* (Chicago, October 5-9, 1998), pp. 1336-1340. Santa Monica: HFES.

John, B. E. & Mashyna, M. M. (1997). Evaluating a multimedia authoring tool. *Journal of the American Society for Information Science*, 48(11), 1004-1022.

Lewis, C., Polson, P., Wharton, C. & Rieman, J. (1990). Testing a walkthrough methodology for theory-based design of walk-up-and-use interfaces. In *Proceedings of the ACM CHI'90 Conference* (Seattle, April 1990), pp. 235-242. New York: ACM Press.

Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J. & Mack, R. L. (Eds.): *Usability Inspection Methods*, pp. 25-62. New York: John Wiley.

Perstrup, K., Frøkjær, E., Konstantinovitz, M., Konstantinovitz, T., Sørensen, F. S. & Varming, J. (1997). A World Wide Web-based HCI-library designed for interaction studies. In *Third ERCIM User Interfaces for All Workshop* (Obernai, France, November 1997).

Wharton, C., Rieman, J., Lewis, C. & Polson, P. (1994). The cognitive walkthrough method: a practitioner's guide. In Nielsen, J. & Mack, R. L. (Eds.): *Usability Inspection Methods*, pp. 105-140. New York: John Wiley.

(This page is intentionally left blank)

Appendix 4

Jacobsen, N.E., Hertzum, M., & John, B.E. (1998a). The evaluator effect in usability tests. In C.-M. Karat & A. Lund (Eds.), Human Factors in Computing Systems CHI'98 Summary: ACM, 255-256.

(This page is intentionally left blank)

The Evaluator Effect in Usability Tests

Niels Ebbe Jacobsen
HCI Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3891
jacobsen@cs.cmu.edu

Morten Hertzum
Department of Computer Science
University of Copenhagen
DK-2100 Copenhagen, Denmark
hertzum@diku.dk

Bonnie E. John
HCI Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3891
bej@cs.cmu.edu

ABSTRACT
Usability tests are applied in industry to evaluate systems and in research as a yardstick for other usability evaluation methods. However, one potential threat to the reliability of usability tests has been left unaddressed: the evaluator effect. In this study, four evaluators analyzed four videotaped usability test sessions. Only 20% of the 93 unique problems were detected by all four evaluators and 46% were detected by only a single evaluator. Severe problems were detected more often by all four evaluators (41%) and less often by only one evaluator (22%) but a substantial evaluator effect remained.

Keywords
Usability, user testing, usability test, evaluator effect

INTRODUCTION
Usability testing—also known as think-aloud studies—is probably the single-most important method for practical evaluation of user interfaces [2]. Although there is no standardized procedure for running usability tests they are typically conducted in controlled environments and include a number of sessions involving a user working on set tasks while thinking out loud. Based on live observations, or analyses of video tapes, an evaluator constructs a problem list from the difficulties the users have accomplishing the tasks. Previously, it has been shown that four or five users detect roughly 80% of the problems in an interface [4] as long as the average likelihood of a user detecting a problem ranges between 0.32 and 0.42 [1].

While the sufficient number of users has been studied in different ways, the effect of different evaluators analyzing the same sessions has been left untouched. This study investigates how evaluators differ in analyzing identical video tapes of a usability test. So far the evaluator effect has only been studied for Heuristic Evaluation, where it has a substantial impact. In a test of a voice response system a single novice evaluator found 22% (on average) of the problems found by all evaluators in the study. Usability specialists and so-called double experts did somewhat better finding on average 41% and 60% of the problems, respectively [3].

METHOD
Evaluators
Four HCI research evaluators, all familiar with the theory and practice of usability testing, analyzed four video tapes. Table 1 shows the evaluators' experience with the system evaluated in this study, and their evaluation experience in terms of the total number of users previously analyzed.

Evaluator	Occupation	System experience	Evaluation experience
E1	Associate professor, HCI	10 hours	52 users
E2	Doctoral student, HCI	5 hours	4 users
E3	Assistant professor, HCI	2 hours	6 users
E4	Usability lab manager	12 hours	66 users

Table 1. Evaluator profiles

Video tapes
Four Macintosh users spent about an hour thinking aloud as they worked through four or five set tasks in a multi-media authoring system [6]. Although they were experienced computer users, none of the users had previous experience with the system and no instructions were given. The system resembles an advanced word processor insofar that the user can create documents consisting of plain text, still graphics, movies, and animations. The users were asked to create several pages consisting of some text, a figure, and an animation, to add some items to the glossary and the table of contents, and to switch two pages. The same experimenter ran all four experiments, and he did not interrupt the users unless they forgot to think aloud, explicitly gave up solving a task, or got stuck for more than three minutes.

Procedure
Evaluators E1 and E2 knew the authoring system well, while evaluator E3 and E4 were asked to familiarize themselves with it prior to their analysis. The evaluators had access to a written system specification (35 pages) and the running system throughout their participation in the study. The evaluators were asked to detect and describe all problems in the interface based on analyzing the four tapes in a preset order. No time constraints were enforced. The evaluators were requested to report three properties for each problem detected: (a) evidence consisting of the users' action sequence and/or verbal utterances, (b) a free-form problem description, and (c) the criteria for identifying the problem. The evaluators used nine predefined problem criteria: (1) The user articulates a goal and cannot succeed in attaining it within three minutes, (2) the user explicitly gives up, (3) the user articulates a goal and has to try three or more actions to find a solution, (4) the user produces a result different from the task given, (5) the user expresses surprise, (6) the user expresses some negative affect or says something is a problem, (7) the user makes a design suggestion, (8) a system crash, and (9) the evaluator generalizes a group of previously detected problems into a new problem.

RESULTS AND DISCUSSION

Based on the four evaluators' individual problem reports a master list of 93 unique problem tokens (UPTs) was constructed. This was done independently by the first two authors. They agreed on 84% of the UPTs; disagreements were resolved through discussion and a consensus was reached.

The percentage of the UPTs reported by E1, E2, E3 and E4 were 63%, 39%, 52%, and 54% respectively. Thus, a single evaluator detected on average 52% of the problems, which is only slightly more than the 41% found by usability specialists in a Heuristic Evaluation [3].

Table 2 shows the evaluator effect for all UPTs and for just the severe problems. We define a severe problem as a UPT judged by one or more evaluators to violate problem criteria 1, 2 or 8. The evaluator effect for all UPTs is substantial, 46% of all UPTs were found by only a single evaluator. Though the severe problems were generally detected by more than one evaluator, only 41% of the severe problems were detected by all four evaluators. Furthermore, 73% of the severe UPTs were judged as violating problem criteria other than 1, 2 or 8 by some evaluator. Thus, any given evaluator should not be expected to use problem criteria as a reliable method to judge severity on UPTs.

	1 evaluator	2 evaluators	3 evaluators	4 evaluators
Severe UPTs	8 (22%)	7 (19%)	7 (19%)	15 (41%)
All UPTs	43 (46%)	19 (20%)	12 (13%)	19 (20%)

Table 2. Number of UPTs detected by groups of 1, 2, 3, and 4 evaluators.

The effect of adding more evaluators to a usability test resembles the effect of adding more users; both additions increase the overall number of UPTs found. As illustrated in Figure 1, the average increase was 46% going from one to two evaluators, 23% going from two to three evaluators, and 17% going from three to four evaluators when all four users were included in the calculation (the four points on the rightmost vertical). Calculating the effect of running more users we found an increase of 55% going from one to two users, 26% going from two to three users, and 23% going from three to four users when all evaluators were included in the calculation (the topmost curve). The declining number of new UPTs detected as more users are added confirms the results from similar studies [1, 4, 5].

We were able to describe the number of UPTs found based on the number of users and the number of evaluators:

$$\text{No. of UPT} = 19.35 * (\text{no. of evaluators})^{0.505} * (\text{no. of users})^{0.661} \quad \text{(eq.1)}$$

The fit between the mathematical model and our data is highly significant (R^2=0.997; SEE=2.6%). The four or five users that others have found to be sufficient in a usability

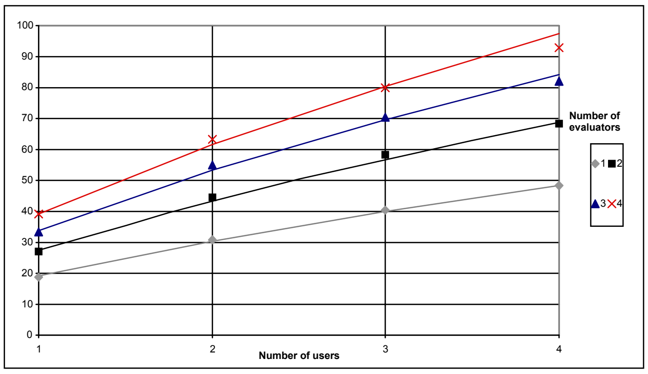


Figure 1. Data points are the observed average number of UPTs detected by all sets of users and of all sets of evaluators. The curves plot Eq. 1 for 1, 2, 3, and 4 evaluators.

test could, in our study, be traded for two evaluators running two or three users without decreasing the number of detected UPTs.

CONCLUSION

This study demonstrates that four evaluators, each analyzing video tapes of four usability test sessions differed a great deal in their analysis. Only a fifth of the UPTs were detected by all evaluators, while almost half of the UPTs were detected by only one evaluator. Severe problems faired better, with two fifths detected by all evaluators and only one fifth by a single evaluator. Given previous studies of individual differences the results of this study is not very surprising but certainly overlooked or neglected in the field of usability testing. The evaluator effect puts usability testing in perspective, and questions the use of data from usability tests as a baseline for comparison to other usability evaluation methods.

Additional research is needed to understand how this effect varies by evaluator experience, problem severity, task-type, system-type, and other variables important to usability practitioners and researchers.

REFERENCES

1. Lewis, J. Sample Sizes for Usability Studies: Additional Considerations. *Human Factors 36, 2* (1994), 368-378.
2. Nielsen, J. *Usability Engineering*. Academic Press, Boston, 1993.
3. Nielsen, J. Finding usability problems through heuristic evaluation, in *Proceedings of CHI'92* (Monterey, CA, May 1992), ACM Press, 373-380.
4. Nielsen, J. Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human-Computer Studies, 41* (1994), 385-397.
5. Nielsen, J. & Landauer, T.K. A Mathematical Model of the Finding of Usability Problems. in *Proceedings of INTERCHI'93* (Amsterdam, Holland, April 1993), ACM Press, 206-213.
6. Pane, J.F. & Miller, P.L. The ACSE multimedia science learning environment, in *Proceedings of the 1993 International Conference on Computers in Education* (Taipei, Taiwan, 1993).

Appendix 5

Jacobsen, N.E., Hertzum, M., & John, B.E. (1998b). The evaluator effect in usability studies: problem detection and severity judgments. In Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting: Santa Monica, CA: Human Factors and Ergonomics Society, 1336-1340.

(This page is intentionally left blank)

THE EVALUATOR EFFECT IN USABILITY STUDIES: PROBLEM DETECTION AND SEVERITY JUDGMENTS

Niels Ebbe Jacobsen
Morten Hertzum
University of Copenhagen
DK-2300 Copenhagen, Denmark

Bonnie E. John
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Usability studies are commonly used in industry and applied in research as a yardstick for other usability evaluation methods. Though usability studies have been studied extensively, one potential threat to their reliability has been left virtually untouched: the evaluator effect. In this study, four evaluators individually analyzed four videotaped usability test sessions. Only 20% of the 93 detected problems were detected by all evaluators, and 46% were detected by only a single evaluator. From the total set of 93 problems the evaluators individually selected the ten problems they considered most severe. None of the selected severe problems appeared on all four evaluators' top-10 lists, and 4 of the 11 problems that were considered severe by more than one evaluator were only detected by one or two evaluators. Thus, both detection of usability problems and selection of the most severe problems are subject to considerable individual variability.

INTRODUCTION

A usability study – also known as a think-aloud study – is probably the single-most important usability evaluation method (UEM) in practice (Nielsen, 1993), and it is undoubtedly the most investigated UEM. Many dimensions of usability studies have been investigated including the sufficient number of users (e.g. Lewis, 1994; Virzi, 1992), individual versus cooperating users (Hackman & Biers, 1992), the level of experimenter intervention (Held & Biers, 1992), task settings (Held & Biers, 1992; Karat et al., 1992), retrospective versus concurrent think-aloud (Ohnemus & Biers, 1993), and the impact of usability tests in real-life settings (e.g. Jørgensen, 1989). Moreover, usability testing has been compared to and used as a yardstick for other UEMs (see e.g. Bailey et al., 1992; Cuomo & Bowen, 1994; Henderson et al., 1995; John & Marks, 1997; John & Mashyna, 1997; Karat, 1994).

However, the effect of the *evaluator* on the process has been left virtually untouched in usability tests, although it has been studied in other UEMs such as Heuristic Evaluation (Nielsen & Molich, 1990). For example, in his chapter entitled "Usability Testing", Nielsen (1993) discussed the effect of variability in *users* as the *only* threat to the reliability of usability tests. Furthermore, Virzi et al. (1993) state that the "...think-aloud method incorporates the users' perspectives directly, without being filtered by an intermediary..." (p. 312) On the other hand, Holleran (1991) observed that there can be substantial disagreement among evaluators because the collected data are primarily subjective in nature, but he supplied no data to confirm this assertion. This paper extends the study of Jacobsen et al. (1998) in that it addresses how the detection and severity rating of usability problems depend on the evaluators who observe and analyze the usability test sessions. We focus on a quite controlled variant of usability tests where identical test sessions are conducted in a usability lab, under the management of an experimenter who only interferes in the user's work if strictly necessary.

METHOD

The usability test sessions

Four experienced Macintosh users spent about an hour thinking aloud as they individually worked through a set of tasks in a multimedia authoring system hereafter called the *Builder* (Pane & Miller, 1993). None of the users had previous experience with the Builder, and they did not receive any instructions in the use of the system. The Builder resembles an advanced word processor in that the user can create documents consisting of plain text, still graphics, movies, and animations.

The users were asked to create a new document based on a printed target document consisting of pages containing text, figures, and animations. The users had to add and edit some glossary items, add entries to a table of contents, delete a page, switch two pages, and save their document in two different versions. The same experimenter ran all four sessions. He did not interrupt the users unless they forgot to think aloud, explicitly gave up solving a task, or got stuck for more than three minutes. Prior to each session the experimenter introduced the study, taught the user to think out loud, and handed out the first task to the user after having read it out loud. Whenever the user finished a task, the experimenter handed out the next task. The sessions were videotaped for later analysis. In the following all users will be addressed as "he" though both males and females participated.

Evaluators

Four HCI research evaluators, all familiar with the theory and practice of usability testing, analyzed the four videotapes. Table 1 shows the evaluators' experience with the Builder and their evaluation experience in terms of the total number of users previously analyzed. The authors of this paper were themselves evaluators in the study (three of the four evaluators). The third author designed the usability study, but the experimenter who ran the test sessions was not aware of our evaluator-effect research. The first two authors, who had no input into the design of the usability study, conducted the compilation and analyses of the evaluator's responses. In the following all evaluators will be addressed as "she" though both males and females participated.

Eval-uator	Occupation	Number of users previously analyzed	Initial experience with the Builder	Average analysis time per tape
E1	Associate professor	52 users	10 hours	3.8 hours*
E2	Doctoral student	4 users	5 hours	2.7 hours
E3	Assistant professor	6 users	2 hours	2.9 hours
E4	Usability lab manager	66 users	12 hours	4.5 hours

Table 1. The HCI research evaluators' previous usability test experience, their experience with the Builder and the average time spent analyzing each tape (each tape lasted approximately 1 hour). *The analysis time shown for E1 is the time spent analyzing the last tape, as she did not keep track of the time she spent on the first three tapes.

Procedure

Evaluators E1 and E2 knew the Builder well before this study was conducted; evaluators E3 and E4 familiarized themselves with it prior to their analysis. All evaluators had access to a written specification of the Builder (35 pages) and the running system throughout their analysis. The evaluators were asked to detect and describe all problems in the interface based on analyzing the four tapes in a preset order. No time constraints were enforced (see Table 1 for the time spent analyzing a tape). The evaluators were requested to report three properties for each detected problem: (a) a free-form problem description, (b) evidence consisting of the user's action sequence and/or verbal utterances, and (c) one of nine predefined criteria for identifying a problem.

The evaluators used the following set of problem detection criteria: (1) the user articulates a goal and cannot succeed in attaining it within three minutes, (2) the user explicitly gives up, (3) the user articulates a goal and has to try three or more actions to find a solution, (4) the user creates an item in his new document different from the corresponding item in the target document, (5) the user expresses surprise, (6) the user expresses some negative affect or says something is a problem, (7) the user makes a design suggestion, (8) the system crashes, and (9) the evaluator generalizes a group of previously detected problems into a new problem.

Using the four evaluators' individual problem reports (276 raw problem reports), the first two authors (NJ and MH) created a master list of unique problem tokens (UPTs) using the following procedure. First, each author split apart any of the original raw problem reports he thought contained more than one problem. NJ split 16 original reports, producing an additional 23 problems; MH split 17 original reports, producing an additional 18 problems. Eight of these new problem reports were the same, so both authors had a list of 284 problems in common (with MH having an additional 10 and NJ having an additional 15). Each author then examined their lists and eliminated duplicates. Of the 284 problems on both lists, the authors agreed on 245 (86%) as to whether they were unique or duplicated. The authors discussed the disagreements and the problems they did not share, reached consensus, and formed a master list of 93 UPTs.

To study the evaluators' judgment of problem severity the evaluators received a version of the master list containing (1) a short description of each UPT, (2) the number of users experiencing the UPT, (3) the number of evaluators detecting it, (4) the problem detection criteria it was attributed to, and (5) the interface feature it involved. Each evaluator was presented with a scenario in which a project manager had constrained the evaluators to point out the ten most severe UPTs, as a tight deadline forced the developer team to fix only those few UPTs in the next release of the Builder. In the scenario the evaluators were told that their selection of UPTs should be based on the information on the master list and on other factors, such as considerations concerning experienced versus novice users, and the Builder's use in real life settings. The UPTs on the top-10 lists were not prioritized, but each UPT was annotated with the evaluator's reasons for including that particular UPT.

RESULTS

The percentages of the total of 93 UPTs reported by E1, E2, E3, and E4 were 63%, 39%, 52%, and 54% respectively. Thus, a single evaluator detected on average 52% of all known UPTs in the Builder interface.

The effect of adding more evaluators to a usability test resembles the effect of adding more users; both additions increase the overall number of UPTs found. Figure 1 depicts the number of the 93 UPTs detected as a function of the number of both evaluators and users. The average increase in UPTs found was 46% going from one to two evaluators, 23% going from two to three evaluators, and 17% going from three to four evaluators when all four user-videotapes were included in the calculation (the four points on the rightmost vertical).

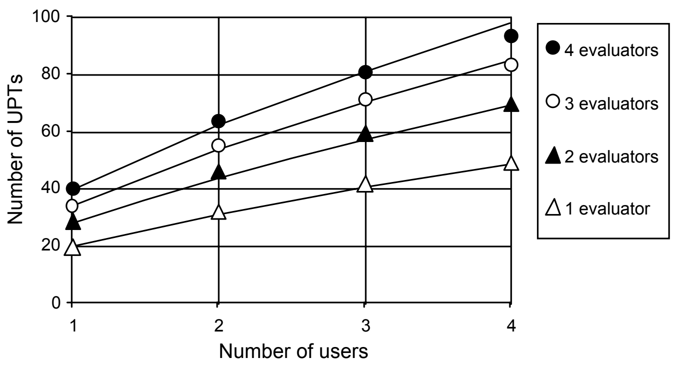


Figure 1. The number of detected UPTs depends on the number of users and the number of evaluators. The data points are the observed numbers, averaged over all combinations of users or evaluators. The curves plot Equation 1 for 1, 2, 3, and 4 evaluators.

Calculating the effect of running more users, we found an increase of 55% going from one to two users, 26% going from two to three users, and 23% going from three to four users when all evaluators were included in the calculation (the topmost curve). The declining number of new UPTs detected as more users are added confirms the results from similar studies (Lewis, 1994; Nielsen & Landauer, 1993; Virzi, 1992).

Our data can be described with Equation 1. The fit between the equation (the curves in Figure 1) and our data (the data points in Figure 1) is highly significant (squared correlation coefficient (R^2) = 0.997; standard error of estimate = 2.6%; $p<0.001$). The equation describes the relationship between the number of detected UPTs, number of users, and number of evaluators for our study. Other studies may result in different values for the constant and the exponents.

$$\text{No. of UPTs} = 19.35 * (\text{no. of evaluators})^{0.505} * (\text{no. of users})^{0.661} \quad (1)$$

The evaluator effect for all UPTs is substantial; as much as 46% of the UPTs were found by only a single evaluator, while 20% were found by all four evaluators (see Figure 2). Problem criteria 9 (a problem identified as a generalization of previously detected problems) might be more likely to differ across evaluators, since the generalization process is quite subjective. However, only 5% of all problem reports were attributed to criteria 9. Hence the evaluator effect cannot be caused by this criteria alone.

To investigate whether the level of agreement among the evaluators differs when detecting more severe problems, we used three methods to extract severe problems. First, we extracted the 37 UPTs attributed, by any evaluator, to one or more of the three problem criteria we thought more severe than the rest: (1) the user articulates a goal and cannot succeed in attaining it within three minutes, (2) the user explicitly gives up, and (8) the system crashes. Second, we looked at the 25 UPTs that appeared on at least one evaluator's top-10 list. Third, we extracted the 11 UPTs that were included on more than one top-10 list. Table 2 shows that the evaluator effect in detecting problems was progressively less extreme for the sets of more severe problem, but even for the smallest set of severe problems it was still substantial.

UPTs	No. of UPTs	Detected by			
		only 1	any 2	any 3	all 4
			evaluators		
All UPTs	93	46%	20%	13%	20%
Violating criteria 1, 2, or 8	37	22%	19%	19%	41%
Any UPTs on top-10 lists	25	20%	20%	8%	52%
More than one top-10 list	11	9%	27%	0%	64%

Table 2. Percentages of the UPTs detected by only 1, any 2, any 3, and all 4 evaluators.

Detection is not the only measure of interest, however. Severity judgment also differed substantially between the four evaluators. Looking at their top-10 lists, we found large differences; 56% of the 25 UPTs that appeared on the four top-10 lists were selected by only a single evaluator, 28% were selected by two evaluators, and 16% were selected by three evaluators. Not a single UPT appeared on all four top-10 lists!

DISCUSSION

Previous studies have shown that no single user will come across all problems in an interface. Our study refined this finding by suggesting that no single evaluator will detect all problems in a usability test. The number of problems revealed in a usability test is dependent on both the number of users and the number of evaluators.

Using Equation 1, we can estimate how many new UPTs a fifth user might find. In our study, five users and one evaluator could be traded for three users and two evaluators without decreasing the number of detected problems. Moreover, the two evaluators analyzing three users will, on average, detect the same number of problems from the union of the top-10 lists as the single evaluator analyzing five users.

Using different approaches to identify the severe problems in the Builder interface we found that more severe problems showed a tendency toward being detected by more evaluators. This tendency is in keeping with Virzi's (1992) results, but given the lack of statistical power in our study it does not contradict Lewis's (1994) result of no significant correlation between problem detection and problem severity.

Problem severity can be judged by the evaluators who initially detected problems in the interface or by a different group of people not affected by the process of detecting problems prior to their severity judgment. The evaluators who initially detected problems in the interface will be fully able to understand the problem descriptions and the interface as they have worked closely with the interface and the videotapes prior to their severity judgment. However, such evaluators may be biased toward the problems they originally detected themselves. Jeffries (1994) found that problem reports are often unclear and ambiguous. Hence, relying on severity judgments made by evaluators who have not been involved in problem detection introduces uncertainty regarding the interpretation of the problem reports. In collecting severity judgments one always has to balance the risk of biased severity judgments against that of misinterpreted problem reports.

It should be noted that our definitions of "severe" problems are not empirically founded, that is, we have little evidence that these problems would indeed be more problematic than the other problems if users were to encounter them in the real world. Certainly, the first method of identification (by problem detection criteria) has some empirical support (i.e., that at least one evaluator saw evidence in at least one user's behavior of excessive delay, giving up, or a system crash), but all three methods require additional research to establish their validity.

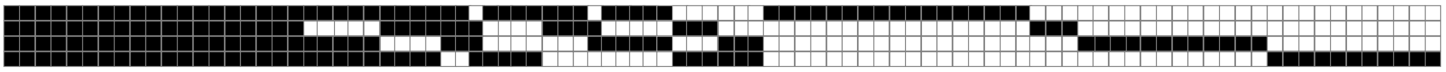


Figure 2. Matrix showing who found which problems. Each row represents an evaluator, each column a problem, and each black square that the evaluator detected the problem.

The substantial differences among the evaluators in terms of their selection of problems for their top-10 lists reveal that judgments of severity are highly personal. In fact, we were surprised that *no* UPT appear on *all* evaluators' top-10 lists. To further investigate the evaluators' strategies in selecting severe problems we extended our study by collecting retrospective reports.

RETROSPECTIVE REPORTS

Though caution should be exercised in using retrospective reports the evaluators were asked to write down their strategy for creating their top-10 lists. Moreover, for each of the 15 UPTs appearing on a different evaluator's top-10 list, but not on their own, they were asked to explain why they judged this UPT to be less severe than those on their own list.

E4 based her severity judgment solely on the frequency information on the master list. She first selected the UPTs experienced by all four users (5 UPTs). She then looked at the UPTs that were experienced by either two or three users but were detected by all evaluators, picking 5 of 10 possible UPTs. She did not read the content of all 93 UPTs. Filtering by frequency of occurrence and then by frequency of detection, she missed identifying a system crash as a severe UPT because it was experienced by only one user.

In contrast, E1, E2, and E3 read through all 93 UPTs highlighting potentially severe problems. The first pass reduced each evaluator's set of problems to between 15 and 25 UPTs. These three evaluators then read through their reduced set of problems removing UPTs gradually until they reached 10 UPTs.

Although they all used this 'homing-in approach' to selecting top-10 UPTs, the details of their approach differed. E1 reported focusing on the needs of experienced users in real-life situations. Six of her top-10 UPTs were explicitly selected for this reason and the same reason was given for excluding 10 of the 15 UPTs the other evaluators chose.

E2 used her opinion of the severity of the problem criteria in many of her decisions (6 of her top-10). The user frequency also played a role in E2's creation of her top-10 list. She explicitly favored general problems over specific ones and tried to balance the needs of novice and experienced users. She rejected evidence that new features were needed (e.g., a *search* command) in favor of evidence that there were problems with existing features.

After the initial pass, E3 removed potentially severe UPTs by comparing them pair-wise, rather than relying on general heuristics. She based these comparisons on her ability to provide sound arguments for one or the other UPT, with no dominant pattern to these arguments. She described her last few decisions as "more or less random", as the compared problems appeared almost equally severe.

In summary, the evaluators' methods for extracting top-10 UPTs varied greatly, according to both concurrent explanations for selecting problems and their retrospective reports. The selection methods were based on multiple aspects such as the evaluators' favor for certain user groups, the number of evaluators and users encountering a problem, the violated problem criteria, expectations about real-world usage of the Builder, etc. All these aspects may catch important dimensions of problem severity but they also point out that severity is an ill-defined concept.

SUMMARY AND CONCLUSION

Our study shows that analyzing usability test sessions is an activity subject to considerable individual variability. When four research evaluators with extensive knowledge in HCI evaluated the same four usability test sessions, almost half of the problems were detected by only a single evaluator, while just 20% of the problems were detected by all evaluators.

The evaluators' detection rate was higher for more severe problems. Severe problems were identified both by problem-detection criteria or inclusion on evaluators' top-10 lists. However, all the sets of severe problems still displayed a substantial evaluator effect. Moreover, the evaluators disagreed substantially in their judgment of what constituted the ten most severe problems. None of the 25 problems in the union of the evaluators' top-10 lists was selected as severe by all evaluators, and 56% appeared on only a single top-10 list.

The evaluator effect revealed in this study shows that usability tests are less reliable than previously reported. No single evaluator will detect all problems in an interface when analyzing usability test sessions, and any pair of evaluators is far from identifying the same set of severe problems.

FUTURE WORK

Clearly, this small study should be followed by larger studies examining how the evaluator effect manifests itself with such variables as different definitions of severity, instructions to the evaluators, problem detection criteria, evaluator training, system type, and task types. Our investigation of the evaluator effect asks many more questions than we can answer at this time.

ACKNOWLEDGMENTS

This work was supported by The Danish Research Councils and by the United States' National Science Foundation, Award# IRI-9457628. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of The Danish Research Council, the Danish Government, NSF, or the U. S. Government. We would like to thank Ejlif E. Jacobsen for his help with the statistical analysis.

REFERENCES

Bailey, R.W., Allan, R.W., & Raiello, P. (1992). Usability testing vs. heuristic evaluation: A head-to-head comparison. In Proceedings of the Human Factors Society 36th Annual Meeting. Santa Monica: HFS, 409-413.

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

Hackman, G.S., & Biers, D.W. (1992). Team usability testing: Are two heads better than one? In Proceedings of the Human Factors Society 36th Annual Meeting. Santa Monica: HFS, 1205-1209.

Held, J.E., & Biers, D.W. (1992). Software usability testing: Do evaluator intervention and task structure make any difference? In Proceedings of the Human Factors Society 36th Annual Meeting. Santa Monica: HFS, 1215-1219.

Henderson, R., Podd, J., Smith, M., & Varala-Alvarez, H. (1995). An examination of four user-based software evaluation methods. Interacting with Computers, 7(4), 412-432.

Holleran, P.A. (1991). A methodological note on pitfalls in usability testing. Behaviour & Information Technology, 10(5), 345-357.

Jacobsen, N.E., Hertzum, M., & John, B.E. (1998). The evaluator effect in usability tests. In ACM CHI'98 Conference Summary. Reading, MA: Addison-Wesley, 255-256.

Jeffries, R. (1994). Usability problem reports: Helping evaluators communicate effectively with developers. In J. Nielsen & R.L. Mack (Eds.), Usability Inspection Methods. New York: Wiley, 273-294.

John, B.E., & Marks, S.J. (1997). Tracking the effectiveness of usability evaluation methods. Behaviour & Information Technology, 16(4/5), 188-202.

John, B.E., & Mashyna, M.M. (1997). Evaluating a multimedia authoring tool. Journal of the American Society for Information Science, 48(11), 1004-1022.

Jørgensen, A.H. (1989). Using the thinking-aloud method in system development. In G. Salvendy & M.J. Smith (Eds.), Designing and Using Human-Computer Interfaces and Knowledge Based Systems. Amsterdam: Elsevier Science Publishers, 743-750.

Karat, C. (1994). A comparison of user interface evaluation methods. In J. Nielsen & R.L. Mack (Eds.), Usability Inspection Methods. New York: Wiley, 203-233.

Karat, C.-M., Campbell, R., & Fiegel, T. (1992). Comparison of empirical testing and walkthrough methods in user interface evaluation. In Proceedings of the ACM CHI'92 Conference. Reading, MA: Addison-Wesley, 397-404.

Lewis, J.R. (1994). Sample sizes for usability studies: Additional considerations. Human Factors, 36(2), 368-378.

Nielsen, J. (1993). Usability Engineering. Boston: Academic Press.

Nielsen, J., & Landauer, T.K. (1993) A mathematical model of the finding of usability problems. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel, & T. White (Eds.), Proceedings of the InterCHI'93 Conference. New York: ACM, 206-213.

Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. In Proceedings of the ACM CHI'90 Conference. Reading, MA: Addison-Wesley, 249-256.

Ohnemus, K.R., & Biers, D.W. (1993). Retrospective versus concurrent thinking-out-loud in usability testing. In *Proceedings of the Human Factors and Ergonomics Society 37th Annual Meeting*, 2. Santa Monica: HFES, 1127-1131.

Pane, J.F., & Miller, P.L (1993). The ACSE multimedia science learning environment. In T.-W. Chan (Ed.), Proceedings of the 1993 International Conference on Computers in Education, (Taipei, Taiwan, December), 168-173.

Virzi, R.A. (1992). Refining the test phase of usability evaluation: How many subjects is enough? Human Factors, 34(4), 457-468.

Virzi, R. A., Sorce, J. F., & Herbert, L. B. (1993) A comparison of three usability evaluation methods: Heuristic, think-aloud, and performance-testing. In *Proceedings of the Human Factors and Ergonomic Society 37th Annual Meeting*. 1. Santa Monica, HFES, 309-313.