Backtracking Events as Indicators of Usability Problems in Creation-Oriented Applications

DAVID AKERS, University of Puget Sound
ROBIN JEFFRIES and MATTHEW SIMPSON, Google, Inc.
TERRY WINOGRAD, Stanford University

A diversity of user goals and strategies make creation-oriented applications such as word processors or photo-editors difficult to comprehensively test. Evaluating such applications requires testing a large pool of participants to capture the diversity of experience, but traditional usability testing can be prohibitively expensive. To address this problem, this article contributes a new usability evaluation method called backtracking analysis, designed to automate the process of detecting and characterizing usability problems in creation-oriented applications. The key insight is that interaction breakdowns in creation-oriented applications often manifest themselves in backtracking operations that can be automatically logged (e.g., undo and erase operations). Backtracking analysis synchronizes these events to contextual data such as screen capture video, helping the evaluator to characterize specific usability problems. The results from three experiments demonstrate that backtracking events can be effective indicators of usability problems in creation-oriented applications, and can yield a cost-effective alternative to traditional laboratory usability testing.

Categories and Subject Descriptors: H.5.2. [**Information interfaces and Presentation**]: User Interfaces—*Evaluation/methodology*

General Terms: Design, Information Systems

Additional Key Words and Phrases: Usability testing, backtracking analysis, undo, cost-effectiveness

ACM Reference Format:
Akers, D., Jeffries, R., Simpson, M., and Winograd, T. 2012. Backtracking events as indicators of usability problems in creation-oriented applications. ACM Trans. Comput.-Hum. Interact. 19, 2, Article 16 (July 2012), 40 pages.
DOI = 10.1145/2240156.2240164 http://doi.acm.org/10.1145/2240156.2240164

1. INTRODUCTION

Creation-oriented software applications such as photo editors, 3D modeling programs, and word processors can be difficult to comprehensively test with traditional laboratory usability testing methods. A diversity of creation goals and strategies causes different people to encounter completely different usability problems. This diversity translates into the need for a large pool of participants in order to identify a high percentage of the problems. However, testing a large pool of participants can be prohibitively expensive, due to the high costs of traditional, expert-moderated think-aloud usability testing.

D. Akers is now affiliated with Google, Inc.
Authors' addresses: D. Akers, Google, Inc., Mountain View, CA; email: dakers@google.com; R. Jeffries, Google, Inc., Mountain View, CA; email: jeffries@google.com; M. Simpson, Google, Inc., Mountain View, CA; email: msimpson@google.com; T. Winograd, Computer Science Dept., Stanford University, Stanford, CA; email: winograd@cs.stanford.edu.

DOI 10.1145/2240156.2240164 http://doi.acm.org/10.1145/2240156.2240164

To address this need, we introduce a usability evaluation method called *backtracking analysis* designed to automate the process of detecting and characterizing usability problems in creation-oriented applications. The key insight is that interaction breakdowns in creation-oriented applications often manifest themselves in simple backtracking operations that can be automatically logged (e.g., undo operations, erase operations, and abort operations). Backtracking analysis synchronizes these events to contextual data such as screen capture video, helping to characterize specific usability problems without requiring the active attention of a human moderator. A paired-participant retrospective discussion provides additional context for each backtracking event, providing further insight into the nature of the usability problems experienced. The primary claim of this article is that backtracking events can be effective indicators of usability problems in such applications, and that backtracking analysis provides a cost-effective alternative to traditional laboratory usability testing.

There are several research challenges faced in demonstrating this claim. First, simply detecting a backtracking event tells us nothing about the specific difficulty encountered by a user. One must find a way to collect the contextual information needed to characterize the nature of users' difficulties without compromising the cost-effectiveness of the approach. Second, it is clear that not all usability problems are accompanied by backtracking events, and that not every backtracking event indicates a usability problem. What if backtracking events yielded a low *hit rate* (percentage of usability problems detected), and a high *false alarm rate* (percentage of events that fail to indicate a usability problem) [Gray 1997; Swallow et al. 1997]? Finally, from a practical standpoint, it might be difficult to integrate backtracking analysis into the usability evaluator's arsenal of tools.

These challenges can be expressed in the form of three primary research questions addressed by this article.

$Q1$ (*Capturing context*). How do we design a usability testing protocol that provides sufficient contextual information to clarify the nature of usability problems associated with backtracking events?

$Q2$ (*Effectiveness*). How do backtracking events compare to other automatic indicators of usability problems in terms of the hit rate (percentage of usability problems identified) and the false alarm rate (percentage of events that fail to indicate usability problems).

$Q3$ (*Comparison to practice*). What are the strengths and weaknesses of backtracking analysis compared with traditional laboratory usability testing practices? (How do the methods compare in cost-effectiveness, and what types of problems are found by each method?)

To set the stage for these questions, Section 2 provides a description of the backtracking analysis usability evaluation method and indicates when, based on the experience reported in this article, the method is appropriate to use. Section 3 situates backtracking analysis within the usability evaluation literature, discussing related work.

Section 4 addresses Q1, investigating the contextual information required to interpret backtracking events and extract usability problems. We experimented with four different levels of context (screen capture video only, screen capture plus concurrent think aloud, screen capture plus retrospective think aloud, and screen capture plus paired-participant retrospective think aloud). The results of these studies suggest that the fourth configuration was substantially more useful than the other three for extracting usability problems from the episodes.

Section 5 describes two experiments that evaluate the effectiveness of backtracking analysis (Q2). These experiments compared backtracking analysis to the user-reported critical incident technique [Hartson and Castillo 1998], a cost-effective usability evaluation method in which participants self-report their difficulties. The first experiment evaluated the use of undo and erase events as indicators of usability problems in the Google SketchUp 3D modeling application, measuring an indicator's effectiveness by the numbers and types of usability problems discovered. For the 35 participants in the experiment, backtracking episodes revealed 5% more severe usability problems than participants self-reported. The false alarm rate for backtracking episodes was 27%, compared to just 1% for self-reporting, but all of the backtracking false alarms originated from erase events (none from undo events). It was surprising that backtracking analysis performed so comparably to self-reporting, a known cost-effective technique. To see whether this surprisingly strong result generalized to other applications, we repeated this experiment with the Adobe Photoshop application. In this second experiment, backtracking episodes identified the same number of severe problems as participants self-reported, and the false alarm rate for backtracking episodes was 4.9% (compared to 3.1% for self-reporting).

Section 6 describes an experiment to compare backtracking analysis with traditional laboratory usability testing (Q3). In a between-subjects study of Adobe Photoshop with 48 participants, we compared backtracking analysis with traditional usability testing conducted by a professional usability test moderator. The results indicate that backtracking analysis may be approximately twice as cost effective as traditional laboratory usability testing when testing with at least eight participants simultaneously. Backtracking analysis and traditional testing proved comparably effective at identifying problems related to choosing parameters, executing actions, and perceiving user interface state. However, backtracking analysis appears less effective at identifying problems related to feature discoverability, and possibly strategy formation.

Finally, Section 7 reflects on these findings, suggesting ways in which backtracking analysis might be combined with other methods such as other automatic detection methods, or traditional usability testing. It also suggests avenues for future research.

2. BACKTRACKING ANALYSIS

2.1 Description of the Method

The description of backtracking analysis is divided into three sections: steps to be taken before the study (2.1.1), during the study (2.1.2), and after the study (2.1.3).

2.1.1 Before the Study.

— *Instrument the software to automatically record backtracking commands (undo, erase, etc.).* To facilitate retrospective analysis centered around each event, it must be possible to record a timestamp associated with each command issued by a user. Depending on the software architecture, this may be relatively easy; many applications provide an extensibility API relatively early in their lifespan, and this is a natural "hook" from which to record events. If this is not possible, it is usually easy to instrument the program directly to record the events of interest (assuming that one has access to the application developers).

— *Decide how many participants to include in each testing session*. Each participant requires a separate computer, but the cost effectiveness of backtracking analysis increases substantially with the number of simultaneous participants (see Section 6).

— *Equip each computer with screen capture software, and software for conducting retrospective review of screen capture video episodes.* Please contact the authors if interested in obtaining the software used in our studies.

— *Design the testing tasks that participants will attempt.* Because participants will typically differ widely in how long it takes them to complete a task, make available "bonus" tasks to those who finish quickly. Otherwise, participants who finish early will experience boredom while waiting for the others to finish (and, if they visibly convey that they are finished, this may change the behavior of the others who are still working). When it is important to include a variety of testing tasks, it is advisable to provide different testing tasks to different participants. (This is especially important given the short task time, as follows.)

2.1.2 During the Study

— *Introduce the experiment.* As in any usability test, it is advisable to emphasize that the goal is to test the software, not the participants.
— *Administer testing tasks*. Each participant works simultaneously (but in isolation) on the set of testing tasks provided by the test moderator. While the participant works, record screen capture video along with time-stamped log events each time a participant uses a backtracking command. In a one hour session, we advise allotting 15-20 minutes for task time, to allow time for retrospective analysis.
— *Process the screen capture video to extract backtracking episodes*. Using the log events to index into the screen capture video, automatically extract short (∼20 sec.) video episodes centered around each backtracking event. If two episodes overlap (e.g., a user repeatedly used undo), merge the episodes to form a single longer episode. The individual backtracking events within each sequence can be marked with a caption in the video.
— *Pair up the participants, in preparation for retrospective discussions*. We advise matching those with the most episodes with those with the fewest, in an attempt to reduce the variance in time required for each pair to complete the retrospective review. If an odd number of participants are present in a session, the moderator can become the listener for the participant with the most episodes. Assign roles to each participant within a pair as "speaker" and "listener." (The participants swap roles halfway through and change computers.) The speaker's job is to watch her own episodes and attempt to answer the prompted questions. The listener's job is to ask follow-up questions until the answers are completely clear. The listener, not the speaker, is responsible for deciding when to move on to the next question. Encourage participants to use a shared mouse to point at objects (rather than using their fingers), so that it is possible to capture these references in the screen capture recording. For each 15 minutes of task time, we advise approximately 20 minutes of retrospectives.
— *Collect screen capture recordings of retrospective think-aloud commentary from each pair.* Our system provides a VCR-like interface to allow participants to explore each of the screen capture video episodes, automatically prompting them with a list of questions about each episode. Participants answer the questions by speaking into their headsets, clicking on a button to indicate when they finish answering each question. We recorded both screen capture video and audio from these retrospective sessions, for use in the post-study usability analysis. In our experiments, we asked participants to answer the following three questions about each episode.

(1) Please describe the events that led you to backtrack. Focus your answer on recounting a "play-by-play" of what you were thinking and doing at the time. If you can't remember, just say so and move on to the next episode.
(2) In the events leading up to your backtracking command, did the behavior of the software surprise you? If yes, explain the difference between your expectations and what actually happened.

(3) Did you find a way around the issue? If so, what did you do to get around it?

2.1.3 After the Study.

— *Analyze the retrospective commentary, identifying usability problems*. In our studies, we followed the extraction process described by Howarth et al. [2007]: we extracted usability problem instances, and merged these instances to form unique usability problems.
— *Rate the severity of each problem.* In our studies, we rated problems by their impact/persistence and frequency. See Section 5.2.2 for details on our approach.
— *Report the usability issues back to the developers.* We do not take a position on the best way to do this.

2.2 When to Use Backtracking Analysis

Like any usability evaluation method, backtracking analysis is useful only in certain circumstances, depending on the usability evaluation goals. One should consider using backtracking analysis in the following circumstances.

— *The application is creation-oriented.* That is, the goal of the application is to create content rather than to explore content. For creation-oriented applications, backtracking events may often indicate difficulty accomplishing a goal. (In contrast, in a non-creation-oriented application such as web search, backtracking might occur naturally as part of sensemaking.)
— *Backtracking events such as undo or erase are frequently used commands in the application.* This is not true of all creation-oriented applications. For example, the Adobe Lightroom application discourages the use of undo by visually exposing command parameters and allowing them to be retroactively modified.
— *The application can be instrumented to record time-stamped backtracking events.* Such event logs are necessary to facilitate the retrospective analysis of backtracking events. This often implies that the application has moved beyond the initial prototype stage, since early prototypes may not include support for backtracking, and may be difficult or impossible to instrument for event recording.
— *Usability evaluation goals require testing directly with end users.* Unlike usability inspection methods such as cognitive walkthroughs [Wharton et al. 1992] and heuristic evaluation [Nielsen and Molich 1990], backtracking analysis involves testing directly with end users of the software. Other methods seek to save costs by employing usability experts as reviewers of the interface.
— *Usability evaluation goals require a large participant pool in order to find a high percentage of the usability problems.* A large participant pool may be warranted when the goals require statistically significant results, when it is necessary to sample a variety of expertise levels [Law and Hvannberg 2004], when users are allowed the freedom to choose their own task goals [Spool and Schroeder 2001], or when there are many ways for users to accomplish the same goals (as with SketchUp and Photoshop).
— *Usability evaluation goals include the identification of problems related to choosing parameters, executing actions, and perceiving user interface state.* Backtracking analysis is not recommended for finding problems related to feature discoverability and strategy formation, since these problems do not appear likely to elicit backtracking behavior (see Section 6.4.2). If there are concerns regarding these types of problems, it is advisable to combine backtracking analysis with another method that identifies such problems.

— *Testing can be conducted in a laboratory environment.* Backtracking analysis has only been tested in the laboratory; future research may seek to adapt the technique in other settings (see Section 7.3).
— *Task goals can be clearly specified.* If task goals are not clearly specified, then backtracking may indicate exploration of design alternatives rather than failure to accomplish a goal. While we have not experimented with the use of less specific task goals, a theoretical discussion can be found in Akers [2010, Chapter 5].

3. RELATED WORK

Ivory and Hearst [2001] classified usability evaluation methods into five types: testing methods (e.g., traditional laboratory usability tests [Dumas and Redish 1999; Rubin and Chisnell 2008], remote usability tests [Hartson et al. 1996], log file analysis [Hilbert and Redmiles 2000], and A/B tests [Kohavi et al. 2007]), inspection methods (e.g., Cognitive Walkthroughs [Wharton et al. 1992], heuristic evaluations [Nielsen and Molich 1990], and pluralistic walkthroughs [Bias 1991]), inquiry methods (e.g., contextual inquiry, questionnaires, or interviews), analytical modeling methods (e.g., GOMS [Card et al. 1983] and CogTool [John et al. 2004]), and simulation methods (e.g., Petri net models [Rauterberg 1995] and information processing models [Robertson et al. 1989]). Backtracking analysis falls into the first category (testing methods); the technique involves performing detailed analysis of end-users, in an attempt to capture the context necessary to characterize the nature of usability problems.

Backtracking analysis is a derivative of the critical incident technique Flanagan [1954], in which significant events during interaction (positive or negative) are collected and analyzed by trained observers. (We chose to focus on negative incidents in this work, since they are more likely to indicate usability problems.) Del Galdo et al. [1986] adapted the critical incident technique for use in HCI, and Winograd and Flores [1985] independently developed the theory of user breakdowns. Hartson and Castillo [1998] devised the user-reported critical incident technique, in which the users of the system are responsible for detecting and describing their own critical incidents as they occur. While their technique was designed for use in remote situations, some of their studies were performed in the lab. The comparison of our approach with this technique forms the central theme of Section 5.

Capra [2002] developed and evaluated an augmented retrospective variant of the user-reported critical incident technique, finding it to be similarly effective to a contemporaneous reporting strategy. In this variant, researchers showed participants a video replay of their entire session, asking them to detect and describe critical incidents as they observed them in the replay. Backtracking analysis uses a hybrid implementation of Hartson and Castillo [1998] and Capra [2002], separating the detection and description phases. Participants detect critical incidents contemporaneously, but description is delayed until a retrospective phase. We chose this approach to avoid unnecessarily interrupting participants during their tasks.

One difficulty Hartson and Castillo found with their first implementation of the user-reported critical incident technique was that users often initiated the reporting long after they experienced a problem, making it more difficult to link the reported incident to contemporaneous video or other context. The solution described in their paper was to decouple the detection and description of incidents, which is exactly what our own implementation does by combining contemporaneous detection of incidents with retrospective description.

A legitimate concern for any usability evaluation method that relies extensively on retrospective commentary is whether important information is lost due to the fallibility of human memory. We are encouraged by results from a study comparing current

and retrospective think aloud protocols for usability evaluation [Van Den Haak and De Jong 2003], which found that these protocols produced similar outcomes when testing an online library catalog. A recent eye-tracking study by Guan et al. [2006] provides further evidence that retrospective protocols are effective.

The pairing of participants during retrospective sessions also builds upon prior work. O'Malley et al. [1984] first described the idea of pairing participants during the task phase of think-aloud usability testing, coining the term "constructive interaction". A more recent study by Hackman and Biers [1992] discovered that participants generated more think aloud comments when working in pairs during a test. Note that in constructive interaction participants are paired during the task, whereas in backtracking analysis participants are paired only for the retrospective. (It would be reasonable to try constructive interaction in place of retrospective pairing, but this is left as future work.) While we certainly cannot claim to have invented the idea of paired-participant usability studies, we did find a unique application of this approach that proved to be surprisingly beneficial.

Our focus on backtracking events is closely related to a case study described by Swallow et al. [1997]. They instrumented a direct-manipulation visual builder application to record log data for a variety of indicators of critical incidents, including undo and erase. However, their study focused mainly on the hit rate for each indicator (the rate at which each indicator revealed actual usability problems). The authors performed no severity analysis of problems, and did not compare their methods with any other usability evaluation methods, as we do in Sections 5 and 6.

As in many recent studies [Bruun et al. 2009; Hartson et al. 2000], this article measures the effectiveness of usability evaluation methods by estimating the number and type of usability problems that each method identifies. But as Wixon [2003] has observed, it does not matter how many problems one finds if these problems do not get fixed in the software. During the past decade, Wixon and others have advocated alternative metrics for success that more faithfully capture the "downstream utility" of a usability evaluation method [John and Marks 1997]: how often do the findings of a method lead to actual design changes that improve the software? A key element of downstream utility not addressed by this article is persuasiveness; how effectively does a usability evaluation method communicate its findings to developers, convincing them to instigate changes? This is left as future work.

4. CAPTURING CONTEXT FOR BACKTRACKING EVENTS

Backtracking events, like any other event-based indicator of usability problems, are not useful without the contextual information needed to interpret the meaning of these events. For example, a usability evaluator would want to answer contextual questions about any backtracking episode, in order to classify specific usability problems that might have occurred.

— What happened before and after the backtracking event?
— Was the user surprised by the behavior of the software? If so, how?
— If there was a problem, was the user able to recover? How?

These types of questions are relatively easy to answer when a human moderator is present during the test. If there is uncertainty about some aspect of a user's problem, the moderator can just ask questions about it, either during the task or afterwards during a retrospective review. However, this approach does not scale well; we would be limited to testing one participant at a time.

The solution employed by backtracking analysis is to pair up participants and ask them to answer questions about the context of each backtracking episode, prompted

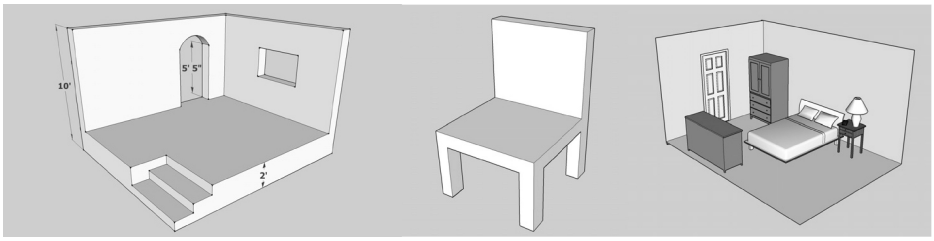

Fig. 1. Usability testing tasks for testing with SketchUp. In the "room" task (left), we asked participants to model the room, including the specified dimensions. In the "chair" task (middle), we asked participants to model the chair, ensuring that its legs were the same height and shape. In the "furniture" task (right), we asked participants to arrange the pre-made furniture within this room.

by screen capture video of the episode. We also considered three other approaches for capturing context, which vary according to the quality of the contextual information gathered and the participant time required to collect this information. We conducted a series of exploratory experiments with each of these alternate approaches, concluding that only the paired-participants approach provided the richness of data necessary for interpreting backtracking episodes.

4.1.1 Experiment 1: Screen Capture Video. In the first experiment, we tried synchronizing backtracking events with screen capture video recordings made during a laboratory usability test of Google SketchUp. This approach was attractive because it requires no additional time or effort on the part of participants. We recruited 54 participants of varying SketchUp expertise, bringing them to a laboratory in groups of 10-15 at a time. During a 90 minute usability test, we asked them to attempt three simple SketchUp tasks: a "room" task, a "chair" task, and a "furniture task" (see Figure 1). Participants worked side by side, and their backtracking commands were logged using a Ruby plug-in (see Akers [2010, Appendix C]).

At the end of the test, we used the time-stamped backtracking events to index into the screen capture video, and showed the screen capture episodes to two SketchUp user interface designers. We discovered that knowing the users' end goals and viewing a recording of their actions was helpful, but often insufficient to identify specific usability problems. In nearly 50% of the episodes, it seemed likely that there was some usability problem, but unclear what the problem might have been. These numbers motivated the search for a more effective method to characterize the usability problems that many of these backtracking events seemed to identify.

4.1.2 Experiment 2: Screen Capture Video + Concurrent Think Aloud. The next experiment required participants to think aloud as they worked; it was expected that the audio recording would provide the necessary context to interpret backtracking episodes. As in the first experiment, this approach required no additional time from participants (assuming that the verbalization would not slow them down). The results of this experiment showed that participants often forgot to verbalize their thoughts, and the commentary that they did provide was often broken and terse. Moreover, asking participants to think aloud seemed to have detracted from their ability to model successfully. (We are unsure why this might have occurred.) Since the experiments were conducted in large groups, it was not feasible to individually remind participants to think aloud as they worked, which may have contributed to the reticence we observed.

4.1.3 Experiment 3: Screen Capture Video + Retrospective Think Aloud. In this experiment, the computer automatically generated short screen-capture episodes centered around

each backtracking event. The computer prompted participants with three questions about each episode (see Section 2.1.2). Participants answered these questions by speaking aloud into a microphone, alongside other participants who were doing the same.

The results of this study revealed that even this approach was insufficient. Participants' commentary was often abrupt, and generally did not help in describing the underlying usability problems. They reported that it felt awkward talking to their computer, especially when sitting alongside others who were doing the same. There were sometimes awkward silences in the room when all of the participants happened to fall silent simultaneously.

Another drawback of retrospective analysis is its increased cost, requiring substantial time for participants to answer the questions about each episode. Complicating matters, the cost proved to be highly variable; those with only a few backtracking episodes finished quickly, while those with many episodes required as much as twice the amount of time spent attempting the testing task. Since we typically compensate each participant the same amount for taking part in a study, the uneven distribution forced us to increase the standard compensation amount to account for outliers.

4.1.4 Experiment 4: Screen Capture + Paired Retrospective. The final exploratory study paired up participants and asked them to discuss the questions together during the retrospective. This is the approach used in backtracking analysis; it is summarized in Section 2.1.2.

The result was a success; the commentary was greatly improved over the previous experiments. Interestingly, listener participants rarely asked follow-up questions, but their mere presence seemed to have changed the way that speakers responded to the questions. Of the four techniques that we experimented with, only paired-participant retrospectives approached the quality of a human-moderator working one-on-one with each individual participant.

Pairing up participants for the retrospective analysis does increase the costs further, since each participant must participate in two separate retrospective sessions (once as a speaker, and once as a listener). However, by carefully pairing up participants, it is possible to substantially reduce variability between pairs.

5. EFFECTIVENESS

This section asks how backtracking events compare in effectiveness to other automatic indicators of usability problems. We define effectiveness as a combination of the hit rate (percentage of usability problems identified) and false alarm rate (percentage of events that fail to indicate usability problems). Section 5.1 elaborates upon the motivation for the studies of effectiveness. Sections 5.2 and 5.3 describe two empirical studies that attempt to answer the question.

5.1 Study Motivation

It is not inherently obvious that backtracking events would make useful indicators of usability problems. First, consider that usability problems can take on multiple forms. One can experience difficulty while planning a sequence of actions, translating one's intent into an action, physically executing an action, or evaluating that action's success [Norman 1986]. It is not immediately clear which of these types of problems would be indicated by backtracking events.

Second, it is also important to acknowledge that backtracking serves more than one function in creation-oriented applications; backtracking events might not always indicate usability problems. For example, in addition to helping users to recover from errors, backtracking makes it easier to explore and learn the functionality of an

interface. Learning difficulties can sometimes be attributed to usability problems in the interface, but in some cases it is easier to learn by experimenting than by reading a user manual. Such examples cast doubt on the effectiveness of backtracking events as indicators of usability problems. Perhaps most backtracking events would indicate false alarms from a usability perspective?

Accordingly, the studies described in Sections 5.2 and 5.3 address the following research question.

*Q*2. How do backtracking events compare to the other indicators of usability problems in terms of the hit rate (percentage of usability problems identified) and the false alarm rate (percentage of events that fail to indicate a usability problem)?

To address this question, we chose to compare backtracking analysis with the user-reported critical incident technique [Hartson and Castillo 1998], a usability evaluation method in which participants report their own usability difficulties. We chose this method for comparison because it has been found to be cost-effective compared to traditional usability testing [Bruun et al. 2009] and is similar enough to backtracking analysis in its procedure to allow for a tightly-controlled comparison.

5.2 Comparison to Self-Reporting: Google SketchUp

This first study (published in Akers et al. [2009]) compared backtracking analysis with the user-reported critical incident technique, using Google SketchUp as the test application. The following sections detail the experimental procedure, data processing, and results. A more detailed discussion of the results and limitations is deferred until Section 5.5.

5.2.1 Experimental Design and Recruitment of Participants.. We used a within-subjects design to compare usability testing methods; participants reported problems as they worked (self-report data), and we simultaneously logged their undo and erase commands (backtracking data). The retrospective interviews included both self-report and backtracking screen-capture video episodes. We chose a within-subjects design in order to eliminate variance from individual differences, and because it allowed us to ask participants to speculate on why they failed to report problems that were detected by backtracking analysis (see Section 5.2.4).

To facilitate the experiment, we instrumented SketchUp to record time-stamped occurrences of undo and erase events, and added an on-screen button for participants to report critical incidents. As in the exploratory studies of Section 4, we used SketchUp's embedded ruby api to implement these extensions, thereby avoiding having to make any modifications to the source code to SketchUp.

There was one experimental trial per participant. Each participant attended one of six 90-minute group usability testing sessions, each consisting of 5–7 participants who worked in parallel on independent laptops. Each of the laptops was an IBM ThinkPad T61p, with identical software configurations including a development version of SketchUp. Laptops were also equipped with screen capture recording software and dual headsets with microphones for the paired-participant retrospectives.

The primary goal in recruitment was to attract participants of a variety of backgrounds and SketchUp expertise levels. This variety increased the generality of the study, and made inter-group comparisons possible. We recruited a total of 43 participants, of whom 35 provided usable data (see Section 5.2.3). Of these 35 participants, most (29) responded to flyers posted at coffee shops and in academic buildings at the university of colorado at boulder. To attract a higher percentage of SketchUp experts, we also enlisted six employees of Google who are specialists in 3D modeling with SketchUp.

Of the 35 participants in the experiment, 19/35 (54%) were a near-equal mix of undergraduate and graduate students from a wide range of academic departments. Of the participants, 10/35 (28%) had never used SketchUp before, 9/35 (26%) described themselves as novices with the interface, 9/35 (26%) described themselves as intermediate users, and 7/35 (20%) described themselves as experts. For a 90 minute session, participants were compensated with a $10 gift check, a short-term license to SketchUp pro, and Google Swag.

5.2.2 Usability Testing Protocol. Each 90-minute session was divided into the following sections: training in SketchUp (15 minutes), training in identifying critical incidents (20 minutes), practice (10 minutes), modeling task (15 minutes), and retrospective commentary (30 minutes).

Training in SketchUp (15 minutes). To familiarize participants with SketchUp, participants watched three previously produced new-user training videos.[1] The three videos were: "New Users 1: Concepts," "New Users 2: Drawing Shapes," and "New Users 3: Push/Pull." participants were encouraged to take notes.

Training in Identifying Critical Incidents (20 minutes). To ensure that participants were adequately trained in reporting critical incidents, we gave extensive instructions and examples of incidents. For the purposes of a fair comparison, this study attempted to mimic the style and content of the training described by Hartson and Castillo [1998]. We did make several changes motivated by an early pilot experiment. This experiment (n = 12) revealed that participants seemed less likely to report problems when they attributed the problems to themselves (rather than the software). We adjusted the instructions to emphasize that we were testing the software, not the participants. We also decided to refer to critical incidents as "interface issues," hypothesizing that the more neutral terminology would encourage reporting. (We did not test this hypothesis, however.)

Practice (10 minutes). Participants were given 10 minutes to practice using SketchUp and reporting critical incidents. Participants were told to explore interface features and build whatever they wanted during these 10 minutes.

Modeling Task (15 minutes). We randomly assigned participants to one of two tasks: some completed the bridge task (Figure 2, top), while others completed the room task (Figure 2, bottom). Having a second task increased the generality of the study, but there was not enough time to give both tasks to each participant. Each task had two phases; if participants finished the first phase (shown at left), they could raise their hand and receive printed instructions for the second, more difficult phase (shown at right). This was intended to keep all participants busy throughout the session, regardless of their expertise level in SketchUp. See the caption for Figure 2 for descriptions of the specific instructions provided to participants.

We asked participants to report critical incidents as they worked. To report an incident, they clicked a "Report an Issue" button. Pressing this button triggered a log message that was written to a file; it had no visible impact to the user. After the task was finished, an automated system extracted video episodes around each marked event and prompted participants to reflect on the episodes (see the next section on "Retrospective Commentary"). Here we deviated from the approach of Hartson and Castillo [1998], in which the user was asked to fill out a form immediately upon experiencing each incident. We chose to delay the commentary in order to minimize disruption to the

[1]http://www.youtube.com/user/SketchUpvideo.

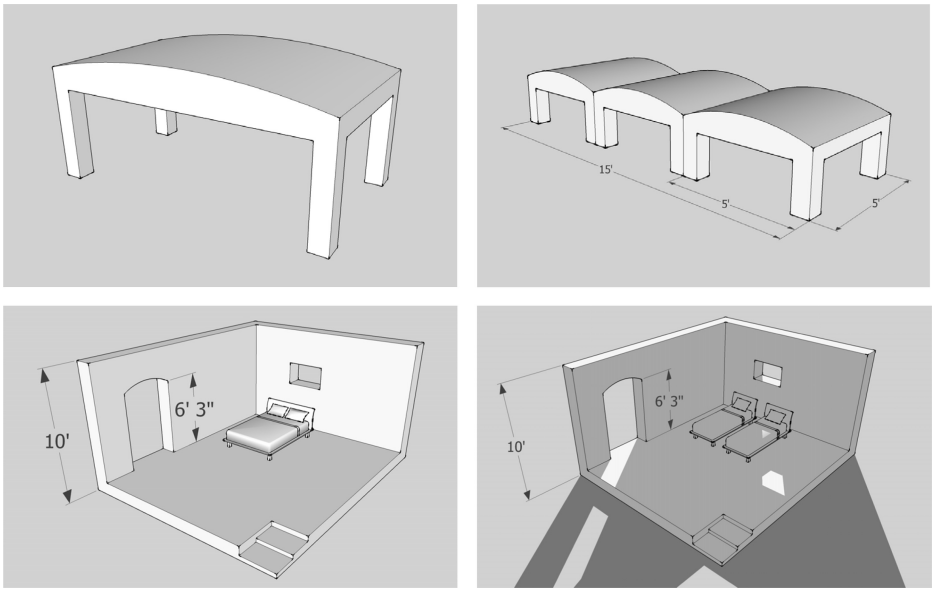

Fig. 2. The two tasks used in the laboratory study of Google SketchUp. In the "bridge" task (top, left), we asked participants to make all four legs the same height and shape. If participants finished early, they were asked to resize the bridge to 5 ft. x 5 ft. and make three copies of it, laying them end to end (top, right). In the "room" task (bottom, left), we asked participants to ensure that the room was 10 ft. high, and that the doorway was 6 ft. 3 in. high. They did not need to model the bed; they could insert it from the "components browser" and position it in the room. If participants finished early, they were asked to modify the bed to form two single beds, and add shadows (bottom, right).

user during the task, and to facilitate a fair comparison with backtracking events (for which commentary must be collected retrospectively to avoid intolerable disruption).

Retrospective Commentary (30 minutes). The retrospective session proceeded according to the plan described in backtracking analysis, but included both backtracking and self-report episodes. The first three questions were the standard questions from backtracking analysis (see Section 2.1.2). For undo and erase episodes, we asked two additional questions specific to this experiment.

(1) Did you report this as an issue?
(2) If you did not report this as an issue, why do you think that you didn't?

Since there were different questions for each event type, the system displayed all of the episodes of each event type together, rather than interleaving them. To avoid confounding the results, we fully counterbalanced the order of the event types.

5.2.3 Usability Problem Identification. This section describes the analysis process employed to extract usability problems from the raw usability data. The extraction process followed that of Howarth et al. [2007]: we extracted usability problem instances, and merged these instances to form unique usability problems. But before we began the extraction and merging process, we took several steps to prepare the data, described as follows.

Step 1: Discarding participants whose data were unusable. From an original set of 43 participants, we removed three participants because their microphones failed to work properly. We also removed one more participant because he did not finish his

retrospective during the time allotted. We decided to remove three more participants because they had been paired with the experimenter; we would like to run further studies to evaluate how the quality of the commentary might differ in these cases. Finally, we removed one participant because she could not even begin to answer the questions about her episodes. (She was utterly lost with SketchUp. Her usability problems were more of the "how do i use a mouse?" variety.) Removing the data from these 8 participants left 35 participants, whose data proceeded to the next phase of analysis.

Step 2: Extracting episodes and retrospective commentary. The 35 participants produced 353 episodes (139 undo episodes, 113 erase episodes, and 101 self-report episodes). This equates to an average of 10.1 episodes per person, or 0.67 episodes per minute of SketchUp usage. The system automatically extracted these episodes, and the associated retrospective commentary.

Step 3(a): Discarding unclear episodes. From this initial set of 353 episodes, we discarded 25 episodes (7%) because the combination of commentary and screen capture video was not clear enough for the researcher to extract a complete usability problem instance. We discarded an additional 4 episodes (1%) in which the user could not remember enough about the episode to answer any of the retrospective questions. This left 324 episodes (129 undo episodes, 103 erase episodes, and 92 self-report episodes).

Step 3(b): Discarding false alarms. We identified 64 episodes (19.8%) that contained no identifiable usability problems. All but one of these "false alarm" episodes (98%) were triggered by erase events. Surprisingly, there were no false alarms triggered by undo events. Only one false alarm was generated by self report, when a participant accidentally pressed the button. This indicates that the overall false alarm rate for backtracking episodes was 63/232 = 27%.

There were two varieties of erase false alarms. First, there were episodes in which a user erased an extra edge that was a byproduct of the normal modeling process. (This is specific to SketchUp; most other 3D drawing programs do not produce such edges.) Second, there were episodes in which users created temporary construction lines to help them align multiple pieces of geometry, and then erased these lines when they were finished. Interestingly, this example could never have resulted in an undo operation; there is no way to undo a temporary construction line without also undoing the alignment action that follows it. This seems to be a general difference between undo and erase, and would likely hold true for other applications besides SketchUp.

Step 3(c): Identifying usability problem instances. After all of the data preparation steps described previously, 260 episodes remained. A single researcher analyzed these episodes to extract 215 usability problem instances, using the definition of usability problems provided by Jacobsen et al. [1998]. The mapping from episodes to usability problems instances was many-to-many, as described next.

Sometimes, a single episode would correspond to multiple usability problem instances. In identifying usability problem instances, we included both problems that were found directly by a method, and those that were "incidental" to the method. For example, if a user experienced some problem, pressed undo because of the problem, and then experienced a second problem unrelated to the first, we would include both problem instances. This process produced 35 additional problem instances.

Sometimes, a single usability problem instance would correspond to multiple episodes. This happened only when multiple episodes overlapped in content. Of course, episodes of the same event type cannot overlap in the screen capture video (since otherwise the process would have merged them into a single longer episode). However,

grouping the retrospective by event type necessitated that we avoid merging events of different event types. Therefore, it was possible for episodes of different event types to overlap in content. When participants saw the same interaction sequence for a second time, their responses to questions during the retrospective session were likely to be terse. ("I have already talked about that; refer to my previous answers.") We resolved such situations as follows: if a problem was mentioned during the commentary for an episode, and that same problem was visible in the video of an overlapping episode, then we counted it as a single problem instance discovered in both episodes. There were 50 pairs of partially overlapping episodes, and 20 triples.

Step 4: Merging usability problem instances. Next, the researcher merged the 215 problem instances to form 95 unique usability problems. It is critical that we applied a consistent merging strategy across all problems. Merging two problems requires generalization, since no two problem instances are exactly the same. Problem instances may differ along many dimensions: for example, the level of granularity of the problem, the immediate cause of the problem, the circumstances under which the problem occurred, the consequences of the problem, etc. We adopted a conservative merging strategy, merging problems only if they differed superficially. Nevertheless, the merge rate (2.26:1) was higher than we expected. This may be due to the degree of specificity of the task goals; different users working on the same task tended to experience the same problems because they were all working toward identical goals.

Finally, the researcher wrote descriptions for each of the 95 usability problems. In describing each problem, the primary goal was to record what happened in the episode(s), and what the user said about what happened. Examples of problem descriptions include the following.

— After creating a hole, one user judged the result by what he could see through the hole. Because the background (the other side of the hole) was similar to the material surrounding the hole, he had low confidence in his success and spent 10 seconds making sure that the action had the intended effect. *(Found by self-report only)*.
— One user experienced difficulty resizing a rectangle with the move/copy tool. He said that he was surprised that it distorted into non-rectangular shapes as he dragged on an edge. He expected that SketchUp would remember that this shape was created as a rectangle, and keep that rectangle constraint through the rest of the modeling process. He worked around the problem by reversing his action and redrawing the rectangle in the new shape. *(Found by undo only)* .
— Several users experienced difficulty when they tried to copy and paste a rectangle, and align their copy to a point on an existing rectangle. The paste operation automatically triggered a "move" command on the copied geometry, selecting a particular corner on the copied rectangle as the anchor point for the move. Users could not find a way to "snap" the copied rectangle into alignment with the edges of the target rectangle, since the anchor point did not correspond to any point on the existing rectangle. They could not find a workaround, and ended up with unaligned geometry. *(Found by all three methods)*

The full list of usability problems can be found in Akers [2010, Appendix A]. Note that we did not attempt to infer the root causes of these difficulties, which can require complex causal reasoning [Koenemann-Belliveau et al. 1994]. Was the training video unclear? Did users' expectations stem from their prior use of other 3D modeling software? This study avoided speculation on such possibilities; its goal was to provide designers and developers with as much information as possible to evaluate the design tradeoffs inherent in addressing the problems.

Table I. Problem Severity Rating Scales Used in the SketchUp Experiment

Problem impact and persistence	
(1)	minor annoyance, easily learned or worked around
(2)	bigger problem (at least 3 minutes lost), but easily learned or worked around
(3)	minor annoyance, but will happen repeatedly
(4)	bigger problem (at least 3 minutes of time lost), and will happen repeatedly
(5)	showstopper (can't move forward without help; data loss; wrong result not noticed)
Problem frequency	
(1)	problem will be extremely rare (less than 1/100)
(2)	some will encounter (at least 1/100, less than 1/3)
(3)	many will encounter (at least 1/3, less than 2/3)
(4)	most will encounter (at least 2/3, less than 100%)
(5)	everyone will encounter (*e.g.*, startup problem)

Coding for Problem Severity. Three independent raters (three knowledgeable SketchUp users, including two user interface designers from Google) coded each of the 95 usability problems for severity. Raters evaluated each problem for its estimated frequency in the general population (rated on a scale of 1–5), and a combination of its estimated impact and persistence (also rated on a scale of 1–5). We modeled these scales roughly after those of Karat et al. [1992], but chose to use five-point scales rather than three-point scales to capture more resolution in the ratings. Frequency was defined as the percentage of occurrence in the general population during an average modeling session. Impact was defined as the time it would take to recover from the problem, while persistence was defined as the extent to which the problem recurred or was difficult to work-around. The actual scales are shown in Table I.

Frequency and impact/persistence ratings were added, and the final severity rating was obtained by reducing this sum by one; this produced an ordinal scale from 1 (least severe) to 9 (most severe). We labeled severity as follows: 1–2: mild; 3–4: medium; 5–9: severe (but 9 was never observed). We divided the 95 problems into three sets: a training set (15 problems), a test set (10 problems), and an independent set (70 problems). Coders used the training set to discuss the severity scales and resolve differences in coding styles. Next, they independently rated the 10 problems from the test set, and then discussed the differences and adjusted their ratings.

Before the discussion, Cronbach's Alpha [Cronbach 1951] was 0.75; after coders adjusted their ratings, it increased to 0.90. Next, they independently rated the remaining 70 problems. For the final set of all 95 problems (using the adjusted ratings from the test set), Cronbach's Alpha was 0.82, indicating strong agreement amongst the coders. To reduce the effect of individual outliers, we chose the median of the three ratings as the severity statistic for each problem.

5.2.4 Results.

Comparison among Undo, Erase, and Self-Report. We define that a problem is detected by a method if at least one participant experienced an instance of the problem, as evidenced by video episodes and retrospective commentary associated with that method (undo, erase, or self report). We define that a problem is detected by a set of

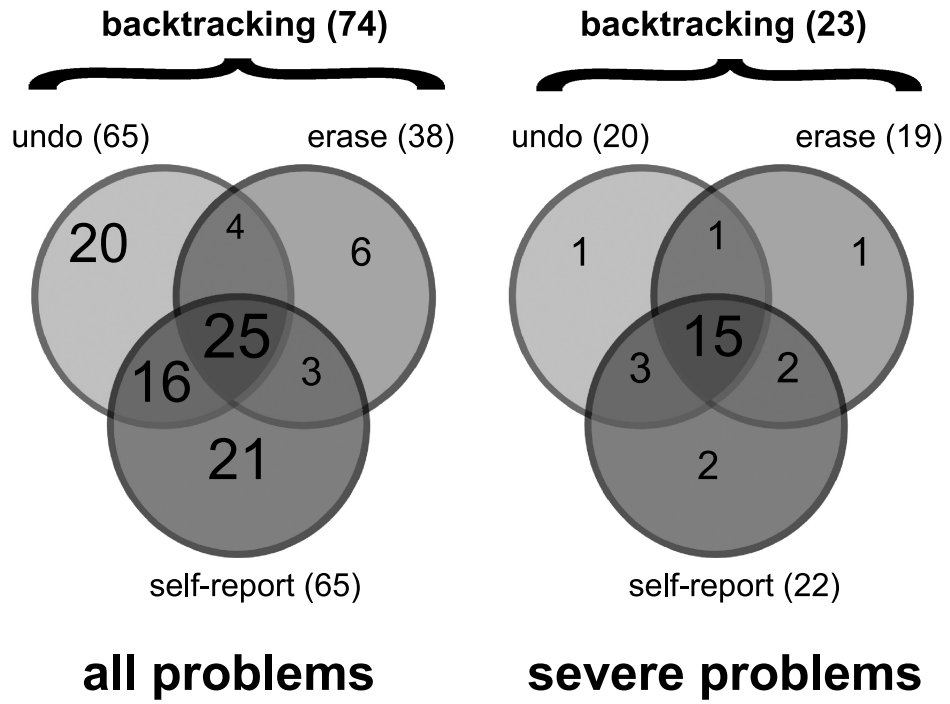

Fig. 3. Two Venn diagrams depicting the number of usability problems detected in Google SketchUp by each of the three methods. The left diagram shows the results for all problems, while the right diagram focuses on problems rated as severe. Problems in the middle of each Venn diagram were detected by all three methods, while those on the outsides were detected by only one method. Note that undo and erase combined to detect more severe problems (23) than self-report (22).

methods if the given statement is true for each method in the set (even if no particular participant would have contributed evidence of the problem from all of the methods in the set).

Figure 3 depicts the number of problems detected by each method or set of methods. On the left are the results for all problems, while on the right are the results for problems rated as severe (those whose severity rating is at least 5). In each figure, problems identified by only one method are non-overlapping, while those that were found by a set of methods are depicted as overlapping with the other methods. While there is substantial overlap amongst the indicators, only 25 problems (26%) were detected by all three indicators.

To compare these results with those discussed in the literature, we estimated the average probability (λ) of a participant finding the average usability problem for backtracking analysis ($\lambda = 0.042$), self-reporting ($\lambda = 0.027$), and both methods ($\lambda = 0.055$). Using a Poisson model [Nielsen and Landauer 1993], these values of λ imply that one would need over 30 participants to discover 85% of the usability problems, even if using a combination of backtracking analysis and self-reporting. This is hardly consistent with the "magic number five" rule of thumb which claims that for many usability tests, one can test only five participants and find 85% of the problems [Lewis 1994; Nielsen and Landauer 1993; Virzi 1992]. Our numbers are more comparable to a recent study of four web applications [Spool and Schroeder 2001] ($\lambda \approx 0.1$), which also questioned the universal applicability of the "magic number five" rule. It is difficult to attribute these differences to specific aspects of the applications being tested, since

the value of λ also depends on the usability evaluation method employed and the tasks used in the study, among other factors. Nevertheless, it seems likely that the diversity of strategies available in SketchUp increased the diversity of usability problems encountered by participants (just as the diversity of semantic content likely contributes to the diversity of problems experienced in Web applications).

Reasons Problems Were Not Reported. Consider the problems in the top three sections of the Venn diagram: those that were detected by erase and/or undo, but not detected by self-reporting. Why would people fail to report problems, when these problems were detected with other techniques? To begin to answer this question, we assessed the data collected on question #5 in the retrospective session: for erase and undo events that were not reported, why did the participant think that he did not report it? Of those times when people ventured to speculate, the explanations were revealing: 30/52 (58%) said that they did not report the problem because they blamed themselves rather than SketchUp. (This happened despite repeated attempts to emphasize to participants that they should disregard the attribution of blame.) Another 16/52 (31%) said that the problem was too minor to report. The remaining 6/52 (11%) said that they should have reported it, but simply forgot. While it is easy to draw conclusions from these numbers, it is important not to over-interpret; people are notoriously bad at introspecting about their own high-level cognitive processes [Nisbett and Wilson 1977]. However, the data combined with the subjective comments warrant further investigation into the reasons people do and do not report problems.

5.3 Comparison to Self-Reporting: Adobe Photoshop

Encouraged by the positive results from the Google SketchUp study, we conducted a second experiment to see if these results would generalize to another application, Adobe Photoshop. Photoshop is an enormous application, with many use cases: creating digital artwork, retouching photographs, authoring flyers/posters, etc. Since it was impossible to test all of these use cases in a single usability study, we decided to focus the evaluation on Photoshop's facilities for basic image retouching. The design of this study was quite similar to the previous; the following sections describe only the important differences.

5.3.1 Recruitment. For this study, we recruited 28 participants, of whom 24 provided usable data (see Section 5.3.3). These participants covered all different experience levels with image manipulation in Photoshop. All participants responded to flyers posted in academic buildings at Stanford University.

Of the 24 participants in the experiment, there were 15 graduate students, 8 undergraduate students, and one software engineer. Before beginning the experiment, participants described their prior experience retouching images in Photoshop (fixing colors, removing imperfections, etc.). Of the 24 participants, 3/24 (13%) said that they had never used Photoshop for this purpose, 11/24 (46%) described themselves as novices, 6/24 (25%) described themselves as intermediate users, and 4/24 (17%) described themselves as experts. As compensation for a 90 minute session, each participant received $20.

5.3.2 Usability Testing Procedure. As with the SketchUp experiment, there was only one experimental trial per participant, and the experiment was divided into 90-minute sessions. Due to laboratory space constraints, we recruited participants in smaller groups (2 at a time, rather than the 5–7 in the SketchUp experiment). Each participant worked on an identically configured installation of Photoshop CS3. The workspace was configured according to the software defaults, with a few exceptions: we enabled the

Fig. 4. The "tulips" task in the Adobe Photoshop usability test. Beginning with the image on the left, participants first rotated and cropped the image. If they finished early, they attempted to increase the saturation of the tulips, emphasize highlights on the statue, and change a tulip's color from yellow to red. [*Photo by Andrew Faulkner, af.studio.com*]

history palette, placing it between the navigation and color palettes. Also, we changed the tool palette from one-column mode to two-column mode, to match the format used in the video tutorial.

The instrumentation of Photoshop differed somewhat from that of SketchUp. Since the tasks in this study were modification-oriented rather than creation-oriented, erase was not a useful command to the participants. After pilot testing found no instances of erase events, we decided to simplify the protocol instructions by removing erase events from the retrospective session. We instrumented Photoshop to record undo operations using the "history log" feature, thereby avoiding having to make any modifications to the source code (see Akers [2010, Appendix C]).

The 90 minute experiment was divided into the same five sections as in the SketchUp experiment: training in Photoshop (15 minutes), training in identifying critical incidents (20 minutes), practice (10 minutes), modeling task (15 minutes), and retrospective commentary (30 minutes).

Training in Photoshop (15 minutes). To familiarize everyone with the basic layout of the Photoshop interface, participants watched one 15-minute training video prepared by the first author. The training video was modeled after the first chapter of the Adobe Photoshop "Classroom in a book" [Faulkner and Walthers Von Alten 2007]. This video covers a basic introduction to the tools, image adjustments, palettes, and filters. It also includes the help system and undo functionality. A full transcript is included in Akers [2010, Appendix B].

Training in Identifying Critical Incidents (20 minutes). The training was similar to what was provided for SketchUp. a transcript of the training video is included in Akers [2010, Appendix B].

Practice (10 minutes). Participants were given 10 minutes to practice using Photoshop and reporting critical incidents. Participants were provided with a "rubber duck" image (the same image used in the training video), and were allowed to freely explore the interface during this time.

Modeling Task (15 minutes). We randomly assigned participants to one of two tasks: some completed the "tulips" task (Figure 4), while others completed the

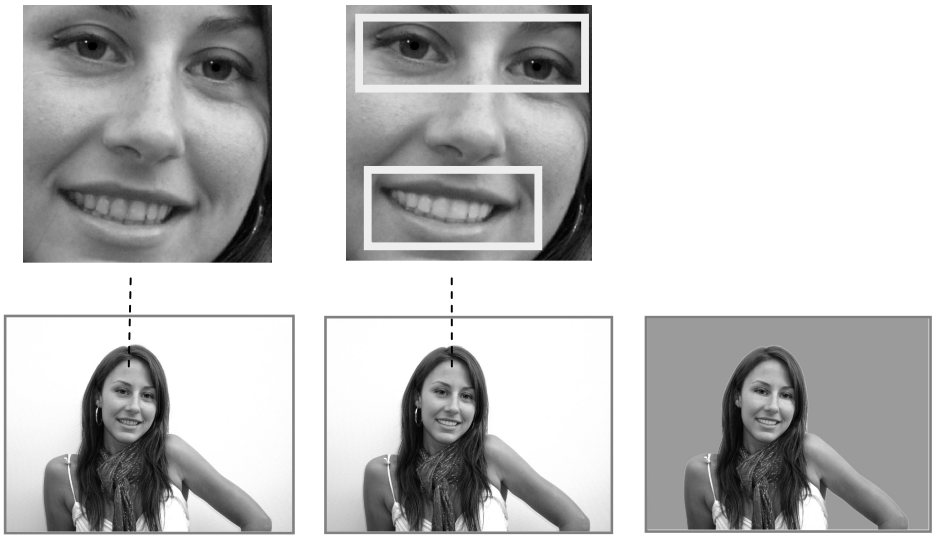

Fig. 5. The "portrait" task in the Adobe Photoshop usability test. Beginning with the image on the left, participants first changed the eye color from brown to blue, and brightened the teeth. If they finished early, they removed both earrings, reduced eye shadows, and changed the background color from white to grey. [*Photo by Rick Hawkins*]

"portrait" task (Figure 5). As in the SketchUp experiment, each task had two phases; if participants finished the first phase, they received printed instructions for the second (more difficult) phase. See the figure captions for descriptions of the specific instructions provided to participants.

Retrospective Commentary (30 minutes). Immediately following the completion of the task, the system automatically processed the video to extract 20 second episodes centered around each undo and self-reported incident.

5.3.3 Usability problem extraction. The data analysis process for this study was almost identical to that of the SketchUp study. To facilitate controlled comparisons between experiments, the same researcher was responsible for processing the data and extracting usability problems.

Discarding Participants Whose Data Were Unusable. From an original set of 28 participants, we removed one participant because of a screen capture glitch. We also removed one participant because he did not finish his retrospective during the time allotted. We removed one participant because his partner did not show up, requiring us to pair him with the experimenter during the retrospective. Finally, we removed one more participant because his nonnative english commentary was often unintelligible to the experimenter. Removing the data from these 4 participants left 24 participants, whose data we used in the next phase of analysis.

Discarding Unclear Episodes. The 24 participants produced 255 episodes (130 undo episodes and 125 self-report episodes). Cumulatively, this equates to an average of 10.6 episodes per person, or 0.71 episodes per minute of Photoshop usage. From this initial set of 255 episodes, we discarded 42 episodes (16%) because the combination of commentary and screen capture video was not clear enough for the researcher to extract a complete usability problem instance. We discarded an additional 15 episodes

(6%) in which the user could not remember enough about the episode to answer any of the retrospective questions.

The number of unclear episodes was significantly higher for Photoshop than for SketchUp. One possible reason is that Photoshop is a much more complex application than SketchUp, with many hidden modes and settings that cannot be inferred from screen capture. Another possibility is that the Photoshop tasks involved subtle modifications to images, making it much more difficult to infer the user's progress in the task by examining screen capture (in contrast to SketchUp, in which the user's progress is evident from the state of the 3D model).

This process left 198 episodes (102 undo episodes and 96 self-report episodes).

Discarding False Alarms. From the remaining set, we identified eight episodes (4%) that contained no identifiable usability problems. Three of these false alarms were from self-report episodes (two accidental presses of the 'report issue' button, and one error induced by a window focus problem with the reporting button itself). The remaining five false alarms were due to backtracking events that failed to indicate usability problems. Thus, the overall false alarm rate for backtracking episodes was 5/102 = 4.9%.

Of the five backtracking false alarm episodes, two resulted from reversing purposeful explorations, in which participants were simply experimenting with Photoshop to learn its functionality. One false alarm occurred because the user continued to work after he was told to stop. (He rushed his work, making a slip.) One false alarm occurred when a user purposefully cleared his selection to get a better view of his image, then used undo to recover the selection and continue working on it. Finally, one false alarm was a duplicate of the window focus problem described in the preceding paragraph.

Identifying Usability Problem Instances. After all of the data preparation steps described previously, 190 episodes remained. A researcher analyzed these episodes to extract 222 usability problem instances. As with SketchUp, the mapping from episodes to usability problems instances was many-to-many. There were 14 problem instances that were incidental to the triggering event, and 18 overlapping pairs of undo/self-report episodes.

Merging Usability Problem Instances. Next, a researcher merged the 222 problem instances to form 106 unique usability problems. Again, we took a conservative approach to the merging process, matching instances only when their differences were superficial. The merge rate was 2.09:1, which is similar to what was found for SketchUp. The full list of usability problem descriptions can be found in Akers [2010, Appendix A].

Coding for Problem Severity. Three knowledgeable Photoshop users coded each of the usability problems for severity, and were compensated with gift checks worth approximately $30 / hour. For rating impact/persistence, we reused the scale from the SketchUp experiment (see Table I). Initially, we also planned to reuse the frequency scale from the SketchUp experiment. However, piloting this approach for Photoshop revealed a floor effect. Since Photoshop has so many features, only a small percentage of which would be used during any particular session, almost all of the problems found in the study would be rated a 1/5 or 2/5 on the frequency scale. To allow for better resolution on the low end of the frequency scale, we replaced the 1-5 ordinal scale with a 0-100 ratio scale. Raters were given the following instructions.

> Out of 100 Photoshop users, how many would experience this problem during a typical session? (A rough estimate is fine.) Keep in mind that a problem will not occur if the relevant application feature is not used.
>
> Your answer should be between 0 and 100. Answer '0' only if you think the problem would almost never happen.

To calculate a single severity score for each rater, we multiplied the two scales together, resulting in a severity score between 0 and 500 for each problem. Raters could see the effect of the scores on the relative severity rank for each problem. They adjusted their ratings until the relative ranks of all problems matched their intuition. (This was a slight change to the severity rating process for SketchUp, in which raters were unaware of how the frequency and impact/persistence scores would be combined into a single rating.)

The 106 problems were rated as part of a larger set of 179 problems, including an additional 73 problems found in the next study (see Section 6). We divided the 179 problems into three sets: a training set (7 problems), a test set (10 problems), and an independent set (162 problems). Coders used the training set to discuss the severity scales and resolve differences in coding styles. Next, they independently rated the 10 problems from the test set, and then discussed the differences and adjusted their ratings. Finally, they independently rated the remaining 162 problems.

As in the SketchUp experiment, we used Cronbach's Alpha [Cronbach 1951] to estimate the consistency of the three raters. Since the relative ordering of problems is more relevant than the absolute scores, we estimated inter-rater reliability on the ordinal ranks of each set of problems. Before the discussion, Cronbach's Alpha was 0.86; after coders adjusted their ratings, it increased to 0.98. Coders then independently rated the remaining 162 problems. For the final set of 106 problems in this study (using the adjusted ratings from the test set), Cronbach's Alpha was 0.82.

To reduce the effect of individual outliers, we chose the median of the three scores as the severity score for each problem. We then ranked all problems according to the median scores, producing an ordinal severity scale for all problems. (Problems with the same score were assigned to the same rank.) Inspecting the final ranked list of problems, we assigned categories to ranges of problems. Problems with median scores from 0 to 9 (ranks 1−10) were labeled as mild, those between 10 and 19 (ranks 11–16) were labeled as medium severity, and those with scores >= 20 (ranks 17−26) were labeled as severe.

5.3.4 Results.

Comparison between Backtracking and Self-Report. Figure 6 depicts the number of problems detected by each method or set of methods. On the left are the results for all problems, while on the right are the results for problems rated as severe. The data show that backtracking events detected 66 of the 106 problems, while self-reporting detected 76. Focusing on the most severe problems, backtracking and self-reporting each detected 14 problems.

Reasons Problems Were Not Reported. As in the SketchUp study, self-report failed to detect some of the problems found by backtracking analysis. Of the 49 responses to the retrospective question about reasons for not reporting, 19/49 (39%) indicated a failure to report because the participant blamed himself for the problem rather than Photoshop. Another 14/49 (29%) said that the problem was too minor to report. Another 10/49 (20%) said that they should have reported the problem, but simply forgot. Interestingly, 5/49 (11%) said that they did not report the problem because they thought the testing task was not realistic. (This may reflect a difference between the testing tasks,

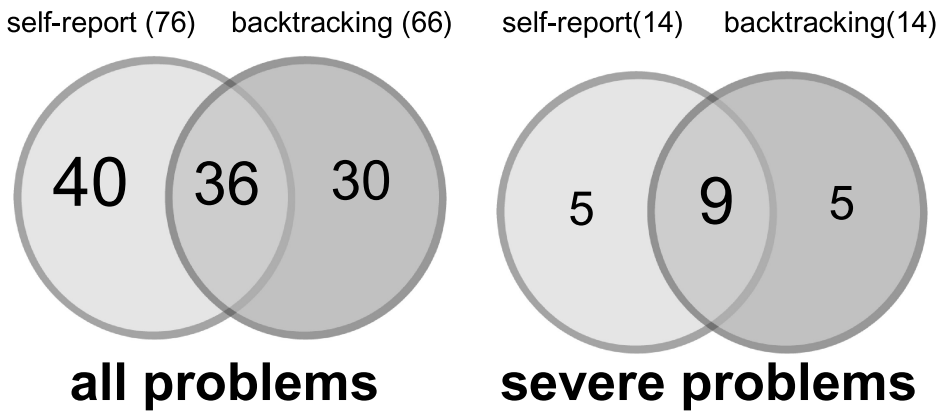

Fig. 6. Two Venn diagrams depicting the number of usability problems detected in Adobe Photoshop by each of the two methods or combination of methods. The left diagram shows the results for all problems, while the right diagram focuses on problems rated as severe. Problems in the middle of each Venn diagram were detected by both methods, while those on the outsides were detected by only one method. Note that backtracking events detected the same number of severe problems (14) as self-report.

Table II. A Comparison of the Number of Usability Problems Found in Each Experiment

	SketchUp			**Photoshop**		
	Self-report	Backtracking	Diff	Self-report	Backtracking	Diff
Severe	**22**	**23**	**+5%**	**14**	**14**	**0%**
Medium	28	31	+ 11%	10	10	0%
Mild	15	20	+ 33%	52	42	- 19%
Total	65	74	+ 14%	76	66	- 13%

and how these particular participants use Photoshop in their daily life.) And finally, one participant said that he did not report a problem because he could not imagine a way to fix the software to avoid the problem. These results echoed the concerns with attribution of blame identified in the SketchUp experiment. If anything, the failure to self-report was worse in this study, as it extended to severe problems: self-report detected only 14/19 (74%) of the severe problems, as opposed to 22/25 (88%) of the severe problems in the SketchUp study.

5.4 Comparing SketchUp and Photoshop

Table II compares the key results from the Photoshop and SketchUp experiments. It is evident from both studies that backtracking analysis and self-reporting detect comparable numbers of usability problems, and that this similarity is consistent across problem severity levels. It is also evident that backtracking analysis performed somewhat better with SketchUp than with Photoshop. There are two possible reasons for the difference in performance. First, consider that the numbers for backtracking analysis in this study include undo events only; since the tasks involved modification of existing content, there were no erase events. Second, consider that Photoshop is a much more complex, full-featured application than SketchUp. A much larger percentage of the problems in Photoshop involved feature discoverability. These problems often failed to induce backtracking events, unless the participant did not know what

feature to look for (and therefore searched for the feature by trial-and-error, requiring the repeated use of undo). Finally, it is apparent that SketchUp presented a much higher percentage of severe problems than Photoshop. This is an unsurprising result, given that Photoshop is a much older, more mature application.

The false alarm rate for backtracking analysis in this experiment was only 4.9%, which is considerably lower than in the SketchUp experiment (27%). However, recall that most false alarms in the SketchUp experiment could be attributed to two specific uses of the erase command. In contrast, the false alarms found in Photoshop were more varied, as described in Section 5.3.3.

5.5 Discussion

This section reflects on the results of the two studies, and discusses possible threats to internal and external validity.

5.5.1 Interpreting the Results.. We were initially surprised to find that backtracking events detected so many problems, compared to self-reporting. Many problems, for example, do not seem likely to produce a backtracking event.

Upon reflection, there are two possible reasons why the problem detection rates for backtracking events were so high. First, sometimes backtracking operations happened in circumstances we would not have expected. Consider the following sequence from the experiment: a user attempted to move a piece of geometry, and nothing happened (because, unbeknownst to the user, the geometry was anchored to its position). While it would seem that there was no reason to undo (since there was no actual change to the geometry), the user still executed an undo just to make sure that he hadn't changed anything.

Second, recall that we recorded problem instances even when the discovery of the problem was incidental to the method. Within the 20 second window of each screen capture episode, we often detected participants having difficulties unrelated to the backtracking event that triggered the episode. This accounted for nearly 20% of the problem instances for SketchUp, and 6% of the instances for Photoshop.

5.5.2 False Alarms. We expected to see two types of false alarms from backtracking events: purposeful uses of backtracking for design exploration, and purposeful uses of backtracking associated with learning the interface. Neither study produced false alarms related to the exploration of design alternatives. In hindsight, this is likely due to the fact that the task goals were specified so precisely. More surprisingly, we observed no learning-related false alarms in SketchUp, and only two such false alarms (1% of all episodes) in Photoshop. We speculate that we saw so few false alarms because the study protocols included both a 15-minute training and a 10-minute exploration period. By the time participants began the tasks, they were ready to work rather than explore. We had also emphasized the availability of online help materials during the study and provided users the option to review the training videos at any time during the tasks. Participants were observed repeatedly referring to these resources, instead of learning by exploring the interface.

Some false alarms we did find revealed categories of backtracking events that we did not anticipate. The creation of construction lines in SketchUp is an example of a "temporary action": an action that doesn't directly further a goal, but helps us to think or work more efficiently. Backtracking events associated with temporary actions are not indicators of usability problems, but rather indicate sophisticated uses of backtracking. The erasure of system-drawn lines is an example where the operation being reversed is a system action. The system-drawn lines are only of aesthetic importance; some participants decided to keep them, while others erased them.

5.5.3 Time on Task. Of the 90 minutes in each experiment, participants spent only 15 minutes working on the task. The task time is shorter than that of a typical usability study. (In contrast, a real-world usability study of SketchUp allocated 20 minutes of actual task time for a 60 minute study.) To some extent, the short task time reflects the goals of these studies, which were to compare two usability evaluation methods in an experiment rather than to use them in practice. Increasing the task time is a direction for future work (see Section 7.3). However, it is also useful to note that 20 of the 90 minutes were spent training participants in the use of the user-reported critical incident method. While it may be possible to reduce the amount of training, this is a fundamental difference between self-reporting and event-based methods. Event-based methods can be employed without any up-front training (and even without users' prior awareness that they are being monitored).

5.5.4 Internal Validity Concerns. The study designs minimized several possible threats to internal validity. Since we varied the method in a within-subjects manner, we counterbalanced the order of the methods in the retrospectives to avoid learning or fatigue effects. One other internal validity threat lies in the process of merging of usability problem instances. If we were inconsistent in how we merged problems, problems might end up at substantially different levels of granularity. (Consider the difference between, "Users have trouble selecting objects" and, "Users have trouble understanding how to select objects when using the scale tool.") The former would likely attract a higher severity rating, and would also be more likely to be detected by all three methods. Aware of this potential threat, we tried to write problem descriptions that were much more like the latter than the former, and took a conservative approach to the merging of problems. Unfortunately, there is no simple test for success; merge rates naturally vary with the frequency of a problem's occurrence, as well as its level of granularity. Future studies might seek to verify the validity of the merging process by employing multiple researchers and assessing the inter-researcher reliability.

5.5.5 Generalizing to Other Tasks. The conclusions of these experiments may depend to some extent on the chosen testing tasks. SketchUp (and especially Photoshop) are large and complex applications; it is not possible to comprehensively evaluate their usability by choosing a few tasks. That said, we tried to choose tasks that are representative of new user goals.

5.5.6 Generalizing to Other Applications. These experiments showed that backtracking analysis was effective for two creation-oriented applications. We were somewhat surprised by the low incidence of learning-by-exploration false alarms for SketchUp and Photoshop, but we cannot see anything specific about these applications (or our testing tasks) that would discourage learning by exploration. We suspect that backtracking analysis will generalize to work effectively with the broader class of creation-oriented applications such as word processors and page layout tools, but such generalizations are left as future work.

5.5.7 Generalizing to Other Evaluators. The evaluator effect [Jacobsen et al. 1998] for traditional usability testing is also a concern for backtracking analysis. Since a single evaluator was responsible for identifying usability problems, it is possible that some of the findings of this study would not generalize to other evaluators. Hopefully, this effect is smaller for backtracking analysis than for traditional testing (since in backtracking analysis the episodes are automatically selected). The follow-up study described in Section 6 employs multiple evaluators to mitigate this concern.

5.5.8 Summary. This section described two experiments that compared backtracking analysis with the user-reported critical incident technique. The experiments measured the effectiveness of backtracking analysis in two ways: the hit rate (percentage of problems detected), and the false alarm rate. In both studies, we found that the number of problems detected by each method was comparable, and verified that this result held for the problems rated as severe. Like most event-based approaches, backtracking analysis did produce false alarms, but at lower rates than we expected (27% in the SketchUp study, and 4.9% in the Photoshop study).

These results are particularly exciting because backtracking analysis may provide a viable method in circumstances where self-reporting is not an attractive option. Consider, for example, "in the wild" studies where users are often more interested in getting their work done than reporting problems with the interface [Nichols and McKay 2003]. However, it should be noted that adapting backtracking analysis for use in the wild presents its own challenges, as discussed in Section 7.3 (future work).

6. COMPARISON WITH TRADITIONAL USABILITY TESTING

This section describes an experiment designed to estimate the strengths and weaknesses of backtracking analysis as compared to traditional laboratory usability testing. Many variants of traditional laboratory testing exist, differing in their recommendations for running the tests and analyzing the results. Complicating matters, there is little data about which variants are used most often in practice. There is, however, some commonly cited popular literature describing commonly-agreed best practices for laboratory testing [Dumas and Redish 1999; Dumas and Loring 2008; Rubin and Chisnell 2008]. This study's implementation of traditional laboratory testing followed these best practices.

The specific goals of the experiment were to estimate the cost-effectiveness of each usability evaluation method, to compare the types of problems found by each method, and to solicit the opinions of professional usability evaluators on the potential use cases for each method.

6.1 Experimental Design and Recruitment

This experiment used Adobe Photoshop as the test application. Unlike in the experiments from Section 5, for this experiment we varied the usability testing methods in a between-subjects manner. This allowed us to reuse the raw data (episodes and retrospective commentary) from the previous Adobe Photoshop study described in Section 5.3. We recruited 24 new participants for the traditional laboratory study condition, taking care to match participants across conditions on their prior expertise with Photoshop. After the new study was complete, three professional usability evaluators identified usability problems from both conditions. Note that the evaluators performed a fresh analysis of the raw data from the backtracking analysis study; we did not reuse the usability problem descriptions found in the prior experiment by a different evaluator. Keeping the evaluators consistent across conditions avoided introducing a significant confounding variable since different evaluators may identify different usability problems [Jacobsen et al. 1998]. It also allowed the evaluators from this experiment to speculate on the differences between methods (since they had experienced both). The study resulted in two types of data: quantitative data (usability problem counts and severity ratings), and qualitative data (semi-structured interviews with the professional evaluators). These data were used to assess the strengths and weaknesses of backtracking analysis in three ways. Section 6.2 offers a preliminary investigation into the cost-effectiveness of backtracking analysis, suggesting that it may be substantially more cost effective than traditional laboratory testing when usability testing is

performed in groups of at least four participants. Section 6.3 compares the types of usability problems found by backtracking analysis and traditional testing, revealing that traditional testing may be better suited for identifying problems related to feature discoverability (and possibly strategy formation). Section 6.4 investigates the practical applicability of backtracking analysis, finding that usability evaluators were most excited about the use of paired-participant retrospectives.

The recruitment of testing participants for the backtracking analysis condition was previously described in Section 5.3.1. For the traditional laboratory study condition, we recruited an additional 24 participants from a variety of academic departments at Stanford University. Participants in both conditions responded to flyers posted in academic buildings at Stanford University.

We were careful to control for participants' prior expertise in Photoshop in forming the sample for traditional laboratory testing. To accomplish this, we recruited the participants for testing within levels of Photoshop expertise, so as to match the distribution of expertise in the backtracking sample. The 24 participants in each condition had the exact same distribution of responses to the question asking them to rate their prior experience retouching images in Photoshop. Of the 24 participants in each condition, 3/24 (13%) had never used Photoshop before for this purpose, 11/24 (46%) described themselves as novices, 6/24 (25%) described themselves as intermediate users, and 4/24 (17%) described themselves as experts. For a 60 minute session, each participant received a $15 gift check.

For the traditional condition, we recruited a professional usability test moderator to run each session, and three professional usability evaluators to identify and report usability problems. The moderator had two years of experience running usability tests as a user experience researcher with two organizations (a design consulting company and a software company). She also had formal training in conducting usability tests while a Masters Student. The three usability evaluators (whom we will refer to as Evaluator A, Evaluator B, and Evaluator C) had considerable industry experience in usability evaluation, and varying degrees of experience with the Adobe Photoshop application. Evaluator A was a user experience analyst with 18 months of experience conducting and analyzing think-aloud usability tests. He described himself as an expert user of Photoshop, with prior job experience developing Photoshop plug-ins for image retouching. Evaluator B was a freelance usability consultant, with general expertise in design, rapid prototyping, and usability testing. He had 10 years of experience moderating and evaluating think aloud usability tests, both in his role as a consultant, and as an employee at a financial institution and a security startup company. Before the test he was a beginning user of Photoshop, but had extensive experience with other software for image editing and technical illustration. Evaluator C was a freelance usability consultant with ten years of experience moderating and evaluating think-aloud usability tests. She had the least experience of the three with Photoshop; she had only used it occasionally to touch up her own personal photographs, and described herself as a high-novice or low-intermediate user.

The usability test moderator received $2,500 as compensation for conducting the entire test, while usability evaluators each received $1,500 for their work. Evaluator C volunteered to work without payment. All three evaluators received a copy of Photoshop CS3, graciously donated by Adobe.

6.2 Usability Testing Procedure

For the backtracking condition, we reused the raw participant data from the experiment described in Section 5.3. Please refer back to this previous section for a complete description of the experimental protocol. The remainder of this section describes the

test procedure for the traditional laboratory testing condition. The study was performed at Stanford University. The computer was configured with a copy of Adobe Photoshop CS3, and screen capture recording software. We also placed a single video camera behind the desk, aiming the camera to provide a clear view of the participant's face and body.

To facilitate comparison at the level of individual participants, we designed the experimental protocol for the traditional condition to approximately match the session length of a typical session of backtracking analysis. Recall that each backtracking session was 90 minutes in length, but approximately 30 minutes of that time was spent on the self-reporting portion of the study (20 minutes of training in self-reporting, and approximately 10 minutes of retrospective commentary specific to self-report episodes). Accordingly, we designed each laboratory testing session to last 60 minutes.

Each 60 minute laboratory test was divided into the following sections: greeting (2 minutes), training in Photoshop (15 minutes), instructions on thinking aloud (3 minutes), practice (10 minutes), tasks (25 minutes), and retrospective (5 minutes).

Greeting (2 minutes). Since the moderator plays a much more active role in traditional laboratory testing than in backtracking analysis, we devoted careful attention to the initial greeting process. A rough transcript of the greeting is provided in Akers [2010, Appendix B], but the moderator ad-libbed somewhat rather than reading verbatim from the script.

Training in Photoshop (15 minutes). The training video was identical to that used in the previous study. Please refer to Section 5.3 for details.

Practice (10 minutes). Participants were given 10 minutes to practice using Photoshop. As in the backtracking condition, participants were provided a "rubber duck" image, and were allowed to freely explore the interface during this time. The moderator purposefully did not ask participants to think aloud during the practice phase. Pilot studies had also revealed a concern that participants might explore the interface less thoroughly when their actions were being actively watched by an observer. Accordingly, the moderator moved to the other side of the room during the practice phase.

Training in Thinking Aloud (3 minutes). To prepare the participant to think aloud while working on the task, the moderator provided some training in how to effectively think aloud.. The moderator demonstrated the think aloud process while replacing the staples in a stapler. She then asked the participant to practice thinking aloud while refilling the tape in a tape dispenser.

Modeling Task (25 minutes). As in the backtracking analysis study, we randomly assigned participants to one of two tasks (described in Section 5.3): half completed the tulips task, while the other half completed the portrait task. Note that the time allotted for the task was 10 minutes longer than the time allotted in the backtracking condition; we did not make any attempt to control for task time across conditions. (It would be difficult to devise a fair control for time on task even if we had wanted; the think aloud process is likely to change the speed at which participants are able to complete the task.)

The moderator followed the advice of Dumas and Loring [2008] in deciding how to interact with each participant during the task. The complete instructions are included in Akers [2010, Appendix B].

Retrospective (5 minutes). Immediately following the completion of the task, the moderator interviewed the participant. This interview provided a chance for the moderator to ask follow-up questions, and to probe the participant's understanding of features used during the task. At the end of the interview, the moderator always asked the participant for suggestions to improve the software.

6.3 Usability Problem Extraction

6.3.1 Training the Usability Evaluators. Usability evaluation is a subjective process; different evaluators may identify different usability problems even when reviewing the same data [Jacobsen et al. 1998]. To help mitigate this effect, we provided some instructions to the evaluators. The complete set of instructions is included in Akers [2010, Appendix B], highlights of which are described in the following.

To instruct evaluators in identifying usability problem instances, we provided a list of criteria for identifying usability problems from Jacobsen et al. [1998]. We also provided Table 1 from Skov and Stage [2005], which classifies usability problems along two dimensions: how the problem is detected, and how the problem impacts the user. Hornbæk and Frøkjær [2008] used these same two resources to train usability evaluators to identify usability problems in a recent study of problem matching techniques.

Identifying usability problems sometimes involved extra work in the backtracking condition, because of a complication in reusing the data from the previous study. The complication arose when a backtracking episode overlapped with a self-report episode, and the commentary for the self-report episode was collected first. We resolved this in the same manner that we handled overlapping episodes in the previous experiment; we included the commentary from the overlapping self-reporting episode as additional evidence. We asked evaluators to ensure that the evidence of a problem was visible in the backtracking episode before reporting a problem described in the overlapping self-report episode.

We also provided instructions on how to report usability problem instances. To report a problem, evaluators filled out a form answering the following four questions.

(1) What actually happened in this episode, and what did the user say about it? (2–3 sentences)
(2) How did the user work around the problem, if at all? (1–2 sentences)
(3) What was the broader context in which the problem occurred? What was the user trying to accomplish? (1–2 sentences)
(4) Provide a one-sentence headline for the problem.

We provided three "golden rules" for reporting usability problem instances.

(1) Focus on describing symptoms rather than inferring causes.
(2) Avoid trying to read users' minds when describing their intentions or thoughts; rely on evidence.
(3) Clearly distinguish between the user's actions and explanations.

To ensure that evaluators had a chance to practice identifying and reporting usability problem instances, we asked them first to evaluate a separate "training set" of four participants (two participants from each condition, balanced according to the testing tasks). The data obtained during this training phase was not included in the final results. After each evaluator had finished reporting problems for the training set, we provided feedback on problem identification and reporting. Evaluators B and C seemed to be classifying many learning-related difficulties as false alarms; we reminded them that learnability should be considered an important goal. We also needed to remind

Evaluator C that it was not necessary to report a problem for every backtracking episode; some might represent false alarms.

6.3.2 Collecting Usability Problem Reports. After the training was complete, we partitioned the original set of 48 participants into 3 sets of 16 participants, one set for each evaluator. We randomly assigned participants to evaluators, taking care to balance for condition, task, and prior Photoshop expertise of participants. To mitigate learning effects during evaluation, we required the evaluators to alternate between conditions as they worked (participant 1 from backtracking, participant 2 from traditional, participant 3 from backtracking, participant 4 from traditional, etc.)

This process resulted in 219 problem reports, including 72 backtracking reports and 147 traditional reports. Most of these reports originated from Evaluators A (94/219, 43%) and B (97/219, 44%); Evaluator C, who had considerably less experience with Photoshop, submitted only 13% of the reports (28/219).

Generating Usability Problem Instances. A researcher inspected all 219 problem reports to generate usability problem instances. In most cases, the mapping was one-to-one; we simply copied the report description to form an instance of a usability problem. There were a few exceptions, described as follows.

In some cases, there was no clear description of a difficulty apparent in the report. (The evaluator was not clear what had happened, and could not pinpoint the difficulty.) We discarded these 13 reports (5 backtracking reports, and 8 traditional reports). We discarded one additional traditional report because the evaluator had misinterpreted the task instructions, blaming a user for failing to accomplish a subtask that was not required. Discarding these 14 reports yielded 205 problem reports (67 backtracking reports and 138 traditional reports).

In some cases, a single problem report contained more than one usability problem. We split such reports into individual usability problem instances. This process resulted in an additional 13 usability problem instances (9 backtracking instances and 4 traditional instances). After the splitting process, we had compiled a final list of 218 usability problem instances.

Merging Usability Problem Instances. A single researcher merged the 218 problem instances to form 134 unique usability problems. As in the studies from Section 5, we took a conservative approach to the merging process, matching instances only when their differences were superficial. The merge rate was 1.63:1, which is somewhat lower than what we found in previous studies. The full list of usability problems can be found in Akers [2010, Appendix A].

Coding for Problem Severity. Three knowledgeable Photoshop users coded each of the usability problems for severity and were compensated with gift checks worth approximately $30/hour. Using a similar process to that described in the Google SketchUp study, we used the median severity rating from the three coders to rank the severity of each problem (using a 24-point scale instead of an 8-point scale). We then classified each problem as mild, medium, or severe. Of the 134 problems, 91 were classified as mild, 20 as medium, and 23 as severe. Cronbach's alpha [Cronbach 1951], a measure of inter-rater consistency, was 0.83, which is nearly identical to the value observed in the SketchUp study ($\alpha = 0.82$). A breakdown of how many problems of each type were discovered by each method (including an analysis of overlap) is contained in Akers [2010, p. 106].

Interviews of Usability Evaluators. To supplement the quantitative data, we interviewed each of the evaluators individually to capture the qualitative aspects of their

experience with the two methods. The interviews followed a semi-structured format, and lasted for approximately one hour each. We structured the interviews around the following questions.

— What do you see as the overall strengths and weaknesses of backtracking analysis?
— What types of problems do you think backtracking analysis is best for finding? What types of problems does it tend to miss?
— How do you think backtracking analysis could be improved?
— As an evaluator, how hard do you think that backtracking analysis is to learn to do well?
— What experience do you think an evaluator needs for backtracking analysis, and how does this compare to the experience required for traditional lab testing?
— Would you recommend backtracking analysis as a technique to a colleague? For what kind of a product/problem/situation?
— What other techniques would you combine backtracking analysis with (in a single session, or in a set of studies on a single product)? In what ways do you see those techniques being complementary?
— Suppose someone asked you to do a cost benefit analysis of backtracking analysis compared with traditional usability testing, to help them decide which to use. How would you describe the cost/benefit tradeoffs for backtracking analysis vis-à-vis traditional lab testing?

6.4 Results

This section presents a preliminary analysis of cost effectiveness (Section 6.4.1), a characterization of the problems found and missed by backtracking analysis (Section 6.4.2), and a discussion of the usability evaluation contexts in which backtracking analysis might be most useful (Section 6.4.3).

6.4.1 Cost Effectiveness of Backtracking Analysis.

Measuring Costs and Benefits. To analyze cost-effectiveness, we first chose metrics for benefits and costs. As in the rest of this article, we measured the benefits of a usability evaluation method by computing the total number of unique usability problems discovered. We measured the per-participant cost of a usability evaluation method in terms of test moderation (expert hours spent overseeing the running of the test), and test evaluation (expert hours spent analyzing and reporting the results). We purposefully did not include the costs of recruiting, recognizing that these costs will be consistent across methods and will vary greatly depending on the setting and location.

When measuring moderation costs, there is a key difference between traditional laboratory testing and backtracking analysis. In traditional laboratory usability testing, per-participant test moderation costs are fixed, but in backtracking analysis they depend on a single parameter (k): the number of participants that can be tested simultaneously with a single human moderator. Doubling k effectively halves the cost of moderating the test. In this particular experiment, limitations of laboratory space and computer hardware forced us to run only two participants at a time ($k = 2$). However, we have found it possible to test with as many as 8 participants simultaneously with no significant strain placed on the moderator. The following cost-benefit analysis projects moderation costs for $k = 4$ and $k = 8$ and reports the observed moderation costs for $k = 2$.

To account for evaluation costs, each of the three evaluators for this study recorded the amount of time that they spent reviewing the videos and writing problem reports. In the backtracking condition, evaluators recorded the time that they spent reviewing each individual episode (regardless of whether the evaluator reported any problems

Table III.

An aggregate cost-benefit analysis comparing traditional laboratory testing and backtracking analysis. Moderation costs for backtracking analysis are projected for different group sizes (k). The bottom row shows the aggregate time required to discover each unique usability problem; when k = 8, backtracking analysis is approximately twice as efficient as traditional lab testing. Note that evaluation time was much shorter for backtracking analysis (in part because there was less video to watch, and in part because evaluators reported fewer problems in this condition).

	Traditional lab	**Backtracking analysis**		
		$k = 8$	$k = 4$	$k = 2$
moderation time (hh:mm)	36:00	4:30	9:00	18:00
evaluation time (hh:mm)	28:40	12:54	12:54	12:54
total time (hh:mm)	64:40	17:24	21:54	30:54
usability problems found	96	53	53	53
time per problem (hh:mm)	**0:40**	**0:20**	**0:25**	**0:35**

for that episode). In the traditional condition, evaluators reported the total time they spent reviewing the entire video for a participant.

Aggregate Cost-Benefit Analysis. An initial cost-benefit analysis, which aggregates results across all evaluators, is shown in Table III At k = 2 (two participants per session, as in this study), the cost-effectiveness results were comparable across conditions (35 minutes per problem in backtracking analysis, compared to 40 minutes per problem with traditional testing). As the number of participants per session increases, backtracking analysis would perform substantially better. At k = 8, backtracking analysis would be approximately twice as efficient as traditional laboratory testing. Since we have not yet tested with group sizes bigger than 8, it would be inappropriate to extrapolate beyond k = 8. (It is certainly possible that a large enough group would require multiple moderators to coordinate, effectively preventing the cost from scaling.)

Visualizing Cost vs. Benefit. Figure 7 provides a more detailed view of the data, plotting cost (hours of moderation and evaluation time) vs. benefit (number of unique usability problems found). Each curve shows a particular usability evaluation method; there is one curve for traditional usability testing, and there are three curves for backtracking analysis (k = 2, k = 4, k = 8). The shape of each curve was estimated by randomly sampling subsets of the original set of 24 participants in each condition, and computing the costs and average benefits for each subset size (see Akers [2010, Appendix D]). One can interpret these curves as meaning, "For a given amount of time one is willing to spend on evaluation method X, how much benefit is expected?" Note that each curve terminates at a different point along the cost axis; this is because the analysis is limited to the 24 participants for each condition (and the cost of running all 24 participants depends on the evaluation method and its parameters).

6.4.2 Types of Problems Found and Missed by Backtracking Analysis.

Classifying Problems by Severity. Did the median severity of usability problems differ between backtracking analysis and traditional laboratory testing? If backtracking analysis was only better for detecting mild problems, then the cost-benefit results of the preceding section would carry little meaning. To investigate, we computed the median severity rank of problems discovered with each method. The median severity for problems found with traditional laboratory testing (6) was slightly higher than that of backtracking analysis (5). The result, however, was not statistically significant by a Mann-Whitney U test (z = 0.901, p = 0.34).

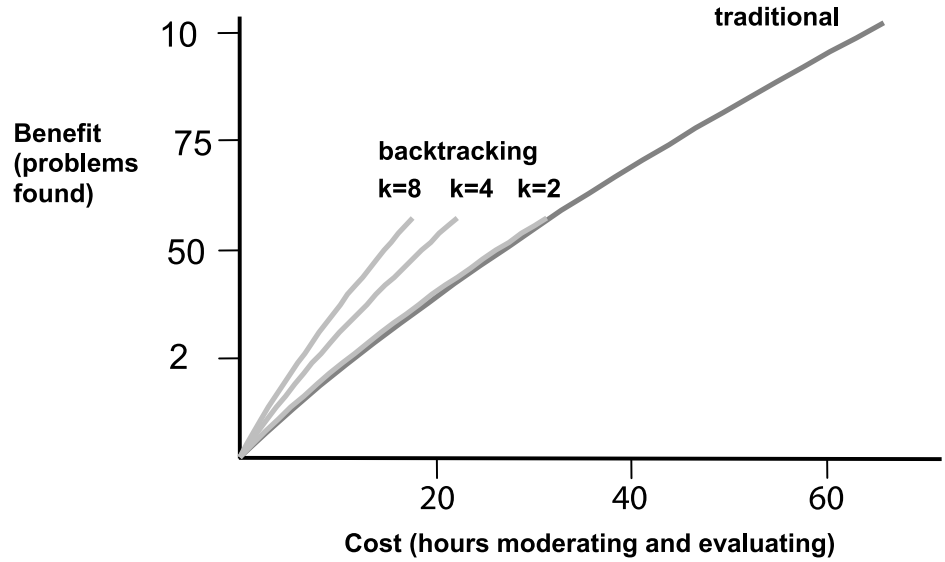

Fig. 7. A detailed cost-benefit analysis comparing backtracking analysis with traditional usability testing. This chart plots costs (expert hours moderating and evaluating) vs. benefits (number of unique usability problems found). Costs for backtracking analysis are projected for three different test group sizes (k). To estimate the shape of each curve, we randomly sampled subsets of the original set of 24 participants in each condition, and computed the costs and average benefits for each subset size. For clarity of illustration, we have divided the costs of running each backtracking analysis session evenly amongst the participants in the session; a plot of the raw data would include discontinuities at multiples of the group size. Note that each curve terminates at a different point along the cost axis, since the costs of running all 24 participants depends on the evaluation method and its parameters. The termination points correspond to the values listed in Table III (aggregate analysis).

Employing Other Problem Classification Schemes. Classifying usability problems only by their predicted severity leaves something to be desired; a usability evaluator considering the use of backtracking analysis might want to know more about the nature of the problems found and missed. To address this issue, we invented our own classification scheme, modeled loosely after the top level of the User Action Framework hierarchy [Andre et al. 2001].

— Forming strategies: difficulty developing a high-level strategy for accomplishing a goal.
— Finding features: difficulty locating an application feature (even if the desired feature is not known).
— Choosing parameters: difficulty choosing the right parameters for an action (e.g., tolerances).
— Executing actions: difficulty executing an action, resulting in surprise or frustration.
— Perceiving state: difficulty interpreting and/or remembering application state.

We set aside 28 problems that did not clearly fit into one of these categories. We classified the remaining 106 problems (61 uniquely found by think aloud, 32 uniquely found by backtracking, and 13 found by both methods); the results are shown in Figure 8.

Two conclusions are apparent from the figure. The first is that difficulties forming strategies were absent with backtracking analysis. (It should be noted that only three problems related to strategy formation were found by the think-aloud method,

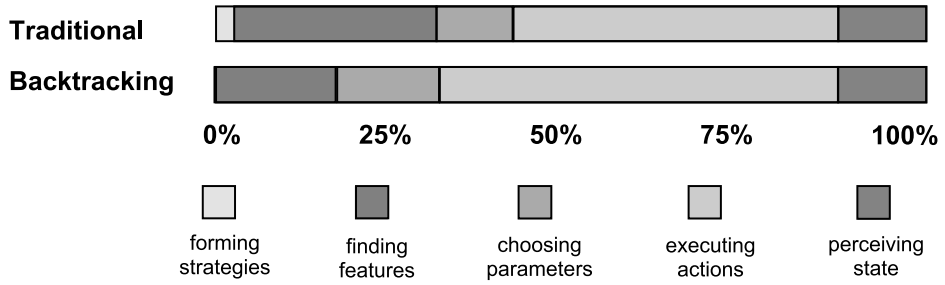

Fig. 8. An informal comparison of the types of usability problems found by traditional laboratory testing and backtracking analysis. Traditional laboratory testing detected a higher percentage of problems related to forming strategies and finding features.

so the differences are certainly not statistically significant.) However, theory also suggests that backtracking analysis would have difficulty detecting such problems, since they generally manifest as ”menu-cruising” behavior or long pauses of inactivity - not as backtracking operations. Also apparent from the figure is that difficulties related to finding features are much more commonly detected by think-aloud testing than by backtracking analysis. Backtracking analysis only detected feature discoverability problems when the user did not know what s/he was looking for and experimented by trial-and-error.

We also asked each of the three usability evaluators to speculate on the types of problems found and missed by backtracking analysis. Evaluator A volunteered that backtracking analysis might systematically fail to detect feature discoverability issues, confirming what was evident in the data. He admitted that it did find some of these issues, however.

Evaluator B speculated that backtracking analysis might be particularly useful for identifying problems during fast-paced interaction episodes where participants' actions are triggered by muscle memory rather than conscious thought. He also indicated that backtracking analysis may be particularly ill-suited for finding “big picture” problems, because the short, automatically selected episodes don't always give a complete picture of the participants' experience. He suggested that it might be possible to partially address this context issue by providing a short general purpose question period at the beginning of the retrospective section. One might ask participants about the hardest parts of the task, for example.

Evaluator C found it impossible to speculate on this topic, indicating that she did not have enough examples to generalize.

6.4.3 How Backtracking Analysis Fits into Practice. During the interviews, we also probed evaluators on how they thought backtracking analysis might fit into usability evaluation practice. Their responses are summarized here.

Suitability for Different Application Types and Usability Evaluation Goals. Evaluator B indicated that he thought backtracking analysis would work better for released applications than for early prototypes. He suggested that it would not be good for answering basic questions like, “Do the users even understand our interaction model?” This relates to his earlier comment that backtracking analysis might tend to miss “big picture” problems. In contrast, Evaluator A said that backtracking analysis was unsuitable for summative evaluations of complete systems, because it might systematically fail to detect certain types of problems. (As an example, he noted problems with feature discoverability.) He recommended it only for informal, formative evaluations

of software. Evaluator C focused on the expertise of the participants, reasoning that backtracking analysis tended to work better with expert participants than for novices. She said that experts tended to "talk about their difficulties more effectively". These comments from Evaluator C might have been influenced by her own lack of experience with Photoshop.

Expertise Required to be an Evaluator. There was no consensus amongst the evaluators concerning the usability or domain expertise required to be an evaluator. Evaluators A and C thought that the expertise required to evaluate would be roughly the same as for traditional usability testing. Evaluator B said that he believes that prior usability evaluation experience is less important in backtracking analysis. He claimed that backtracking analysis is less subjective and more systematic, and indicated that even an engineer with no usability experience might be able to function as an effective evaluator. He did provide a caveat: a less-trained evaluator would need to have an open mind about usability problems; backtracking analysis would not help anyone who was "determined to be skeptical" of the existence of usability problems in the interface.

Combining Backtracking Analysis with Other Methods. Both Evaluators A and B suggested combining backtracking analysis with real-world usage log data. Among other things, this would indicate which system commands are most often reversed, giving a real-world context for the usability problems found by backtracking analysis.

For Evaluator B, the chief concern when choosing between usability evaluation methods was not cost-effectiveness, but thoroughness. Backtracking analysis might provide a way to catch problems that would otherwise be missed during traditional testing. He suggested building support for retrospective analysis of undo episodes into usability testing software tools like Morae. He also suggested that a positive attribute of backtracking analysis is that it does not interfere with a participant's natural interaction with the software. Since backtracking events are logged without the participant's knowledge, quantitative measures such as task time or error counts are not tainted by the experimental manipulations.

Evaluator C suggested combining backtracking analysis with eye tracking data; she thought that this additional context would help her to interpret the episodes, and might also help the participant to remember what was happening in the episode during the retrospective.

Paired Participant Retrospectives. While we did not ask for feedback on the use of paired-participant retrospectives, all three evaluators expressed excitement about this technique. Evaluator B was particularly keen on the idea. Pairing up the participants, he said, makes participants more comfortable in admitting their mistakes. Since each participant has attempted the same task, they can empathize with each other and understand the context of each others' difficulties.

Evaluator B did raise a concern about the retrospective nature of the commentary in backtracking analysis. There is, he explained, a "theatrical" aspect to convincing developers that usability problems should be addressed, and it often helps to extract video clips showing evidence of users having emotional responses to a problem. He noticed that participants were often laughing about their problems when they reviewed them in the retrospective for backtracking analysis, whereas participants were more often annoyed or frustrated in the concurrent think-aloud condition. He speculated that participants become detached from their emotional involvement over time, and that any usability evaluation method purely based on retrospective analysis would fail to capture their original, raw emotion.

6.5 Discussion

Interpreting the Cost-Effectiveness Comparison. Since backtracking analysis turned out to be so much less expensive than traditional testing, we could only compare the methods for costs on the low end of the spectrum (see Figure 7). Nevertheless, these comparisons are meaningful. The 24 participants in the backtracking condition cost the same as it would to test 11.5 participants in the traditional condition (when k = 2), 8.1 traditional participants when k= 4, and 6.5 traditional participants when k = 8. These sample sizes (611) are within the typical range of scales for traditional usability tests, indicating that the comparisons are still relevant to practitioners.

As mentioned in Section 3, it is worth exploring other metrics for effectiveness besides problem counts and severities. The analysis in this study does not account for possible differences in the downstream utility of backtracking analysis and traditional laboratory testing. As described in Section 6.4, one of the professional usability evaluators speculated that backtracking analysis might be less persuasive because of its extensive reliance on a retrospective protocol, which may artificially distance participants from the negative emotions they felt while experiencing usability problems. Further studies would help to determine the extent to which this is indeed a limitation of backtracking analysis.

Construct Validity. In choosing a single way to operationalize traditional laboratory testing, this study has a mono-operation bias. One cannot know how the study results might have been different if we had provided different instructions to the usability test moderator, or the three evaluators. Even the choice to recruit different individuals to perform the testing and the evaluation can be questioned; it is certainly possible that there is an efficiency gain when the moderator and the evaluator are the same person. (The moderator-evaluator could then make use of her notes during the evaluation phase.) However, it is also possible that moderator-evaluators would make more mistakes during evaluation, in the process of reconstructing what happened from partial memories and hastily-scribbled notes.

Internal Validity. As previously mentioned, reusing the backtracking analysis data from a previous study made it impossible to randomly assign participants to conditions. Without random assignment, it cannot be certain that the participants in each condition were similar along all important dimensions. We attempted to compensate for this shortcoming by applying tight statistical controls on what we considered to be the most significant individual difference factor: the prior Photoshop expertise of participants. The 24 participants in each condition had the exact same distribution of prior experience, as measured by self-report.

External Validity. This study is limited in scope. Researchers should be cautious when generalizing from the success observed with backtracking analysis from usability testing experiments in a laboratory setting with three evaluators, two tasks, and a single application.

However, a broader perspective emerges when one considers the successes of this study in combination with those of previous studies described in Section 5. Together, these studies show that backtracking analysis has proved effective for evaluating two different applications, with two different types of tasks (creating content vs. modifying content), with a total of four evaluators with differing backgrounds. This is reason for optimism among practitioners who consider experimenting with backtracking analysis.

One additional concern about generalizability specific to this study is the reliance on a single usability test moderator to run all of the studies in the traditional laboratory

testing condition. It is possible that the test results would have differed if we had chosen a moderator with a different style of interacting with participants. (We are not aware of any formal studies of the "moderator effect," but it seems likely that meaningful differences could exist among moderators.) It is at least encouraging that all three evaluators gave positive feedback on the job performed by the moderator in this study.

6.6 Summary

The results of this third experiment touch upon some the strengths and weaknesses of backtracking analysis compared to traditional laboratory usability testing. Data from the experiment indicate that backtracking analysis can be significantly more cost effective than traditional laboratory usability testing, when one takes into account the ability to test participants in large groups. However, an analysis of the problems found by each method suggests that traditional laboratory testing is better suited for detecting problems related to feature discoverability, and perhaps strategy formation as well. Interviews with the three evaluators provided information about the practical applicability of backtracking analysis; evaluators were most excited about the use of paired-participant retrospectives, even outside the context of backtracking analysis.

7. CONCLUSIONS AND FUTURE WORK

The results from these experiments have convinced us that backtracking analysis provides a promising approach to detect problems in creation-oriented applications such as Google SketchUp and Adobe Photoshop. We have been particularly encouraged by the problem detection rates, especially for severe problems, and by the positive comments of the professional usability evaluators who experimented with backtracking analysis.

7.1 Summary of Experimental Findings

First, a series of experiments showed that it is possible to automatically characterize usability problems from backtracking events using a paired-participant retrospective technique (Section 4). We experimented with three other usability testing protocols (screen capture alone, screen capture plus concurrent think aloud, and screen capture plus retrospective think aloud), but none of these alternative methods provided enough contextual information to identify usability problems.

A set of two experiments with the Google SketchUp and Adobe Photoshop applications (Section 5) demonstrated that backtracking analysis is comparable in effectiveness to the user-reported critical incident technique [Hartson and Castillo 1998]. In the Google SketchUp experiment, backtracking events detected 5% more severe problems than self-reporting, and the false alarm rate for backtracking episodes was 27% (compared to just 1% for self-reporting). While this false alarm rate might seem high, it is noteworthy that all of the backtracking false alarms in this first study were related to the erase operation; none were associated with undo. In the Adobe Photoshop experiment, backtracking analysis found exactly the same number of severe problems as self reporting, and the false alarm rate for backtracking episodes was comparable (4.9% for backtracking vs. 3.1% for self-reporting).

Finally, an experiment with the Adobe Photoshop application compared backtracking analysis with traditional laboratory usability testing (Section 6). An initial cost-effectiveness study suggested that backtracking analysis using groups of eight participants is approximately twice as cost-effective as traditional laboratory usability testing. Both backtracking analysis and traditional testing proved capable of

identifying usability problems related to choosing parameters, executing actions, and perceiving user interface state. However, backtracking analysis failed to detect any problems relating to strategy formation, and performed poorly at detecting problems related to feature discoverability. This limitation may be remedied by combining backtracking analysis with other usability evaluation methods, as described here.

7.2 Implications for Usability Evaluation Practice

When designing a usability testing strategy, the size of the participant pool is not the only consideration (*e.g.*, consider the goal of frequent iteration of testing and software modification [Medlock et al. 2002], or the importance of task coverage [Lindgaard and Chattratichart 2007]). But for situations when study scale *is* a limitation, we hope that our approach is of interest.

In some cases, backtracking analysis might be used as a cost-effective replacement for traditional laboratory usability testing. The results from our initial cost-effectiveness study were promising, with the major caveat that backtracking analysis may be limited in the types of problems that it can identify. If the study is not concerned with identifying problems related to feature discoverability and/or strategy formation, then backtracking analysis could be a good choice.

Backtracking analysis may also be useful in combination with other techniques. To identify a wider variety of usability problems, one might combine backtracking analysis with a separate small scale traditional laboratory study targeted at identifying problems with feature discoverability and/or strategy formation. Employing a hybrid study of this form would be lower cost than a similarly-sized traditional study, and would likely prove more effective. One might also consider combining backtracking analysis with other methods like the user-reported critical incident technique.

7.3 Directions for Future Work

The results of this study have also suggested a number of open research questions.

— *Is it possible to adapt backtracking analysis for use with less constrained tasks?* How well would backtracking analysis work when task goals are less clearly specified? Backtracking commands can also be used to explore design alternatives; if a design change is undesirable, one can backtrack to reverse the change and implement a different solution. Will this substantially decrease the cost effectiveness of the technique, since unconstrained goals might induce more false alarms?
— *Is it possible to adapt backtracking analysis for use outside of the laboratory?* How would we adapt the paired-participant retrospective technique to work in a remote setting? Is it possible to extend backtracking analysis to study users "in the wild," who are not even aware that they are being studied? If so, how can we address the privacy concerns that would emerge? And, how would we capture sufficient context to interpret the event data?
— *Is it possible to increase the proportion of time on task without compromising the quality of the data?* In the current version of backtracking analysis, participants spend only about 25% of the time attempting usability testing tasks. A longer time on task might allow the evaluator to include several different testing tasks, increasing the comprehensiveness of the evaluation. Increasing task time would require decreasing the length of the retrospective sessions. Is it possible to capture sufficient context without pairing participants?
— *How well does backtracking analysis generalize to other application domains?* Would backtracking analysis be useful for evaluating tasks such as word-processing or web search?

— *What other automatic indicators of usability problems might be useful?* Possibilities include detection of repeated patterns of events [Siochi and Ehrich 1991], application-specific events that indicate difficulty (e.g., accidental suicides in a game [Thompson 2007]), or even behavioral or physiological indicators [Andreassi 2006; Tullis and Albert 2008]. Such automatic indicators might enable identification of a wider variety of problems, including those that backtracking analysis fails to detect.

In the long term, we also plan to explore other ways to reduce the costs of usability evaluation. What if one could automatically detect false alarms, and automatically group similar problem instances? The first is a classification problem, and the second is a clustering problem; it may be possible to pose each of these as a machine learning problem (perhaps using the data from studies like ours to train the models). If we could succeed in solving these problems, we could begin to define processes for automated usability evaluation in which the cost becomes a function of the number of problems in the interface, instead of a function of the number of participants. This is a vision worth pursuing.

REFERENCES

AKERS, D. L. 2010. Backtracking events as indicators of software usability problems. PhD Dissertation, Stanford University.

AKERS, D., SIMPSON, M., JEFFRIES, R., AND WINOGRAD, T. 2009. Undo and erase events as indicators of usability problems. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press, 659–668.

ANDRE, T. S., HARTSON, H. R., BELZ, S. M., AND MCCREARY, F. A. 2001. The user action framework: A reliable foundation for usability engineering support tools. *Int. J. Hum.-Comput. Stud. 54*, 1, 107–136.

ANDREASSI, J. L. 2006. *Psychophysiology: Human Behavior and Physiological Response*. Lawrence Erlbaum Associates.

BIAS, R. 1991. Interface-Walkthroughs: Efficient collaborative testing. *IEEE Softw. 8*, 5, 94–95.

BRUUN, A., GULL, P., HOFMEISTER, L., AND STAGE, J. 2009. Let your users do the testing: A comparison of three remote asynchronous usability testing methods. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press, 1619–1628.

CAPRA, M. 2002. Contemporaneous versus retrospective user-reported critical incidents in usability evaluation. In *Proceedings of the Human Factors and Ergonomics Society*. 1973–1977.

CARD, S. K., MORAN, T. P., AND NEWELL, A. 1983. *The Psychology of Human-Computer Interaction*. Erlbaum.

CRONBACH, L. J. 1951. Coefficient alpha and the internal structure of tests. *Psychometrika 16*, 3, 297–334.

DEL GALDO, E. M., WILLIGES, B. H., AND WIXON, D. R. 1986. An evaluation of critical incidents for software documentation design. In *Proceedings of the Human Factors Society*. 19–23.

DUMAS, J. AND REDISH, J. 1999. *A Practical Guide to Usability Testing*. Intellect Books.

DUMAS, J. S. AND LORING, B. A. 2008. *Moderating Usability Tests: Principles and Practice for Interacting*. Morgan Kaufmann.

FAULKNER, A. AND WALTHERS VON ALTEN, J. 2007. *Classroom in a Book: Adobe Photoshop CS3*. Adobe Press.

FLANAGAN, J. C. 1954. The critical incident technique. *Psych. Rev. 54*, 4, 327–358.

GRAY, W. D. 1997. Who ya gonna call! You're on your own [software usability design]. *IEEE Softw. 14*, 4, 26–28.

GUAN, Z., LEE, S., CUDDIHY, E., AND RAMEY, J. 2006. The validity of the stimulated retrospective think-aloud method as measured by eye tracking. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1253–1262.

HACKMAN, G. S. AND BIERS, D. W. 1992. Team usability testing: Are two heads better than one? In *Proceedings of the 36th Annual Meeting of the Human Factors Society*. 1205–1209.

HARTSON, H. R. AND CASTILLO, J. C. 1998. Remote evaluation for post-deployment usability improvement. In *Proceedings of the International Conference on Advanced Visual Interfaces*.

HARTSON, H. R. CASTILLO, J. C., KELSO, J., AND NEALE, W. C. 1996. Remote evaluation: The network as an extension of the usability laboratory. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press, 228–235.

HARTSON, R., ANDRE, T. S., AND WILLIGES, R. C. 2000. Criteria for evaluating usability evaluation methods. *Int. J. Hom. Comp. Interact. 13*, 4, 373–410.

HILBERT, D. M. AND REDMILES, D. F. 2000. Extracting usability information from user interface events. *ACM Comput. Surv*. 32, 4, 384–421.

HORNBÆK, K. AND FRØKJÆR, E. 2008. Comparison of techniques for matching of usability problem descriptions. *Interact. Comput. 20*, 6, 505–514.

HOWARTH, J., ANDRE, T. S., AND HARTSON, R. 2007. A structured process for transforming usability data into usability information. *J. Usability Stud. 3*, 1, 7–23.

IVORY, M. Y. AND HEARST, M. A. 2001. The state of the art in automating usability evaluation of user interfaces. *ACM Comput. Surv. 33*, 4, 470–516.

JACOBSEN, N. E., HERTZUM, M., AND JOHN, B. E. 1998. The evaluator effect in usability tests. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM Press, 255–256.

JOHN, B. E. AND MARKS, S. J. 1997. Tracking the effectiveness of usability evaluation methods. *Behav. Inf. Techn. 16*, 4, 188–202.

JOHN, B. E., PREVAS, K., SALVUCCI, D. D., AND KOEDINGER, K. 2004. Predictive human performance modeling made easy. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 455–462.

KARAT, C.-M., CAMPBELL, R., AND FIEGEL, T. 1992. Comparison of empirical testing and walkthrough methods in user interface evaluation. In *Proceedings of the Conference on Human Factors in Computing Systems*. ACM, 397–404.

KOENEMANN-BELLIVEAU, J., CARROLL, J. M., ROSSON, M. B., AND SINGLEY, M. K. 1994. Comparative usability evaluation: Critical incidents and critical threads. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 245–251.

KOHAVI, R., HENNE, R. M., AND SOMMERFIELD, D. 2007. Practical guide to controlled experiments on the web: Listen to your customers not to the hippo. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 959–967.

LAW, E. L.-C. AND HVANNBERG, E. T. 2004. Analysis of combinatorial user effect in international usability tests. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 9–16.

LEWIS, J. R. 1994. Sample sizes for usability studies: Additional considerations. *Human Factors 36,* 2, 368–378.

LINDGAARD, G. AND CHATTRATICHART, J. 2007. Usability testing: what have we overlooked? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1415–1424.

MEDLOCK, M. C., WIXON, D., TERRANO, M., ROMERO, R., AND FULTON, B. 2002. Using the RITE method to improve products: A definition and a case study. In *Proceedings of the Usability Professionals Association Conference*.

NICHOLS, D. M., AND MCKAY, D. 2003. Participatory Usability: supporting proactive users. In *Proceedings of the 4th Annual Conference of the ACM Special Interest Group on Computer Human Interaction: New Zealand Chapter (CHINZ'03)*. ACM SIGCHI, 63–68.

NIELSEN, J. AND LANDAUER, T. K. 1993. A mathematical model of the finding of usability problems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, 206–213.

NIELSEN, J. AND MOLICH, R. 1990. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 249–256.

NISBETT, R. E. AND WILSON, T. D. 1977. Telling more than we can know: Verbal reports on mental processes. *Psych. Rev. 84*, 231–259.

NORMAN, D. A. 1986. Cognitive engineering. In *User Centered System Design*, D. A. Norman and S. W. Draper Eds., Lawrence Erlbaum Associates, 31–61.

O'MALLEY, C. E., DRAPER, S. W., AND RILEY, M. S. 1984. Constructive interaction: A method for studying human-computer-human interaction. In *Proceedings of IFIP Interact. 84*, 269–274.

RAUTERBERG, M. 1995. From novice to expert decision behaviour: A qualitative modelling approach with Petri nets. *Adv. Human Factors Ergonomics 20*, 449–449.

ROBERTSON, G., CARD, S. K., AND MACKINLAY, J. D. 1989. The cognitive coprocessor architecture for interactive user interfaces. In *Proceedings of the 2nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology*. ACM, 10–18.

RUBIN, J. AND CHISNELL, D. 2008. *Handbook of Usability Testing*. 2nd Ed. Wiley.

SIOCHI, A. C. AND EHRICH, R. W. 1991. Computer analysis of user interfaces based on repetition in transcripts of user sessions. *ACM Trans. Inf. Syst. 9*, 4, 309–335.

SKOV, M. B. AND STAGE, J. 2005. Supporting problem identification in usability evaluations. In *Proceedings of CHI Australia.* ACM Press, 1–9.

SPOOL, J. AND SCHROEDER, W. 2001. Testing web sites: Five users is nowhere near enough. In *Extended Abstracts on Human Factors in Computing Systems (CHI'01)*. ACM, 285–286.

SWALLOW, J., HAMELUCK, D., AND CAREY, T. 1997. User Interface instrumentation for usability analysis: A case study. In *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research*.

THOMPSON, C. 2007. Halo 3: How Microsoft Labs invented a new science of play. *Wired 15*, 9.

TULLIS, T. AND ALBERT, B. 2008. *Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics*. Morgan Kaufmann.

VAN DEN HAAK, M. J. AND DE JONG, M. D. T. 2003. Exploring two methods of usability testing: concurrent versus retrospective think-aloud protocols. In *Proceedings of the IEEE International Professional Communication Conference.*

VIRZI, R. A. 1992. Refining the test phase of usability evaluation: How many subjects is enough? *Hum. Factors 34*, 4, 457–468.

WHARTON, C., BRADFORD, J., JEFFRIES, R., AND FRANZKE, M. 1992. Applying cognitive walkthroughs to more complex user interfaces: Experiences, issues, and recommendations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 381–388.

WINOGRAD, T. AND FLORES, F. 1985. *Understanding Computers and Cognition*. Ablex.

WIXON, D. 2003. Evaluating usability methods: Why the current literature fails the practitioner. *Interactions 10*, 4, 28–34.

Received July 2011, revised December 2011, accepted March 2012