In *Proceedings of AVI '98 (Advanced Visual Interfaces)*. L'Aquila, Italy:, 22-29.

Remote evaluation for post-deployment usability improvement

H. Rex Hartson
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106 USA
Tel: 1 540 231 4857
Email: hartson@vt.edu

José C. Castillo
U S WEST Information Technologies
1801 California St. Suite 1640
Denver, CO 80202 USA
Tel: 1 303 965 2946
Email: jxcast2@uswest.com

ABSTRACT

Although existing lab-based formative evaluation is frequently and effectively applied to improving usability of software user interfaces, it has limitations that have led to the concept of *remote usability evaluation*. Perhaps the most significant impetus for remote usability evaluation methods is the need for a project team to continue formative evaluation downstream, after deployment.

The usual kinds of alpha and beta testing do not qualify as formative usability evaluation because they do not yield detailed data observed during usage and associated closely with specific task performance. Critical incident identification is arguably the single most important source of this kind of data. Consequently, we developed and evaluated a cost-effective remote usability evaluation method, based on real users self-reporting critical incidents encountered in real tasks performed in their normal working environments. Results show that users with only brief training can identify, report, and rate the severity level of their own critical incidents.

Keywords

Remote usability evaluation, evaluation method, user-reported critical incident method, critical incidents, user-initiated, usability data, software deployment

INTRODUCTION

Interactive system developers spend increasing amounts of resources on user interface evaluation conducted in usability laboratories, where a small number of selected users is directly observed by trained evaluators. This laboratory-based formative usability evaluation has become an effective and standard part of iterative user interaction improvement.

Problem Statement

Although existing lab-based formative evaluation is frequently and effectively applied to improving usability of software user interfaces, it has limitations. Project teams want higher quality, more relevant, usability data – more representative of real world usage. The ever-increasing incidence of users at remote and distributed locations (often on the network) precludes direct observation of usage. Further, transporting users or developers to remote locations can be very costly. As the network itself and the remote work setting have become intrinsic parts of usage patterns, the users' work context is difficult or impossible to reproduce in a laboratory setting. These barriers led to extending usability evaluation beyond the laboratory to the concept of remote usability evaluation, typically using the network itself as a bridge to take interface evaluation to a broad range of users in their natural work settings.

Perhaps the most significant impetus for remote usability evaluation methods, however, is the need for a project team to continue formative evaluation downstream, after implementation and deployment. Most software applications have a life cycle extending well beyond the first release. The need for usability improvement does not end with deployment, and neither does the value of lab-based usability evaluation, although it does remain limited to tasks that developers believe to represent real usage.

Fortunately, deployment of an application creates an additional source of real-usage usability data. However, these post deployment usage data are not available to be captured locally in the usability lab. Thus, the need arises for a remote capture method.

In this regard, post-deployment evaluation often brings to mind alpha and beta testing, but these kinds of testing usually do not qualify as formative usability evaluation. Typical alpha and beta testing in the field is accomplished by asking users to give feedback in reporting problems encountered and commenting on what they think about a software application. This kind of post hoc data (e.g., from questionnaires and surveys) is useful in determining

In *Proceedings of AVI '98 (Advanced Visual Interfaces)*. L'Aquila, Italy:, 22-29.

user satisfaction and overall impressions of the software. It is not, however, detailed data observed during usage and associated closely with specific task performance – the kind of data required for formative usability evaluation.

Relevance of Critical Incident Data

This detailed data, perishable if not captured immediately and precisely as it arises during usage, is essential for isolating specific usability problems within the user interaction design. This is exactly the kind of data one obtains from the usability lab, in the form of particular critical incident data and usability problem descriptions.

Despite numerous variations in procedures for gathering and analyzing critical incidents, researchers and practitioners agree about the definition of a critical incident. A critical incident is an event observed within task performance that is a significant indicator of some factor defining the objective of the study [2]. In the context of formative usability evaluation, a critical incident is an occurrence during user task performance that indicates something (positive or negative) about usability.

The origins of the critical incident technique can be traced back to studies performed in the Aviation Psychology Program of the U. S. Army Air Forces in World War II. The technique was first formally codified by the work of Fitts and Jones [5] for analyzing and classifying pilot error experiences in reading and interpreting aircraft instruments. The work of Flanagan [6] became the landmark critical incident technique, after which this technique, often modified, has been thoroughly described by other researchers [10, 11].

Our Goal

Because of this vital importance of critical incident data and the opportunity for users to capture it, the over-arching goal of our work was to develop and evaluate a remote usability evaluation method [3] for capturing critical incident data and satisfying the following criteria:

- tasks are performed by real users,
- users are located in normal working environments,
- users self-report own critical incidents,
- data are captured in day-to-day task situations,
- no direct interaction is needed between user and evaluator during an evaluation session,
- data capture is cost-effective, and
- data are high quality and therefore relatively easy to convert into usability problems.

The result of working toward this goal is the user-reported critical incident method, described below.

What We Learned

The good news from our informal study is that users with no background in software engineering or human-computer interaction, and with the barest minimum of training in critical incident identification, *can* identify, report, and rate the severity level of their own critical incidents. This result is important because success of the user-reported critical incident method depends on the ability of typical users to recognize and report critical incidents effectively.

The bad news is that we found the point in time when users initiate a critical incident report is often significantly delayed after the onset of the critical incident. Because video clips used for context are composed of screen activity captured just before critical incident reporting, the delay in reporting destroys the relevance of the clips, nullifying their value to usability problem analysis. This outcome led to redesign of the video capture method.
More detailed results of this study are described in the "Results and Expectations" section.

THE USER-REPORTED CRITICAL INCIDENT METHOD

Description

The user-reported critical incident method is a remote usability evaluation method for gathering critical incident data from real-world post-deployment usage that satisfies all the above criteria (in "Our Goals"). Critical incident reports are augmented with task context in the form of screen-sequence video clips and evaluators analyze these contextualized critical incident reports, transforming them into usability problem descriptions.

Critical Incident Reporting Tool

A software tool residing on the user's computer is needed to support collection of critical incident reports from users about problems they encounter during task performance. Users are trained to identify critical incidents and use this tool to report specific information about these events.

Whenever usage difficulty is encountered, users click on a *Report Incident* button, a single interface object available from every screen of the application being evaluated. The click activates an instrumentation routine external to the application that:

- opens a textual form, in a separate window from the application, for users to enter a structured report about the critical incident encountered, and
- causes the user's computer to store a screen-sequence video clip showing screen activity immediately prior to clicking the button, for the purpose of capturing the critical incident and events leading up to it*.

* This describes the way the tool is intended for normal use. In order to capture more complete data in our

In *Proceedings of AVI '98 (Advanced Visual Interfaces).* L'Aquila, Italy:, 22-29.

Each contextualized critical incident report is sent asynchronously via the network to evaluators to be analyzed into a usability problem description.

Critical Incident Reports

Data gathered in critical incident reports include the following:

- URL (or location) where user encountered critical incident
- Description of user task in progress when critical incident occurred
- Expectations of user about what system was supposed to do when critical incident occurred
- Detailed description of critical incident (what happened and why user thought it happened)
- Indication of whether user could recover from critical incident and, if so, description of how user did so
- Indication of user's ability to reproduce critical incident
- Severity rating of critical incident
- Additional comments or suggested solutions to problem

Feasibility Case Study

In an earlier case study [8], we determined that the user-reported critical incident method was a feasible method in the sense that it could provide approximately the same amount and value of qualitative data that can be obtained from laboratory-based formative evaluation. The case study employed user subjects, with no prior training in usability methods and only 15 minutes of training to recognize critical incidents during their own usage, plus expert subjects trained in usability methods.

We videotaped all sessions of these user subjects performing tasks and simultaneously identifying critical incidents. A panel of three expert subjects viewed the tapes to detect any critical incidents missed by the user subjects. After the experimenters edited the tapes into sets of video clips, each centered around the critical incident, two expert subjects (different from the first three) analyzed the clips, converting them into usability problem descriptions.

To summarize the results, expert subjects found very few critical incidents missed by user subjects, and those problems missed were of lower severity. Of two video sources, the tape of the scan-converted screen provided the most valuable data for the expert subjects and the experimenter, as compared to a camera view of the user and computer. Also, by informal experimentation we determined that a 60 second video clip centered around a critical incident provided economical coverage for most of the data in this study. Thus, the case study indicated the need to examine the use of screen capture only, reducing the bandwidth requirement over that of continuous video. Results also showed that task information (i.e., about what user subject was trying to do) was essential for the expert subjects to be able to identify associated usability problems and design flaws that led to them.

present study, screen activity was captured continuously via a scan converter and video tape.

RELATED WORK

Traditional Laboratory-Based Usability Evaluation

Traditional laboratory-based usability evaluation is the yardstick for comparison with most new methods. Lab-based evaluation is usually considered "local evaluation" in the sense that user and evaluator are in the same or adjacent rooms at the same time. Data collected are both quantitative (e.g. task performance time) and qualitative (e.g., critical incident descriptions and verbal protocol), the latter serving to identify usability problems and their causes within the interface design [9].

Critical Incident Reporting Tools

Human factors and human-computer interaction researchers have developed software tools to assist identifying and recording critical incident information.

del Galdo et al. [4] investigated use of critical incidents as a mechanism to collect end-user reactions for simultaneous design and evaluation of both on-line and hard-copy documentation. As part of this work, del Galdo et al. designed a software tool to collect critical incidents from user subjects.

Researchers at IBM in Toronto developed a software system called UCDCam, based on Lotus® ScreenCam™. This application, running in Windows3.1™, is used for capturing digitized video of screen sequences during task performance, as part of critical incident reports.

When a user first activates UCDCam, the application opens a "Session Info" window to store the user name, name of users' organization, name of the product being evaluated, and the hard drive where video clips and reports would be stored. While users work with their normal tasks, UCDCam runs as a "background" process, continuously recording all screen activity in a current buffer, and it also retains a separate holding buffer that holds the two-minutes of screen activity that occurred prior to the initialization of the current buffer. To report critical incidents, users click a button that opens an editor for entering comments about the problem. Users make selections from various list-boxes to indicate what they were trying to do when the problem occurred (i.e., user task) and to rate critical incident severity. List box entries (possible choices) are configurable by the evaluator. UCDCam also counts the number of reports sent by each user.

In *Proceedings of AVI '98 (Advanced Visual Interfaces)*. L'Aquila, Italy:, 22-29.

UCDCam automatically saves a screen-sequence clip of all activity that occurred for an interval prior to clicking the button (current n-second buffer plus two-minute previous buffer if the current buffer is less than one-minute long). Approximately 200 KB is required to store one minute of screen action in the user's computer, depending on display resolution and amount of screen updates. If the user has not pressed the "Incident" button within the two-minute buffer interval, then a new current buffer is initialized, and the old current buffer is used to replace the old holding buffer. That way, the most recent interval of screen action is always captured at any point in time, with screen-clips of 1 to 2 minutes in duration. Buffer duration is static, but configurable by the evaluators (up to 20 minutes in length).

The intention is that UCDCam will automatically package a screen-sequence clip with user comments (textual report) and send this package of information via the network to evaluators. Evaluators then use UCDCam to watch the clips, analyze activity that occurred prior to when the user reported an incident, and create usability problem descriptions.

We did, in fact, consider UCDCam for this study but, instead, used continuous video to record users during evaluation sessions to analyze some timing problems discussed later. This decision turned out fortunate, indeed, since much of the important data was outside the range of the capture interval we intended to use.

Other Remote Usability Evaluation Techniques

Remote evaluation is defined as usability evaluation where evaluators are separated in space and/or time from users [8]. The term *remote* is used relative to the developers and refers to users not at the location of developers. Similarly, the term *local* refers to location of the developers.

Space limitations preclude all but the briefest review of some different types of remote evaluation methods, which include the following:

Remote Questionnaire or Survey. Without any added instrumentation, evaluators can send questionnaires to users (via mail or email) or can give users the URL to a Web-based questionnaire. In an approach more directly associated with usage, software applications can be augmented to trigger the display of a questionnaire to gather subjective preference data about the application and its interface (e.g., User Partnering Module® from UP Technology® [1]).

Live or Collaborative Remote Evaluation. In collaborative usability evaluation via the network [7], evaluators at a usability lab are connected to remote users via commercially available teleconferencing software, as an extension of the video/audio cable [8]. Typical tools of this kind (e.g., Sun Microsystems® ShowMe™, Microsoft® Netmeeting™) support real-time application sharing, audio links, shared drawing tools, and/or file transfer capabilities. Other tools like TeamWave® Workplace™ also include a shared space or virtual room for a work group.

Instrumented or Automated Data Collection for Remote Evaluation. An application and its interface can be instrumented with embedded metering code to collect and return a journal or log of data, such as Internet usage, program usage, keystrokes and mouse movements, occurring as a natural result of usage in users' normal working environments (e.g., ErgoLight Usability Software® ErgoLight™, the WinWhatWhere™ family of products). The logs or journals of data are later analyzed using pattern recognition techniques [12] to deduce where usability problems have occurred.

Commercial Usability Services. Developers use the network to communicate design documents, software samples, and/or prototypes to remote contractual evaluators, but the network is not used to connect to remote users. The evaluation is local to the contractual evaluators and remote from the developers, with results being returned via the network (e.g., Vertical Research, Inc. [13] for Microsoft® Windows™-based products).

THE INFORMAL STUDY

We performed a study that we describe as informal because, although we obtained quantitative data, it was not the kind of summative study that uses statistically significant results to prove or refute an experimental hypothesis. Rather, it was an exploratory study to gain insight and understanding, under practical operating conditions, about the strengths and weaknesses of the method.

Objectives of the Study

In particular, the primary research questions were:

- Can users report their own critical incidents and how well can they do it?
- Can evaluators use critical incident data to produce usability problem descriptions and how well can they do it?
- What are the variables and values that make the method work best?

Pilot Study

Three volunteer pilot subjects assisted in pre-testing for this study. Two pilot subjects, who were human-computer interaction researchers trained in usability methods, helped the experimenter evaluate the overall setup of the study. A third pilot subject, an undergraduate student who had no prior training in usability methods, pre-tested the evaluation tasks to ensure that they could be performed by typical users. In addition, this pilot subject tested a Web-based tool that user subjects would use to report critical incidents during performance of the evaluation tasks.

In *Proceedings of AVI '98 (Advanced Visual Interfaces)*. L'Aquila, Italy:, 22-29.

Phase I: Critical Incident Gathering

Participants. Twenty four students (6 female and 18 male, 22 undergraduate and 2 graduate) participated in the study as volunteer user subjects. The experimenter administered a background questionnaire to students from two introductory Computer Science courses for non-CS-majors (with students from many disciplines) and selected user subjects based on a minimum knowledge of Web browsing and Web-based information retrieval. No experience was required with the Internet Movie Database, the application evaluated in the study.

Equipment. The best location for users in a study of a remote evaluation method is their own workplace. However, the study itself (not the user-reported critical incident method) required a scan converter and videotape deck to make a complete continuous recording of the computer screen during task performance. Since it was not feasible to lend this equipment to each user subject, the experimenter provided the next best thing for the user subject: a closed and quiet room, isolated from other people, including the experimenter. (Once the user-reported critical incident method is fully developed and disseminated, screen capture software and disk storage will suffice for any user in any location.)

The experimenter was located in a room adjacent to that of the user subjects. An intercom system was installed in both rooms as a safety net in case user subjects experienced any hardware or software problems that prevented continuing with the tasks, but the evaluator did not have any interaction with user subjects during task performance. A scan converter, lapel microphone, and Hi-8 videotape deck recorded video from the screen and audio from the user subject.

The application evaluated in this study is the Internet Movie Database (IMDb) at http://www.imdb.com, a Web-based service providing extensive, free access, movie information. The IMDb search page, which contains mechanisms for simple and advanced searching over 100,000 movies with over 1,500,000 filmography entries, is financed by advertising and sponsorship.

The critical incident reporting tool is a Web-based tool that allows user subjects to send structured reports about critical incidents they identify during their experimental session. The Remote Evaluation Control window contains the *Report Incident* button and "floats" on the desktop, running independently from the application. User subjects use the mouse to arrange the application window and the control window so that they can see some of each window on the screen.

Clicking on the *Report Incident* button, activates an instrumentation routine that opens the Remote Evaluation Report window, which contains a form with questions about the critical incident. These questions (see the previous section "Critical incident reports" under "The user-reported critical incident method") provide content-based structure for consistently gathering information so that evaluators can easily transform it into usability problem descriptions.

The Remote Evaluation Report window is independent from the application window, allowing user subjects to click back and forth between the report and application windows to work on both the task and the critical incident report.

Protocol. We gave all user subjects brief training, followed immediately by a practice session in which they gained experience identifying and reporting critical incidents via the Web-based tool while performing a representative task using the Internet Movie Database.

During the study, each user subject performed the same six search tasks using the Internet Movie Database. These tasks were created by the experimenter to approximate tasks that a typical user would perform with the movie database (e.g., "Find the titles of the four most recent movies directed by Steven Spielberg"). User subjects wrote their retrieved responses to these queries on a participant answer sheet.

Data Collection and Analysis. During these tasks, users reported each critical incident they believed they encountered, using the Remote Evaluation Report window. Following the evaluation session, each user subject completed a questionnaire asking about the experience as a remote user. After all data gathering was completed, the experimenter reviewed the 24 one-hour videotapes twice, tagging and coding critical incident data. Additionally, the experimenter randomly selected one good (i.e., complete and precise) critical incident report for each of the six tasks and extracted six corresponding video tape clips, producing contextualized critical incident packages for evaluator subjects in Phase II.

Phase II: Transformation of Critical Incident Data Into Usability Problem Descriptions

Participants. Four volunteer participants (two graduate students from Computer Science and two from Industrial and Systems Engineering, all trained in usability methods) served as evaluator subjects. Their role was to analyze selected critical incident reports sent by user subjects and convert them into usability problem descriptions.

Equipment and Materials. Critical incident reports were presented to all evaluator subjects on paper. Two of the evaluator subjects received only the paper reports. Two subjects also used a VCR and a video monitor to watch video clips containing visual context for critical incident reports.

Protocol and Data Collection. Two evaluator subjects each created a list of usability problem descriptions by analyzing six contextualized critical incident packages (reports and video clips). The other two evaluator

In *Proceedings of AVI '98 (Advanced Visual Interfaces)*. L'Aquila, Italy:, 22-29.

subjects analyzed only the critical incident reports (no video clips). In addition, all four evaluator subjects completed a questionnaire about their experience as evaluators in a remote usability situation.

Data Analysis. The experimenter examined the four lists of usability problem descriptions created by evaluator subjects and analyzed their questionnaires to determine:

- the feasibility (i.e., time and effort) of transforming remotely-reported critical incident data into usability problem descriptions;
- the level of agreement about severity ratings of critical incidents between user subjects and evaluator subjects;
- the quality of the content in critical incident reports; and
- the role of video, text, and audio during analysis of critical incident data.

As a general matter, the small number of evaluator subjects and individual differences among them strongly colored the results of this phase of the study, implying a low significance for results. Therefore, the "results" obtained here are only hypotheses and not conclusions, and will serve as expectations for a future study.

Expectations and Results

The results are reported here as a series of expectations, each given as a header in italics followed by a discussion of the results.

We expected user subjects to be able to report their own critical incidents during task performance.

Across all user subjects, the experimenter found 97 critical incidents (Figure 1): 66 also reported by user subjects and 31 identified only by the experimenter (mostly of low severity). User subjects sent a total of 74 critical incident reports (a mean of 3.1 reports per user subject, standard deviation of 1.7). Interestingly, eight low severity critical incidents were reported by user subjects in cases where the experimenter did not recognize from review tapes that user subjects were experiencing a critical incident. The experimenter did not, however, consider these as gratuitous reports sent to please the experimenter, concluding that these critical incidents were known in the minds of the user subjects but not evident visually in the videotapes. Nevertheless, these 8 reports were not considered during data analysis.

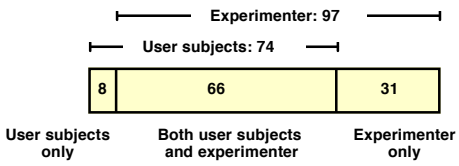

Figure 1. Number of critical incidents identified by user subjects and experimenter

For most cases, we expected the experimenter to agree with severity ratings made by user subjects.

Users made severity ratings on a scale of one through five, with one being the lowest severity and five the highest. As an abstraction, the experimenter converted the ratings to severity rankings, where the low severity rank corresponds to ratings one and two, medium severity rank corresponds to rating three, and high severity rank corresponds to ratings four and five.

Across all 24 user subjects, the experimenter's rankings agreed with those of users subjects for 55 out of 66 (83%) of the critical incidents reported by both user and evaluator subjects. The others were balanced, with six reports being lower severity than the experimenter and five higher (Figure 2). Thus, we conclude that users can rate self-reported critical incidents with reasonable accuracy compared to an expert evaluator.

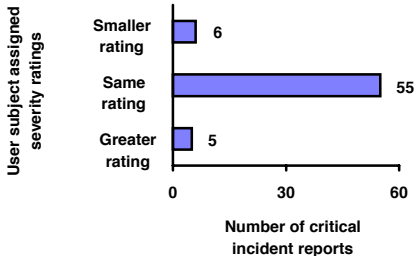

Figure 2. User subject severity ratings compared to the experimenter's rating

We expected user subjects to identify the majority of critical incidents, across all severity levels, as identified and ranked by the experimenter.

User subjects mostly met expectations by reporting 21 out of 28 (75%) of the critical incidents identified by the experimenter as high severity (Figure 3), 19 out of 24 (79%) medium severity critical incidents, and 15 out of 45 (33%) low severity ones, as ranked by the experimenter.

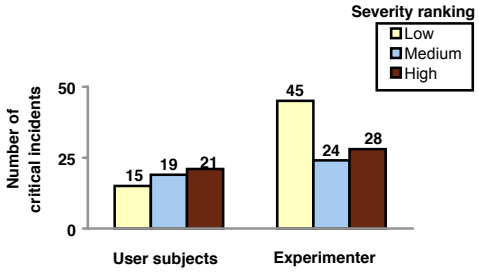

Figure 3. Number of reported critical incidents by severity ranking

Thus, user subjects identified 40 out of 52 (77%) of the important (medium and high severity) critical incidents. Further, the experimenter found that 26 of the 31 critical

In *Proceedings of AVI '98 (Advanced Visual Interfaces).* L'Aquila, Italy:, 22-29.

incidents not reported by user subjects were of low severity, the least important ones.

In sum, results indicated that users working in remote environments and lacking interaction with evaluators are capable of self-reporting critical incidents encountered during task performance. Specifically, the study indicated that users are generally capable of identifying high and medium severity critical incidents and most of the critical incidents missed by users, but which might be identified by an expert, were low severity.

We expected the flow of the activities during task performance and reporting to be somewhat structured; in particular, we expected most users subjects will report critical incidents immediately after they occur.

The expectation for a structured flow of the reports during task performance (e.g., task performance, critical incident identification, and reporting) was not met in most cases. Not surprisingly, high severity critical incidents had the most disruptive impact on task performance and flow of activities. Eleven user subjects sent high severity critical incident reports for tasks they never completed. Sometimes when encountering a critical incident, user subjects gave up and continued to the next task without any effort to complete the current task. Sometimes they jumped to the next task but later came back to work on the troublesome task (with or without success).

We observed considerable further variation in user subject behavior with regard to timing of critical incident reports. User subjects reported critical incidents during task performance, immediately after the task ended, at a later time, and sometimes never did report the critical incident. Although we directed user subjects to send a report immediately after encountering a critical incident, they sent 52 (or 70%) of all 74 critical incidents reports (Figure 4) *after* the task ended.

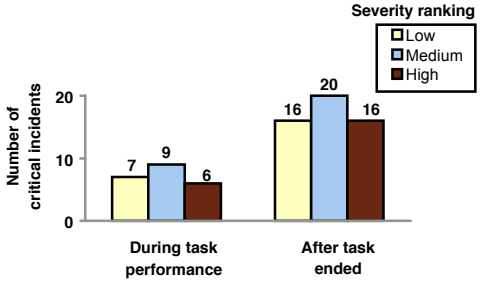

Figure 4. Most critical incident reports were sent after task completion

Possible reasons for waiting until completing the task to report a critical incident include the natural desire for closure on one task before doing another, the desire to wait until enough information was acquired to make a complete report, and the desire to avoid interference of the report with task performance.

In an attempt to learn more about the nature of the timing issues, the experimenter reviewed all 24 videotapes yet again. For each critical incident report, the experimenter determined the point in time when it was first evident that a critical incident had occurred. As illustrated in Figure 5, the average time interval between this time and the point of reporting was 2 minutes and 42 seconds (standard deviation of 5.5). The shortest delays occurred when reporting low severity critical incidents (mean of 24 seconds, standard deviation of 0.5), the longest for high severity (mean of 4.5 minutes, standard deviation of 6.6).

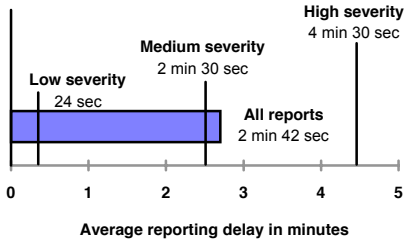

Figure 5. Average delay in reporting after clear onset of critical incidents

Delays roughly corresponded with severity of the critical incident. The presumption is that a more severe critical incident requires more information to report and, therefore, results in a larger delay before reporting.

Based on the results of our feasibility case study, we expected user subjects to spend an average of about 3 minutes typing critical incident reports.

As seen in Figure 6, user subjects took significantly longer to make critical incident reports, spending an average of 5 minutes and 24 seconds (standard deviation of 2.3) per report, from a minimum time of 2 minutes and 5 seconds to a maximum of 12 minutes and 15 seconds.

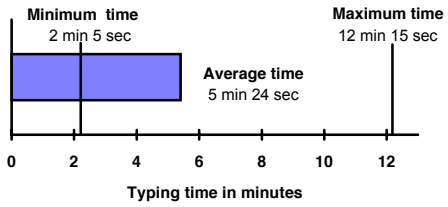

Figure 6. Average typing time for critical incident reports

User subjects who reported critical incidents during task performance spent more time (mean of 6 minutes and 42 seconds, standard deviation of 2.7) typing critical incident reports than those user subjects who waited until the task ended (mean of 4 minutes and 48 seconds, standard deviation of 1.6).

We expected user subjects generally will want to report critical incidents anonymously.

Contrary to the expectations of this study, 17 (or 71%) of the 24 user subjects disagreed that anonymity was important for them, mainly because the desire for feedback was incompatible with anonymity.

We expected an indication from some user subjects that identifying and reporting critical incidents interfered with their tasks (i.e., at least in some cases of high severity critical incidents), and that longer reports would lead to a stronger perception of interference.

Usage problem reporting by remote users working on real tasks for real work has considerable potential to interfere with task performance. However, contrary to our expectations, 19 (or 79%) of the 24 user subjects moderately or strongly agreed that identifying and reporting critical incidents was not intrusive and did not interfere with their tasks. This counter-intuitive result was reinforced by the fact that some of the best and most complete critical incident reports came from user subjects who said that they felt that reporting did not interfere much. Perhaps the feeling of interference was offset by the satisfaction of being able to identify and report problems.

We expected evaluator subjects to be able to analyze critical incident data

Generally, all evaluator subjects were capable of analyzing critical incident data to produce usability problem descriptions. For example, report-only evaluator subjects reported similar or related usability problem descriptions for five out of six critical incident reports. Clip-and-report evaluator subjects showed similar results in their ability to produce usability problem descriptions. However, differences among the lists of usability problem descriptions among the 4 evaluator subjects made it difficult to compare them (e.g., one evaluator subject's list described usability problems found for each task while the other list described usability problems found in the user interface as a whole).

Because of the additional data, we expected more analysis effort required of clip-and-report evaluator subjects than of report-only evaluator subjects.

Contrary to expectations, clip-and-report evaluator subjects took somewhat less analysis time on average than report-only evaluator subjects. Report-only evaluator subjects spent an average of 1 hour and five minutes analyzing all six critical incident reports into usability problem descriptions (average 10 minutes and 50 seconds analyzing each report). On the other hand, clip-and-report evaluator subjects spent an average of 50 minutes analyzing critical incident data (textual reports and videotape clips) and creating a list of usability problem descriptions (average of 8 minutes and 20 seconds analyzing each pair of critical incident reports and videotape clips). It is possible that report-only evaluator subjects had to spend more time analyzing critical incident data because they had less information, and therefore needed more time to understand the critical incidents.

We expected user subjects might prefer having the* Report Incident *button and critical incident reporting tool built into the application being evaluated.

For this study, we implemented the reporting tool as an independent piece of software, separate from the application. It also could have been built-in as part of the Internet Movie Database software application. Building it in requires modification of the Internet Movie Database software, not a practicable alternative in the study, since the experimenter did not have access to the source code. For reasons of convenience and logical clarity, however, 16 (or 89%) out of 18 users subjects expressed a preference for having the *Report Incident* button built into the application being evaluated, rather than in a separate window.

Lessons Learned About Design

The observations in this part of the study indicated a need for much more flexibility in structure, quantity, format, and timing of reports than was expected. A new design (Figure 7) for the critical incident reporting tool addresses these concerns simultaneously.

Need to De-Couple Critical Incident Identification From Reporting. Perhaps the most significant observation in the study involved the delay in reporting critical incidents, as described above. Automatic capture of the screen-sequence video clips that accompany critical incident reports as visual context requires some trigger mechanism to initiate the capture process. We intended to use the clicking by the user on the *Report Incident* button to be this trigger mechanism.

However, the results of this study show a highly variable, and often large, delay between the time when the video clip should begin at the clear onset of a critical incident and the time when user subjects click the *Report Incident* button. The result is often clips that contain user actions irrelevant to the critical incident. Thus, this study reveals that a different trigger mechanism, separate from the

Figure 7. New critical incident reporting tool

In *Proceedings of AVI '98 (Advanced Visual Interfaces)*. L'Aquila, Italy:, 22-29.

Report Incident button is required. That trigger mechanism is the *Begin Incident Capture* button of the new design, as shown near the top of the window in Figure 7, to be clicked by a user at the inception of a critical incident, when an occurrence of difficulty is beginning, even though information may yet be insufficient for effective reporting. Users who understand how the mechanism works can even "re-enact" a critical incident for capture. Knowing that recording is triggered by this button, users can do something on the screen that they want the evaluator to see and then click the button. Later, when ready, users can click the *Send Full Report* button to send a detailed report of the incident.

Need for Short, Quick Reports. Users expressed a need for more than one kind of problem report. For situations where is not desirable to send a complete critical incident report (e.g., the critical incident is not seen as important enough), users asked for a "quick report" capability, which is accommodated in the new design via the *Send Quick Comments* button.

Need to Browse and Review Previous Reports. Users indicated a need for support in situations where they have identified a critical incident but are not sure whether they have reported that particular incident earlier.

This problem is solved in the new design by way of a new feature in the critical incident reporting tool. A list (at the bottom of the window in the figure) is used to keep track of reports previously sent by a user. Reports related to a given task can be selected from this list to be browsed in read-only mode.

SUMMARY

We developed a remote usability evaluation method as a means to continue formative usability evaluation after deployment of a software application. The method is based on user-reported critical incident data, captured during usage in real task performance. In an informal study we evaluated the method and determined that users with a minimum of training can identify, report, and rate the severity level of their own critical incidents.

We also discovered a delay between the time many users encounter a critical incident and the time they initiate reports. This delay defeated our mechanism for capturing, as context, a video clip of screen activity. A new design for the critical incident reporting tool addressed this and other problems revealed by the study.

REFERENCES

1. Abelow, D. Automating Feedback on Software Product Use. *CASE Trends*. (December 1993), 15-17.

2. Andersson, B. E., and Nilsson, S. G. Studies in the Reliability and Validity of the Critical Incident Technique. *Journal of Applied Psychology*. *48*, 6 (1964), 398-403.

3. Castillo, J. C., and H. R. Hartson, "Remote evaluation." Internet WWW page at: <http://hci.ise.vt.edu/~josec/remote_eval/>. (Most recently accessed: 1/28/98).

4. del Galdo, E. M., Williges, R. C., Williges, B. H., and Wixon, D. R. An Evaluation of Critical Incidents for Software Documentation Design. In Proceedings of *Thirtieth Annual Human Factors Society Conference* Human Factors Society, Anaheim, CA, 1986, 19-23.

5. Fitts, P. M., and Jones, R. E. Psychological Aspects of Instrument Display: Analysis of Factors Contributing to 460 "Pilot Error" Experiences in Operating Aircraft Controls. Reprinted in *Selected Papers on Human Factors in the Design and Use of Control Systems (1961)*. Sinaiko ed. Dover Publications, Inc., New York, 1947, 332-358.

6. Flanagan, J. C. The Critical Incident Technique. *Psychological Bulletin*. *51*, 4 (1954), 327-358.

7. Hammontree, M., Weiler, P., and Nayak, N. Remote Usability Testing. *interactions*. (July 1994), 21-25.

8. Hartson, H. R., Castillo, J. C., Kelso, J., Kamler, J., and Neale, W. C. Remote Evaluation: The Network as an Extension of the Usability Laboratory. In Proceedings of *CHI Conference on Human Factors in Computing Systems* ACM, New York, 1996, 228-235.

9. Hix, D., and Hartson, H. R. *Developing User Interfaces: Ensuring Usability Through Product & Process*. John Wiley & Sons, Inc., New York, 1993.

10. Meister, D. *Behavioral Analysis and Measurement Methods*. Wiley, New York, 1985.

11. Shattuck, L. W., and Woods, D. D. The Critical Incident Technique: 40 Years Later. In Proceedings of *38th Annual Meeting of the Human Factors Society*, 1994, 1080-1084.

12. Siochi, A. C., and Ehrich, R. W. Computer Analysis of User Interfaces Based on Repetition in Transcripts of User Sessions. *ACM Transactions on Information Systems*. *9*, 4 (October 1991), 309-335.

13. Vertical Research, Inc. Home Page, "About our Remote Inspection Service." Internet WWW page at: <http://www.vrix.com/serv/rem_insp.htm>. (Most recently accessed: 1/22/98).