THE USER-REPORTED CRITICAL INCIDENT METHOD AT A GLANCE*

José C. Castillo, H. Rex Hartson, Deborah Hix
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061-0106
josec@hci.ise.vt.edu, {hartson, hix}@vt.edu

ABSTRACT

The over-arching goal of this work is to discuss the user-reported critical incident method, a cost-effective remote usability evaluation method for real-world applications involving real users, doing real tasks in real work environments. Several methods have been developed for conducting usability evaluation without direct observation of a user by an evaluator. However, contrary to the user-reported critical incident method, none of the existing remote evaluation methods (nor even traditional laboratory-based evaluation) meets all the following criteria:

- data are centered around critical incidents that occur during task performance,
- tasks are performed by real users,
- users are located in normal working environment,
- 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,
- there is a cost-effective way to capture data, and
- data are high quality and therefore relatively easy to convert into usability problems.

---

* This paper is based on the fourth chapter of the thesis work by Castillo (1997).

OVERVIEW OF METHOD

The user-reported critical incident method is a usability evaluation method that involves real users located in their own working environment, doing everyday tasks, and reporting critical incidents without direct interaction with evaluators. 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.

RELEVANCE OF CRITICAL INCIDENT INFORMATION

In real world task performance, users are perhaps in the best position to recognize critical incidents (specifically negative critical incidents) caused by usability problems and design flaws in the user interface (Figure 1). Critical incident identification is arguably the single most important kind of information associated with task performance in a usability-oriented context. This information is perishable, however, and can be lost if not captured immediately as it arises during usage. Whether gathered by users or evaluators, once critical incident information is collected, evaluators analyze it and create a list of usability problem descriptions used by interaction designers to determine design solutions and correct the problem.

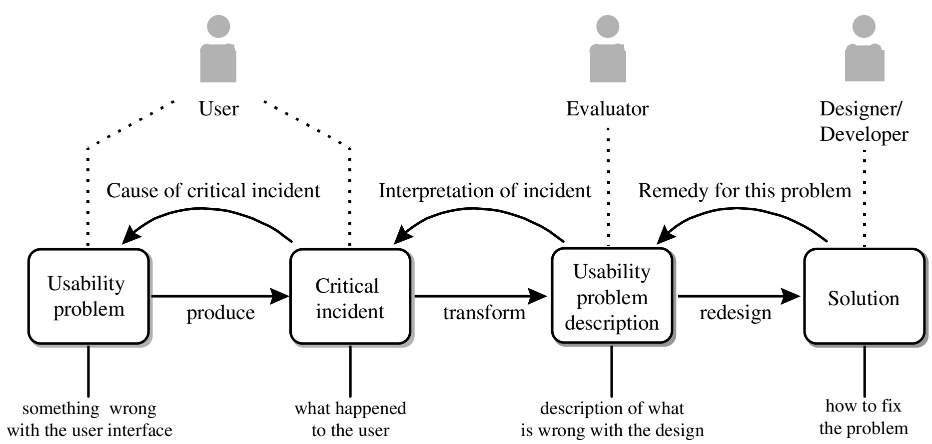

Figure 1. Overview of process for improving interaction design

RELEVANCE OF USER TRAINING

Success of the user-reported critical incident method depends on the ability of typical users to recognize and report critical incidents effectively, but there is no reason to believe that all users have this ability naturally. Therefore, this study was designed with the assumption that some training would be required, an expectation supported by an initial case study discussed by Hartson, Castillo, Kelso, Kamler, and Neale (1996).

Fitts and Jones (1947) obtained detailed factual information about “pilot-error” experiences in reading and interpreting aircraft instruments from people not trained in the critical incident technique (i.e., an eyewitness or the pilot who made the error). In contrast, Flanagan (1954) employed trained observers (not users), who were domain knowledgeable data collectors, making observations of ongoing activities in the user’s normal working environment. Laboratory-based formative usability evaluation has critical incident identification done by an evaluator trained in human-computer interaction, which might lead to skepticism about casting a user in that role. However, in fact, the critical incident technique has been successfully adapted for human-computer interaction so that critical incidents are identified by untrained users during their own task performance (del Galdo, Williges R.C., Williges B.H., and Nixon, 1986). Although users cannot be expected to be generally trained in human-computer interaction, the work reported here shows that users can play a more valuable role in usability evaluation if they are given minimal training for the specific task of identifying and reporting critical incidents.

DESCRIPTION AND EVOLUTION OF METHOD

DESCRIPTION OF METHOD

The user-reported critical incident method is illustrated in Figure 2.

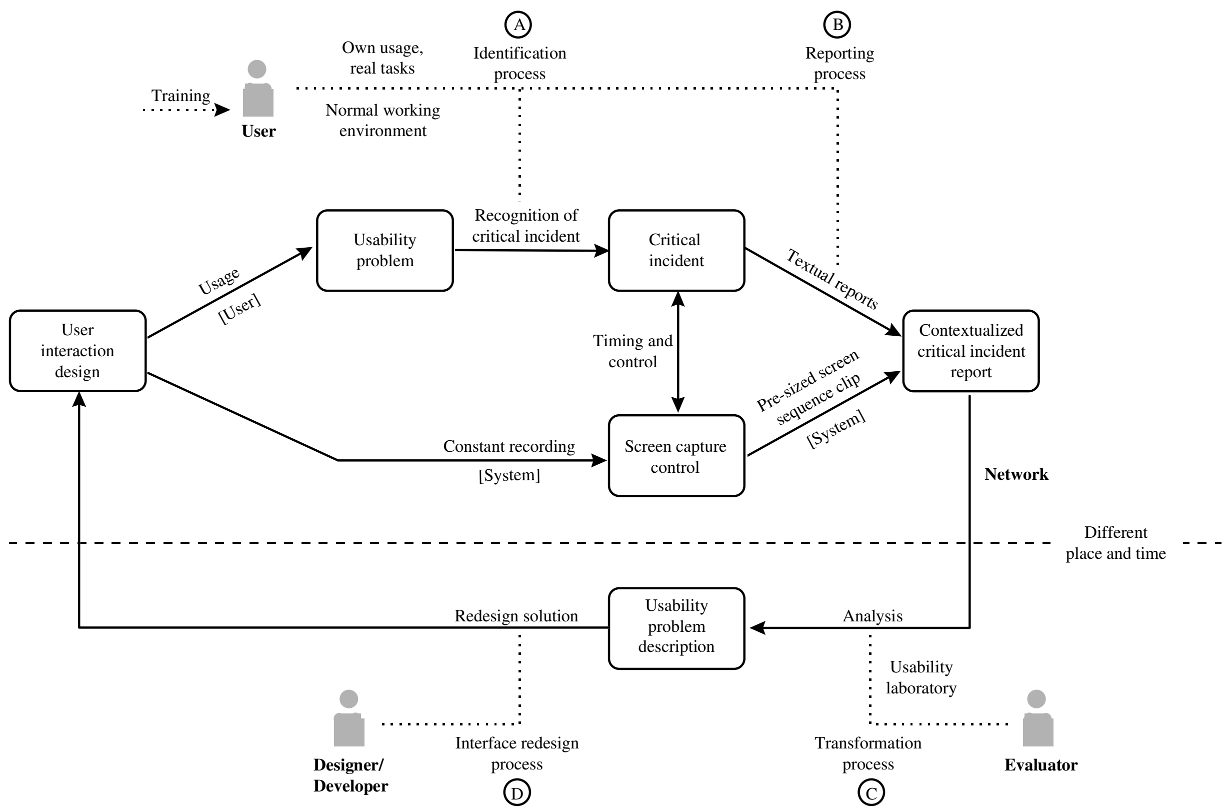

Figure 2. Setup for the user-reported critical incident method

Users are located in their own working environment, identifying critical incidents as they occur during the normal course of on-the-job task performance (at point A). Whenever usage difficulty is encountered, the user clicks on a *Report Incident* button (at point B), a single added object consistently appearing on all screens of the application being evaluated. The click activates an instrumentation routine (e.g., software outside 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 system to store a screen-sequence video clip showing screen activity immediately prior to clicking the button (hopefully capturing the critical incident and events leading up to it).

This package of data, the critical incident report and the screen-sequence clip taken together, is called a contextualized critical incident report and is sent asynchronously via the network to evaluators to be analyzed into usability problem descriptions (at point C). Outside the scope of this evaluation method, designers use problem descriptions to drive redesign solutions (at point D), which go back into the interaction design.

COMPARISON WITH LABORATORY-BASED USABILITY EVALUATION

Figure 3 depicts a view of the traditional laboratory-based formative evaluation process for comparison with the setup for the user-reported critical incident method.

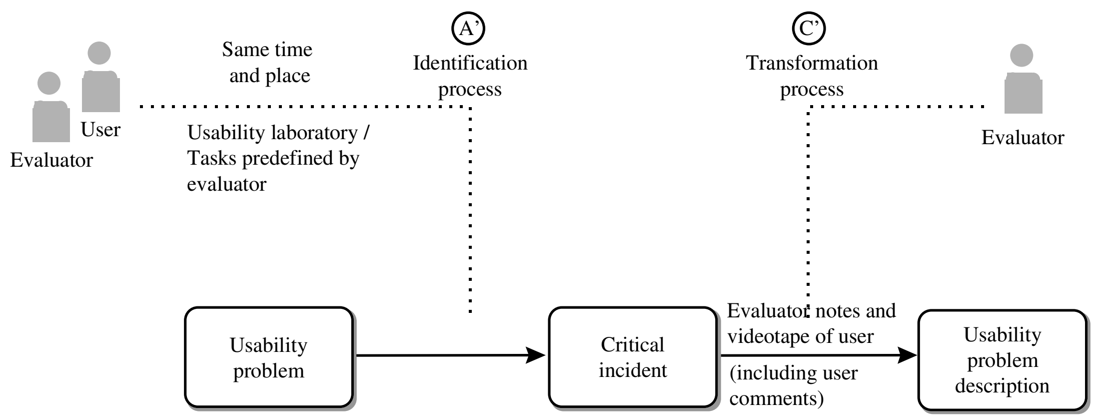

Figure 3. Configuration for traditional laboratory-based usability evaluation

While directly observing a user during task performance, an evaluator produces a list of critical incidents (at point A'), and later analyzes it to create a list of usability problem descriptions (at point C'). Users often help evaluators identify critical incidents by talking aloud and explaining usage difficulties they encounter (at point A'). However, users themselves do not produce a list of critical incidents and/or report critical incidents (e.g., typing a report to describe the problem) to evaluators

CONTEXTUAL FACTORS FOR CRITICAL INCIDENTS

Dzida, Wiethoff, and Arnold (1993) discussed various contextual factors that can be associated with a critical incident. This present study also uncovered some other similar factors, resulting in the combined list that follows:

- specific beginning and ending of a critical incident,
- task or goal of a user just before critical incident occurs,
- expectations of user just before critical incident occurs,
- detailed description of critical incident (what exactly happened, what caused it),
- severity of the critical incident,
- frequency of the critical incident,
- indication from the user of any knowledge on how to recover from critical incident, and
- potential design solutions to solve the problem.

SCREEN-SEQUENCE VIDEO CLIPS AND TIMING ASPECTS

One challenge for this study was to collect screen-sequence video clips containing meaningful information about usability problems that caused critical incidents. To ensure that the captured information was useful, it was necessary to determine:

- an effective point in time to start recording the clip, and
- an effective length of the clip.

The effective clip length is reserved for future study. In any case, the recording time of screen sequences must be of fixed length because the screen sequence capture software (e.g., IBM® UCDCam®, Lotus® ScreenCam™) cannot be adaptive in the field (i.e., cannot produce video clips of varying length depending on type of critical incident). Longer length gives a higher probability that the most useful data will be included. Shorter length allows evaluators to spend less time and effort analyzing the clip into usability problem descriptions, and requires less bandwidth to transmit each clip (e.g., at least 200 KB per minute of clip) over the network.

Because the user-reported critical incident method sends, as part of a critical incident report, a screen-sequence clip of action occurring just *before* the user clicks the *Report Incident* button, a mechanism is required to ensure that this recent history of user and system actions is available for capture when the user does click the button. To accomplish this, the instrumentation routine continuously records all screen actions in a background process. When the user clicks the *Report Incident* button, the routine stores (in the user's computer) a clip containing, say, the last two minutes of screen usage.

The chronology of events associated with a user-reported critical incident starts the moment a user begins to have difficulty with a task. At this point, the user may not yet have recognized it as a critical incident. Eventually, the user recognizes that the situation or event is a critical incident, but has not yet sent a report.

After the user is sure this event is a critical incident, and after enough is known about it to make a report, the user clicks on the *Report Incident* button and an interval of screen-sequence action occurring just prior to that point is captured as the video clip component of the report. Storage and bandwidth requirements impose economic pressure to keep the clip length short. Thus, the key factor in getting the most useful data (in terms of being able to identify the associated usability problem) is the starting point of the clip. Unfortunately, in practice the best starting point can occur over a broad range of time, and no method can be devised to automatically detect this ideal reporting point. As discovered in this study, users often wait until the most useful screen-sequence data have disappeared before they initiate a report.

A possible solution to solve this problem is to separate the act of identifying critical incidents from the act of reporting them, and to encourage users to click a button labeled to the effect of "I am just starting to recognize the occurrence of a critical incident" immediately after they recognize the critical incident (close to the effective beginning of the critical incident), but to click on the *Report Incident* button to report the incident (i.e., create a report) at a later time when more is known about the nature of the critical incident. See Castillo (1997) for more discussion of this difficult problem.

POSSIBLE APPLICATIONS OF METHOD

EARLY FORMATIVE EVALUATION

Early in the software development process (i.e., after paper prototyping during iterative development), selected users can be involved in evaluating prototypes remotely. These users would identify critical incidents during performance of tasks they choose to represent their own work patterns and a set of representative tasks predefined by evaluators, while using the user-reported critical incident method to convey critical incidents to developers.

ALPHA, BETA, AND OTHER FIELD USABILITY EVALUATION

Before new software is released to the market, selected users can assist in evaluation of alpha and beta versions. Rather than the usual "Try it and let us know what you think", remote users, using the user-reported critical incident method, can send data about software bugs and critical incidents of usage that are potentially as useful as qualitative data collected in a usability laboratory.

CUSTOMER SUPPORT

After software is released to the market, the user-reported critical incident method can be utilized to assist in customer support services (e.g., help desk application). Instead of calling a customer support telephone number or sending email to developers, remote users, via this method, now have a more immediate and direct channel for feedback to developers, giving developers more detailed and specific information about systems and interface weaknesses while getting the help they need. The appropriate person or team of developers can receive the report, solve the specific problem, and email solutions back to users.

MARKETING STRATEGIES

Subjective data (e.g., from a satisfaction questionnaire) can be broadened by marketing personnel into market-oriented product reviews revealing:

- user needs,
- new features users want added to the application,
- features that have not been used, and
- ideas to make new products.

REFERENCES

Castillo, J. C. (1997). *The User-Reported Critical Incident Method for Remote Usability Evaluation.* Unpublished thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.

del Galdo, E.M., Williges, R.C., Williges, B.H., & Nixon, D. (1986). An Evaluation of Critical Incidents for Software Documentation Design. *Proceedings of the 30th Annual Meeting of the Human Factors Society*, 19-23.

Dzida, W., Wiethoff, M., & Arnold, A.G. (1993). *ERGOGuide: The Quality Assurance Guide to Ergonomic Software*. Joint internal technical report of GMD (Germany) and Delft University of Technology (The Netherlands).

Elgin, Bruce (1995). Subjective Usability Feedback from the Field over a Network. *SIGCHI Bulletin* (October), 27 (4), 43-44.

Fitts, P.M. & Jones, R.E. (1947). Psychological Aspects of Instrumented Display: Analysis of 270 “Pilot-Error” Experiences in Reading and Interpreting Aircraft Instruments. In H.W. Sinaiko (Ed.), *Selected papers on human factors in the design and use of control systems* (pp. 1-38). New York: Dover Publications, Inc.

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

Hammontree, M., Weiler, P., & Nayak, N. (1994). Remote Usability Testing. *Interactions* (July), 21-25.

Hartson, H.R., Castillo, J.C., Kelso, J., Kamler, J., & Neale, W.C. (1996). Remote Evaluation: The Network as an Extension of the Usability Laboratory. *Proceedings of CHI '96 Human Factors in Computing Systems*, 228-235.