Proposal for participation in the workshop on

New Methods and Practices in Usability Testing

Avi Harel

ErgoLight Ltd., Haifa, Israel

Experience conducting usability testing

Between the years 1975 and 1991 I worked for Rafael, the main R&D institute of the Israeli Ministry of Defense. Between the years 1980-1983 I was the project manager for a major project in the Electronic Division. The main user tasks included data entry and data manipulation. A primary design concern was the user performance. A primary testing concern was to prevent user errors that might result in accidents. In these days I was not yet familiar with usability concepts. However, as usability was a key concern for our project, it was my job to conduct informal tests with the participation of potential users. Between the years 1984-1985, during a sabbatical leave of absence from Rafael, I worked for the DI department of BNR (now Nortel) on the design of a prototype of a touch sensitive telephone set. This was my first acquaintance with formal usability testing. In 1986 I was the first person in the Electronic Division whose job was usability evaluation. Between the years 1987-1991 I worked for the Human Factors department of Rafael, where I conducted various comparative tests with the participation of real users. The user tasks in these tests were video-based navigation and target acquisition and CGI based guidance control. The dominant test criteria in these experiments were based on different measures of user performance.

Between the years 1986-1989 I studied at the Technion of Haifa, Israel, towards a Ph.D. in Behavior and Management Sciences (which unfortunately, I was prevented from completing). During my studies I conducted several simple structured experiments on human performance.

Between the years 1994-1996 I was a freelance consultant. One of the projects was to conduct a comparative test of human performance in target detection and target identification, when using different kinds of video devices.

In 1996 I founded ***ErgoLight*** Ltd., which I continue to manage. ***ErgoLight*** develops tools for usability testing. We use these tools to conduct informal usability tests at beta sites. In addition, we continuously receive a large quantity of feedback from evaluators of our products, which we implement to increase the product usability.

My Initial Position

The scope of current methods

Common practices of usability testing, as described by Rubin (1994) are most effective when:

- The main testing concern is the product learnability, namely, identifying design flaws that prevent the users from using the product at all
- The target population of potential users can be represented effectively bya few test subjects in a laboratory setup (Nielsen, 1994).

ErgoLight Ltd., 6 Giv'on St.,
Haifa 34335, Israel
Email: info@ergolight-sw.com
Web: www.ergolight-sw.com

Tel: +972-4-826-3012
Fax: +972-4-825-8199
USA: 1-877-Use-Ergo
1-877-873-3746

Empirical studies show that during usability inspections, inspectors often detect only a small portion of the existing usability problems (Desurvire, 1994; Nielsen, 1992). This limitation is most significant for two very important types of testing:

- When main testing concern is to prevent errors when operated by experienced users. The rate of errors generated by experienced users is about half of that of novices. Because experienced users work fast, many of their errors are overlooked when using common methods of observation and video recording alone. An overview of user errors is provided by Reason (1990);
- For large-scale testing of anonymous users over the Internet. In this framework developers are required to collect usability data from many anonymous remote test subjects. A typical example is when a vendor makes a beta version of a new software productavailable for download from a Web site. In this example, the test subject typically does not communicate with the developer, except for getting technical support

Testing for errors of experienced users

UI designers find it difficult to anticipate all aspects of user errors. The costs of user errors are (e.g., http://www.cba.hawaii.edu/panko/HumanErr/ (link))

- Users waste a great deal of time recovering the data accumulated before the error
- Some errors are never detected
- User errors may result in accidents.

Errors made by experienced users are important for two types of systems:

- **Performance critical systems**, for which the marketing goal is to increase the user performance. User errors degrade user performance in two ways: by introducing wrong data into the product database and by slowing down the product operation. The degradation in operation speed of experienced user due to their own errors may be as high as 50%. An example of the costs of user error is presented by Bailey (1996, p. 192);

- **Safety critical systems**, for which the main marketing concern is to prevent accidents caused by user errors. There are indications that for each actual accident there are about 100 situations of 'almost accident' (e.g. Hannaman, 1984).

User error identification

A user error may be defined as a situation in which the user action is either not according to the user intention or out of context. Accordingly, the two main types of user errors are:

- **Psychomotoric errors** are user actions that do not match the user's intention
- **Context errors** are user actions that do not match the product's operating context.

To analyze user errors, one should record the user **actions**, the **context** and the user **intention**. Next, the log of user's actions should be matched to the record ofthe user intention and to the record of product context. For the matching, each record should have a time stamp and the recording should be synchronized. To understand the user errors, one needs to capture the user's actions and the operating context and to integrate this data with observation data. Examples of how such integration enables error identification may be found in http://www.ergolight-sw.com/www/Examples.html (link)

Operation tracking

Operation tracking is required to identify unintentional actions of experienced test subjects. As experienced test subjects work fast, they produce many such errors, which are hard to identifysolely by asking the test subject or replaying the video recording. Hilbert and Redmiles (1998) review the research and tools developed for operation tracking.

Modern operating systems support automatic capturing of the user actions. For example, MS-Windows provides hooking APIs that enables tracking GUI components (controls and menu items) provided that they are 'standard', which means that the GUI was designed using proper builders. To enable operation tracking, a developer should be able to identify those GUI components that are not standard, to add code for hooking non-standard components and to verify that the code hooks properly. To facilitate the product instrumentation, tools should be provided that enable identification and verification of both standard and non-standard controls and examples of code for hooking non-standard components.

A typical log of user actions is very long. It is impractical to review such log, not only because of its length but mainly because it does not include vital information about the user intention. Recently, new methods were proposed for using operation tracking for usability testing. According to these methods, usability is validated against a designer's model, expressed as usage expectations (Hilbert and Redmiles, 1998) or task models (Lecerof and Paterno, 1998). Hilbert and Redmiles (1998) proposed to use their method for automatic error detection over the Internet.

These methods assume that the user intention may be automatically interpreted from the log of user actions. Such assumptions should be challenged, whether the test subject is experienced or not. Novices are not able to transform their intentions to actions, because they do notyet know the operational procedures. Experienced users do know the procedures but make errors that should be identified during the testing. Theoretically, the drawback of these methods is that the user errors regarded asa deviation from a designer's model, instead of from the user's model. Suppose, for example, that the test subject unintentionally activates the wrong control. When using a designer's model, this action may be out of context and the user error might be overlooked. Interpreting the user intention from the user actions (e.g. Lecerof and Paterno, 1998) is useful only for ideal error-free operation. However, when testing for user errors, such methods become useless.

A better approach might be that the user intention should be elicited independently of the log of user actions. To identify unintentional user actions, one should synchronize the log of user actions with a log of user intentions, generated using common methods such as thinking aloud, and recorded using an observer logger. The first three examples in http://www.ergolight-sw.com/www/Examples.html (link) are of unintentional user actions identified by applying this type of integration.

Another use of operation tracking is obtaining 'operation profiles', such as statistics of control activation. In addition, for ideal error-less operation, the user tasks may be also deduced from the log of user actions, thus providing statistics of task utilization.

Context tracking

Context buttons, such as check boxes and radio buttons, may impose restrictions on the applicability of software features. For example, for regular printing, the **Print to file** check box in a **Print** dialog box should always be cleared. A common user error is trying to work in the wrong context. For example, trying to print when **Print to file** check box is marked. This error is typical to preliminary

versions of new Windows products, as developers often forget to reset the check box when the user activates the standard **Print** toolbar button.

Context tracking is required to identify situations when the test subject works in the wrong context. An example in text editing is when the user tries to edit in 'read-only' mode. As the context is often invisible, the user may not notice the wrong context. As experienced users work fast, they often overlook the context even when it is visible on screen. Often, the test observers overlook the context as well. For these reasons, context errors are hard to identify using common practices alone, such as asking the test subject or replaying video recording.

Harel (1999) presents a method of context tracking, implemented in ***ErgoLight*** tools, which support identification of context errors. MS-Windows APIs allow reading the state of context buttons. Operation tracking tools should use these APIs to facilitate context definition by capturing context buttons automatically.

Context buttons provide explicit context of standard format. Often, a feature is constrained by compound context, specified implicitly by the application code. For example, when a software flag is set in response to an internal error, to protect a document against accidental changes. Context tracking tools should provide means for hooking the state of implicit context, as well as sample code that shows how to add your own hooks.

To identify context errors, one should synchronize the log of context changes with a log of user intentions, generated using common methods such as thinking aloud, and recorded using an observer logger. The last example in http://www.ergolight-sw.com/www/Examples.html (link) is of a context error identified by applying this type of integration.

Large-scale remote testing

Remote testing is commonly considered as an extension of laboratory testing, where the test subjects are observed and monitored remotely (Castillo et al., 1997; Hartson et al., 1996). The motivation for performing this kind of remote testing is to save the expenses involved in bringing the test subjects to the lab. Chen et al. (1999) provide data showing that when using for the conferencing, the results of remote testing might be almost as good as those of real laboratory testing.

According to current practices, human, professional testers should inspect the behavior of the test subjects. In large-scale remote testing, it may happen that no tester is available to examine the mass of users that download the product from the Internet.

Can mentoring be automated? Developers are often tempted to propose that test subject data, such as reports and questionnaires might substitute for human observation. However, users' hatred of Microsoft paperclip is strong evidence that automated facilitation is not straightforward. Kaasgaard et al. (1999) present a method for large-scale remote testing wherein user intentions are collected in 'use case signatures'. User intention is obtained by prompting the users to fill in their intention in case of an operational difficulty. No human facilitation is incorporated in their method.

This method applies to situations where the main marketing concern is the product learnability. However, because it ignores the user actions, it is not applicable for testing performance critical and safety critical systems.

Software tools for testing performance critical and safety critical systems remotely should capture all the data required to understand user errors, as in a laboratory. The user actions and the operating

context should be recorded continuously. The user intention might be elicited as by Kaasgaard et al. (1999) only when the user experiences an operational difficulty. User errors may then be identified by matching the actions to the intention and to the operating context, using the same method as proposed for usability labs.

ErgoLight tools for remote testing include a feature of automated facilitation, by providing hints regarding the test subject actual actions and the actual context. However, the effectiveness of these tools has yet to be demonstrated.

The role of objective measures

Regular test reports include lists of design flaws, each assigned a subjective severity score. Typically, testers are required to explain their scores. Often, other evaluators disagree with the subjective score. Sometimes, it is difficult to convince the programmer that a particular flaw is indeed severe.

Some of these conflicts may be facilitated if the test report contained an objective measure of severity, such as number of activations of an error-prone control or estimation of the time test subjects waste because of their own errors. Tracking tools can provide objective measures of the product operation, arranged as 'operation profiles'. An example of the potential benefit of objective measures is the error-prone Caps Lock key. Everybody knows that changing the keyboard state using the Caps Lock key is error prone. It may be hypothesized that if keyboard vendors had received any substantial indications that users waste their time because of unintentional key presses, the keyboard design might have changed to fix this problem. Objective measurement of the costs of design flaws might change the industry's attitude to human factors.

The proposed topics

1. Are usability testing practices applicable to performance critical and safety critical systems? Is automation required for valid testing of these systems?

2. Should automatic logging tools be integrated with manual loggers and how?

3. Will objective measurement help in selling usability services?

4. What is required to obtain valid data in remote testing? Can a mentor be automated?

Justification

1. Many usability practitioners are not aware of the different purposes of usability testing and therefore overlook the need for new methods.

2. Many usability practitioners are not aware of the existence of new tools that support the new methods developed to provide valid data on user performance and errors.

3. Practitioners are tempted to reject valuable tools because they tend to object automating human mentoring.

References

Bailey, R.W., (1996) *Human Performance Engineering.* Prentice-Hall Inc.

Chen B., Mitsock M., Coronado J. and Salvendy G. (1999) Remote Usability Testing through the Internet. Submitted for *HCI International 1999*.

Desurvir, H.W. (1994) Faster, cheaper!! are usability inspection methods as effective as empirical testing? In Jakob Nielsen and Robert L. Mack (eds.) *Usability Inspection Methods, ch. 7, pp. 173-202*, John Wiley & Sons, Inc.

Hannaman, G.W. (1984). SHARP: Systematic Human Action Reliability Procedure, *EPRI NP-3583*, Palo Alto.

Harel, A. (1999). Automatic operation logging and usability validation. Submitted for *HCI International 9*9.

Hartson, H., Castillo, J., Kelso, J., and Neale, W. (1996) Remote evaluation: The network as an extension of the usability laboratory. *Proceedings of CHI 96. Human Factors in Computing Systems*, 228-235.

Hilbert, D.M. and Redmiles, D.F. (1998) Extracting usability information from user interface events, *Dept of Information and Computer Science, U. of California, Irvine*, Oct. 29, 1998.

Kaasgaard, K., Myhlendorph, T., Snitker, T. and Sorensen, H.E. (1999) Remote usability testing of a Web site information architecture: testing for a dollar a day. Submitted for Interact 99.

Landauer, T. (1995), *The trouble with computers*. MIT Press.

Lecerof, A. and Paterno, F. (1998) Automatic support for usability evaluation. *IEEE Transactions on Software Engineering* 24/10

Nielsen, J., (1992), Finding usability problems through heuristic evaluation. *CHI'92 Conference Proceedings*, pp. 373-380, ACM.

Nielsen, J., (1994), Guerilla HCI: using discount usability engineering to penetrate the intimidation barrier, In Bias, R.G. & Mayhew, D.J. (Eds.): *Cost-Justifying Usability*, Academic Press.

Finding usability problems through heuristic evaluation. *CHI'92 Conference Proceedings*, pp. 373-380, ACM.

Reason, J.T., (1990), *Human Error*. Cambridge, UK: Cambridge University Press.

Rubin, J., (1994), *Handbook of Usability Testing*. John Wiley & Sons, Inc.