Automatic Operation Logging and Usability Validation

Avi Harel

ErgoLight Ltd.,

6 Giv'on St. Haifa 34335, Israel,

Tel: +972 4 826 3012, Fax: +972 4 825 8199

Email: avi@ergolight-sw.com

Keywords: Usability validation, user productivity, user errors, operation logging, backtrack.

The article discusses tools that contribute to the usefulness validation of software products.

Automatic operation logging of software product is used for market research, to measure the product utility and for usability validation. It is used in two ways. First, for attaining usability measures, based on statistics of the product operation. Second, for enabling backtracking of the sequence of user actions of interest.

Statistics about usage rates are useful for product positioning, to learn which features are utilized frequently and which are not. Control level statistics of time measurements are useful for measuring the user productivity, for identifying controls that users cannot find easily, and for identifying and rectifying controls that, when activated, result in confusing system response. Task level time measurements are derived automatically from task procedures, expressed as "directed graphs" of control activations.

Backtracking is required for exploring user errors that the software product fails to forgive. More than half of the human computer interaction time is wasted on the process of error detection and correction. Backtracking is required to identify unintentional user errors that the product does not forgive and to identify mode errors. Backtracking is most useful for preventing accidents in mission critical and safety critical systems, by identifying rare instances of "almost accidents" caused by errors of the human operator.

Automatic operation logging is used both in usability testing laboratories and in remote testing. In usability labs, it is utilized best through integration with other testing tools, such as those used for video recording and thinking aloud. With loggers, such as Noldus Observer, a tester can scan a think-aloud record to identify an instance of user confusion and then synchronize this record with the other records, such as video, screen grabs, observer notes and user actions, to obtain all available aspects of the user confusion. With automatic operation logging, usability labs can increase the user productivity by attaining measurements and indications of the reasons why power users waste their time during the product operation.

In remote testing, automatic operation logging substitutes for methods that cannot be implemented outside the laboratory. In particular, it partially substitutes for two functions of a human facilitator: The first function is confusion identification, implemented by applying

“designer’s expectations”. The second function is on-line, action-sensitive assistance by providing the user with either a list of his/her recent actions or the actual system mode.

Introduction

The general motivation to this article is to help the industry develop software products that are more useful. Usefulness is the key to getting reduction in failed products, development time and costs and in training time and costs, therefore to increase sales and revenues and productivity (Rohn, 1998).

Automatic operation logging (often called 'action tracking') is a set of techniques for recording the user actions and for obtaining information useful for raising the product quality. The information gathered highlights two aspects of the product usefulness: function utility and usability (Sanders,).

Hilbert and Miller (1998b) provided an overview of research and tools developed for automatic operation tracking. This article presents an overview of the usage of automatic operation logging for increasing the product usefulness. The main focus of this article is in using these techniques for usability validation.

Historically, software development involved three stages (e.g. DeMarco, 1979):

1. System analysis, a 'pencil and paper' procedure of system task breakdown that ends up in a product specification

2. Product design, a procedure of transforming the product specification into implementation instructions

3. Implementations, a procedure of transforming the product design into software code and for making the code work according to the design instructions

The three-stage approach was a great improvement over the chaos in earlier software development. Still, more the 30% of software development are cancelled before completion, primarily because of inadequate user design input (Standish Group, 1995). In addition, 63% of all software projects overran their estimates, with the top four reasons all related to usability (Lederer and Prassad, 1992).

Usability Engineering

To increase the rate of product success, a new discipline has emerged, namely, usability engineering (e.g., Rohn, 1998). In the early days of usability engineering there were the design rules (Shneiderman, 1980) that evolved later to style guides and standards, such as ISO 9126, ISO 13407 and ISO 9241 parts 10-17. All these techniques result in consistent design that fits mostly the designer's needs. However, the limitation of these techniques is that they assume that the user is educated to work according to the standards and guides.

The assumption that users can be educated should be questioned. In a dynamic world of frequent technical changes, users of many software products do not have any chance to experience software products the follow the standards. Paradoxically, style guides may become obsolete even before software developers can benefit from adopting them. In many practical cases, the main contribution of style guides is rather in obtaining within product consistency.

An alternative to constraining the design to standards, is to adapt it to the users. The human centered approach is well know as the User Centered Design (UCD) (e.g. Norman, 1986).

Besides that common acceptance of this approach, there is the acceptance of its limitation. Even the best UI designer cannot represent the user faithfully. The user always surprises.

Usability Validation

Usability validation is the process of verification that the user may be able to work according to the product design. The industry most common practice to get the user acceptance is by asking the users.

Two forms of user reports are widely used. The more popular is the free form user initiated problem report. This method is being implemented nowadays also for reporting over the internet (e.g. Hilbert and Miller, 1998a). The other form is the structured form, by questionnaires composed by usability professionals to provide various usability scales.

User feedback is most effective in the early stages such as at the prototyping and beta stages, mainly to get a rough idea about the product utility and the user's satisfaction (e.g., Bevan). However, the information obtained by the user's own reports is quite limited and is not sufficient for usability validation, because users report about only a small portion of the difficulties that they encounter. Typically, users do not report about problems that they do not identify or that they do not consider as design flaws.

Usability problems that users do not identify

Typically, users identify only a small subset of the problems they encounter. Many usability problems are not identified even if the users are carefully selected to represent the real users. Often, users do not identify deficiencies related to actions that are incompatible to their intention. Some examples follow:

Users do not identify a usability deficiency if they cannot repeat the sequence of actions that ended up in the confusing situation. For example, when the user unintentionally activates a menu item that changes data, and no indication of that action is displayed on screen.

Users do not identify usability deficiencies resulting from mode discrepancy. A mode error occurs when a user tries to execute a function that is not relevant to the current application mode. For example, when the application is in "Read Only" mode and the user tries to modify data.

Usability problems that users would not report

In many other situations, users do identify usability deficiencies but avoid reporting on them for various reasons:

First, users avoid reporting on confusing situations that they have identified, when they are not sure about the reasons or when they cannot easily repeat them.

Second, many severe usability problems are the result of poor design. When the user fails to follow the designer's logic, both the designer and the user tend to consider it the user's fault, rather than a design deficiency. Most users will not report on an operational difficulty, unless they are convinced that the failure was not their own fault. Examples:

Users hesitate to report problems in understanding the application concepts and terminology as well as problems in knowing the procedures, which they need to follow in order to perform a task. Unless the information is not found, users would not report on difficulties they

experience in accessing information in the user documentation, the training program or the on-line help. This type of usability problem is typical of complex systems, supplemented by a large body of user documentation that the user has no chance to read thoroughly, much less to remember.

Users would not report on instances of unexpected system response to inadvertent control activation. Inadvertent control activation occurs after a control has been used frequently. Hence, this type of confusion is typical of experienced users. Inadvertent control activation often occurs in cases of mode discrepancy, when the user wrongly "feels" that s/he is in the proper mode for operating that control. Even when the result of inadvertent operation is disastrous, users usually consider it to be their own fault and avoid reporting it as a design problem.

Third, users would not report on a deficiency that they consider as minor. For example, users typically would not report on problems that they have with the Insert key, which are seemingly negligible. No one has yet measured the overall costs of user wasted time due to data deletion while unintentionally editing in Overwrite mode. It is likely that overall, these costs sum up to millions of dollars.

Fourth, whenever the user feels that s/he has a task to complete, s/he postpones making the report until after the task is completed. Later, the user often cannot remember what the situation or the sequence of operation was. Goal oriented users are cooperative in reporting only on those problems which prevent them from completing their tasks. These users are not likely to report on a problem if they find a way to work around it.

For these reasons, user oriented software developers do not rely on the users' own problem reporting. Rather, they test their users at work by professional usability testers.

Usability Testing

Usability testing is the process of identification of the reasons for users' failure to accomplish their tasks within reasonable time limits. The identification of an instance of user difficulty typically has the following steps:

1. Identify the situations when the user experiences operational difficulties
2. Identify the user's intention that s/he failed to accomplish
3. Analyze the reasons for the user's difficulty
4. Facilitate the users, allowing them to resume operation in spite of the difficulties that they encounter.

The best way to test usability is in usability labs, where various testing techniques may be applied. An effective means to identify situations of user difficulties in a lab is by assigning a test monitor or an observer. Video recording is helpful for reviewing these instances off-line. Next, identification of the user's intention is made easy in labs by asking the users to think aloud. Analysis of the reasons for the user's difficulties typically relies on the test monitor, who should learn the details of the product operation before the testing begins. In addition, the test monitor typically facilitates the users by either directing them to the right actions or by applying work around procedures. Rubin (1994) provides a comprehensive overview of applying various usability lab techniques.

‘Traditional’ usability testing, as described above, is probably the best when the main marketing concern is the product learnability. However, It is not sufficient for two other major categories of software products:

- Performance critical systems, where the main concern is to increase the user performance
- Safety critical systems, where the main concern is to prevent accidents cause by user errors.

For products of these categories, the main testing concern is to understand the user errors rather than to increase the product learnability. Video recording, thinking aloud and observations are not sufficient to capture the sequence of user actions that led to the situation of user confusion.

Test for User Errors

The tools described here are aimed at contributing to this need by tesin reducing product failure

Scope: software products, Windows applications.

Paterno

Types of user problems

- Barrier to getting started
- Partial procedure knowledge
- Partial system tolerance to the user errors

Questionnaires are the cheapest way to learn user difficulties resulting from partial procedure knowledge. However, when filling a questionnaire, it is difficult for the user to explain why s/he could not get started or to report exactly what errors s/he made that the system did not tolerate.

Usability labs are the most effective means to identify barriers to getting started. The test monitor (facilitator) and the observer are in charge of keeping track of the user actions and of eliciting the user intention along the execution of a test scenario.

However, traditional labs do not have the means to understand all user errors.

User errors:

- Psychomotoric errors (unintentional action)
- Mode errors (Caps Lock, Ins, Print to file, Read only)

Two types of psychomotoric errors:

- Inadvertent user action, actions that the user is aware of (expressed by an ‘oops’ utter)

- Errors of which the user is not aware, such as slip, missing key or button press, double press or double click etc. (reference)

Traditional testing in usability labs is good in identifying the first type of user actions, because the user knows that s/he did the wrong action and s/he can report what action s/he intended to take in the first place.

However, it is not most effective in identifying other unintentional actions, of which the user is not aware. To identify the user action, we cannot rely on the user report or on observations, even when they are video recorded. Instead, we need to capture the actual user actions, and to match this record with an indication of the user intention at the time of the error.

In addition, tradition usability testing is not most effective in identifying mode errors. It is very difficult for a test monitor or for an observer to keep track of the change of system modes that are the consequence of the user actions.

Understanding the user errors is important for two purposes:

- Raising the user performance
- Preventing accidents cause by human errors.

The role of operation logging in standard setting.

Not enough research is done so far. For example, the Ins key.

The Caps Lock key. The Print to file check box.

Remote testing

To identify

The role of operation logging tools in usability engineering

Tools are used to support usability engineering during the design phase, the testing phase and the deployment phase.

Theoretically, design tools may incorporate human factors, formulated as "design rules". Practically, however, most design rules apply only for certain situations. A simple example is the contradiction between the rule of "consistent appearance" of application dialog boxes and the rule of "distinctiveness". Until we know how to formalize the applicability of design rules, User Interface (UI) designers will have to rely on usability specialists.

User behavior is often unpredictable, even for usability specialists. Many usability specialists prefer to ask the users for their preferences and to test the users' performance, rather than rely on even their own designs. Still, empirical studies show that during usability inspections, inspectors often detect only a small portion of the existing usability problems (Desurvire, 1994; Jefferies et al., 1991; Nielsen, 1992). Operation logging tools are used at the testing

phase, to collect data from the users, in order to help usability specialists to understand the user's behavior. The objectives of using these tools are to:

- 0 Capture user failure modes that are difficult to capture manually
- ¬ Collect data from remote users
- ∟ Provide objective measures for comparing design alternatives
- ⊏ Provide measures of the costs of design flaws.

The first objective in the list above is critical for enhancing system reliability and is particularly important to developers of mission critical and safety critical systems.

Measuring user productivity

Landauer (1995) indicates that the average software program has 40 design flaws that impair employees' ability to use it. The cost in lost productivity is up to 720%. Recent studies show that subjective evaluation performed by usability specialists is not reliable (Jacobsen, 1998). The advantage of objective usability measures was demonstrated in several comparative studies (e.g., Molich et al., 1998).

Usability measurements are presented in two types of "user operation profiles":

- "User activity profiles" enable developers to identify UI components that requre too much time to find and activate
- "User failure profiles" enable developers to identify specific user failure modes.

User activity profiles

A user activity profile consists of statistics about the usage of GUI components (controls, menu items, etc.) such as the number of control activations and the average operation time (e.g., Harel, 1998).

User activity profiles enable software developers to identify GUI components with exceptional properties, such as:

□ㄹGUI components that are used often

□ㅁGUI components that are rarely used

□ㅂGUI components that are easy to operate (short operation time)

□ㅅGUI components that are difficult to operate (long operation time)

Average operation time is easy to measure but difficult to interpret. For example, suppose that the user activated control A and then control B, that the elapsed time between activating controls A and B is 20 seconds and that a usability specialists considers this value too high. In order to reduce this value, we need to identify which part of the elapsed time users spent evaluating the software response to activating control A, and which part of the elapsed time they spent searching for control B. To enable decision making, the user operation profile should break the operation time interval into significant parts, such as system response time, evaluation time, time spent doing non-UI related tasks and search time.

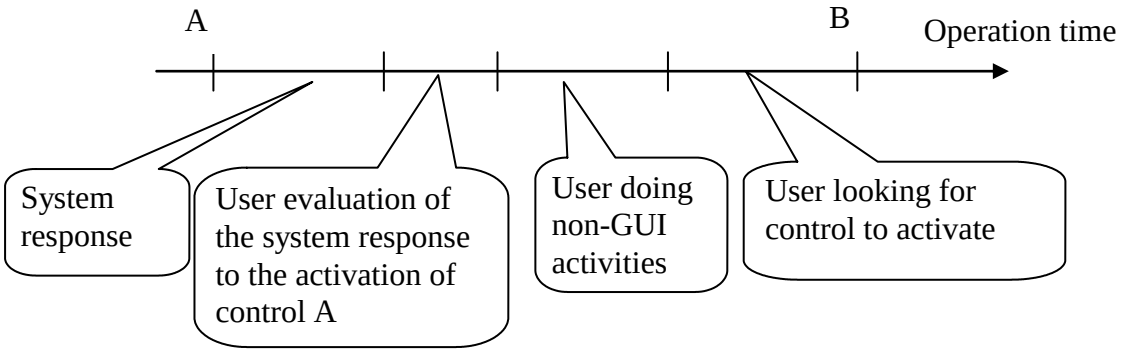

Fig. 1 –Elements of user operation profiles.

Understanding the user's confusion

Just by looking at the record of user actions, it is often impossible to decide whether a particular situation indicates an instance of user confusion. Examples:

□ㅇThe user works in the wrong mode

□ㅈThe user activated a particular GUI component unintentionally or inadvertently

□ㅊThe user hesitates because s/he cannot recall an operational procedure.

Preventing accidents caused by human errors

"User failure profiles" enable developers to identify particular user failure modes. Such information is useful for preventing system errors resulting from user errors.

The percentage of accidents due to human errors is in the range of 15%-90% of the cases of system failures (e.g. Hannaman, 1984). User failure profies are useful for identifying instances of "almost accidents", which is critical for preventing real accidents.

Integrating automatic operation logging in laboratory testing

Automatic operation logging is utilized best through integration with other testing tools available in usability labs, such as those used for video recording and thinking aloud. An example of the benefit of integrating several different tools can be seen in the scenario below, which describes identification of an unintentional user action that result in a confusing system response.

Logging the user operation

In this example, we would expect the following sequence of logging activities to happen:

1. A user is working normally, except that s/he also specifies his/her intention while working, according to the 'thinking aloud' method
2. The user unintentionally activates a certain GUI component.
3. Because the user is instructed to think aloud, s/he utters an 'oops'
4. An observer adds a manual log event that s/he had called 'Surprise', to a log of observation events, using an event logger such as Noldus Observer

Analyzing instances of user confusion

In the evaluation phase, an evaluator may use the observation logger to scan the observation log. For each of the events, s/he will typically perform the following sequence of operation:

1. Set the VCR to a section that is associated with the 'Surprise' event. Thanks to the time stamp, the synchronization is done simply by a button click.
2. Replay the VCR to learn the user's intention and to verify that the user was indeed surprised
3. Backtrack the user actions to find out what the last user action was before s/he was surprised by the system response
4. View the screen recorded using a scan converter, to verify that the screen that the user was viewing matched the user's actual action, rather than his/her intention
5. Classify the event as an event of type 'Unintentional action'
6. Set the VCR at the point where the user activated the wrong GUI component and marks a 'Start' attribute with the VCR time code
7. Set the VCR at the point where the user resumed normal operation after recovering from his/her error and marks an 'End' attribute with the VCR time code.

Measuring the costs of design errors

Each instance of user action has certain attributes that enable measuring the costs of user errors. The costs of a single instance of user confusion are calculated by subtracting the Start time code from the End time code. The costs of design errors are measured by the sum of the costs of all instances of user confusion relating to the same class. For example, the costs of designing an error prone control is calculated as the sum of the user's time wasted on all unintentional activations of this control.

Using Automatic Operation Logging for Remote Testing

'Traditional' remote testing consists mainly of extending the means available in usability labs to remote users (e.g. Castillo et al.). An example of appying this approach in practice was demonstrated by Tondre (1998).

The limitation of this approach is that certain testing tools that are most useful in laboratory testing are not available for remote testing. For example, in many real beta-testing situations it is often unlikely that users might feel comfortable when they are video recorded while doing their job or if they are asked to think aloud in an open space working environment.

An operation logger is most useful to work around these limitations, since it can substitute for some of the functions of the tools and methods used in usability laboratories, as is demonstrated below (Harel, 1998):

1. Situations of user confusion are identified by indicators derived from the user actions, such as deviations from developer's expectations (Hilbert and Miller, 1998) or constrains (Lecerof and Paterno, 1998) or certain controls, marked as Usability Problem Indicators (UPI)

2. The user's intention is obtained by prompting the user to specify his/her intention either by selection from a task breakdown or in free text, using his/her own words

3. Operational dead locks are resolved by presenting to the user action sensitive information, such as recent control activation and the current state of parameters that affect the user's task

Using Automatic Operation Logging for Remote Testing

Overview

In remote testing the procedure is different, because certain testing tools that are most useful in laboratory testing are not available for remote testing. For example, in many real beta-testing situations it is often unlikely that users might feel comfortable when they are video recorded while doing their job or if they are asked to think aloud in an open space working environment.

An operation logger is most useful to work around these limitations, since it can substitute for some of the functions of the tools and methods used in usability laboratories, as is demonstrated below:

4. Situations of user confusion are identified by indicators derived from the user actions, such as deviations from operational procedures (expectations or constrains) or certain controls, marked as Usability Problem Indicators (UPI)

5. The user's intention is obtained by prompting the user to specify his/her intention either by selection from a task breakdown or in free text, using his/her own words

6. Operational dead locks are resolved by presenting to the user action sensitive information, such as recent control activation and the current state of parameters that affect the user's task

Limitations of remote testing

The example of the user activating a control unintentionally will be used to demonstrate this limitation.

1. The user does not think aloud, therefore his/her intention remains unknown

2. The user does not express a surprise utter, therefore the tester does not know that the user experiences an operational difficult

3. An observer does not log events, therefore there is no record of instances of user confusion

4. A video camera does not capture the time codes of starting and ending an instance of user confusion

5. No facilitator is around to help the user in case of operational dead-ends.

An operation logger is most useful to work around these limitations, since it can substitute some of the functions of the tools and methods used in usability laboratories, as is demonstrated below:

Identifying user confusion

With the absence of observers and of video and voice recording, one might consider asking the users to report on these instances of user confusion by their own initiative. The problem with this approach is that users are rarely willing to report on a problem before exploring the situation, in order to exclude the possibility that they will look stupid (Harel, 1998). Since such exploration may require a lot of time (dependent on the user personality) users typically avoid reporting instances of their own confusion.

An operation logger is useful for identifying potential instances of user confusion. For example, Hilbert and Miller (1998) proposed identification of deviations from rules that express the developer's "usage expectations" as a way for identifying potential user confusion. A similar approach is of Cerenof and Paterno (1998) who proposed to identify violation of task constrains as indicators of user confusion.

Another approach, implemented in ErgoLight Usability Validation Suite (EUVS) is to identify potential instances of user confusion by GUI components that activate certain features, such as Undo, Cancel and Help (Harel, 1998). The developer marks these components and Usability Problem Indicators (UPI). The operation logger identifies the activation of UPIs as an indication of potential user confusion.

The advantage of the control level approach over the task/rule level approach is that it is easy for the developers to define them. The limitation of this approach is that it does not cover compound rules, such as mode transition (Hilbert).

Confirming the user confusion

When testing in a usability lab, a facilitator can make sure that the user's surprise indeed indicates an instance of user confusion. The situation is even worse when the indication of potential user confusion is based on the developer's expectations. Developers cannot

anticipate the way the user will behave during the product operation. Often, a user may intentionally perform a sequence of actions that violates the developer's rules or expectations.

When an instance of potential confusion is encountered, the operation logger prompts the user to confirm that s/he indeed experiences an operational difficulty.

Controlling the rate of false alarms

A human facilitator is typically most sensitive to the user's need for help. A computerized facilitator will typically lack this capability. The result is that a computerized facilitator produces high rates of false alarms, which often irritate the user. An example of how not to implement a computerized facilitator is the well-known paperclip of MS-Office. To minimize the user irritation, the operation logger is integrated with other tools that measure the rate of false alarms generated by each of the rules. The developer can review the rate of false alarms generated and to disable these UPIs that generate high rates of false alarms.

Eliciting the user's intention

In a laboratory test, the user intention is elicited by playing back the VCR and hearing the user thinking aloud. The substitute to this way of eliciting the user's intention is by prompting the user to specify their intentions. Users can specify their intentions either by typing it to a memo control or by selection from a task breakdown, prepared beforehand by the developer.

Resolving operational dead-locks

Typical laboratory setup is such that users who started a test session will feel obliged to end it. This assumption is not valid for the case of remote testing. Of particular care is the case when users get to operational dead-ends. In remote testing, users will tend to abort the testing at all.

To get the user cooperation, the operation logger is integrated with facilitation means.

- The display of recent user actions enables the user to identify instances of unintentional control activation

- The display of violated rules/ developer's expectation, such as working in the wrong system mode, provides hints to the user how to change the system setup or the way s/he works.

Test Administration

In a usability lab, it is a test administrator who installs the tested product, who defines scenario of user tasks and instructs the user about the testing procedures. In remote testing, most of these activities should be automated and well documented. The operation logger is installed automatically. The user is able to do his/her real work, rather by scenario. The operation instructions and the technical support facilities are well linked to the logger operation.

Conclusion

Automatic operation logging enables usability testers to identify design deficiencies that are difficult to identify otherwise. The main benefit of automatic operation logging is in the capability to capture user errors, such as unintentional or inadvertent actions or the user working in the wrong system mode.

These advantages are manifested when the testing goal is to increase the user productivity, because current testing procedures are aimed at novices, rather than power users. In addition, automatic operation logging attains for substituting functions implemented using 'traditional' laboratory tools by features derived from the log of user actions.

References:

Castillo, J.C., Hartson, H.R. and Hix, D., 1998, Remote usability evaluation: can users report their own critical incidents?; *CHI 98 Summary, pp 253-254*, ACM Press.

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.

DeMarco, T., 1979, System Analysis and System Design, Yourdon Press.

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

Harel, A. (1998). Usability Testing. *Proceedings of STAR West 1998*.

Hilbert, D.M. and Redmiles, D.F., 1998a, An approach to large-scale collection of application usage data over the internet. In *Proceedings of International Conference on Software Engineering* 1998.

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

Jacobsen, N.E., 1998, The evaluator effect in usability tests. *CHI 98 Summary, pp 255-256*, ACM Press.

Jeffries, R., Miller, J.R., Wharton, C. and Uyeda, K.M., 1991, User interface evaluation in the real world; a comparison of four methods. *CHI'91 Conference Proceedings, pp 261-266*, ACM, 1991.

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

Lecerof, A. and Paterno F., 1998, Automatic support for usability evaluation, *IEE Transactions on Software Engineering, Vol. 24, No. 10.*

Lederer, A.L. and Prasad, J., 1992. Nine management guidelines for better cost estimating. *Communications of the ACM 35(2) 51-59*.

Molich, R., Bevan, N., Curson, I., Butler, S., Kndlund, E., Miller, D. and Kirakowski, J., 1998, Comparative evaluation of usability tests. *UPA 98 Proceedings, pp 189-200*.

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

Norman, D. 1986, User Centered System Design.

Rohn, J.A., 1998, Making the Case for Usability Engineering to Management: Total Costs of Ownership and Cost -Benefit Analysis. In NIST Usability Engineering 3 Proceedings, June 22.

Shneiderman, B. 1980, Software Psychology.

Standish Group, 1995, The Kompass Report, Forbes.

Tondre, A., 1998, Remote usability testing at Sun Microsystems. *UPA 98 Proceedings, pp 279-280*.