Automation in usability validation

A. Harel
ErgoLight Usability Software Ltd.,
6 Giv'on St. Haifa 34335, Israel,
Tel: +972 4 826 3012, Fax: +972 4 825 8199
Email: avi@ergolight.co.il

Abstract

In this article the term "usability" refers to friendliness, ease of use, ease of learning, user productivity and the system resistance to user errors. Usability validation should rely on the examination of real end users when they do their real work, rather than on GUI specification, standards or style guides. Available tools for Software Quality Assurance (SQA) do not address the issue of resolving problems that end users encounter during operation.

Automation in usability testing is required to satisfy three main needs:

- To reduce the costs of testing procedures;
- To shorten testing cycles;
- To identify severe usability problems that are difficult to detect manually.

Software tools used in usability testing should be integrated with human factors methodologies and should support full cycle application development.

ErgoLight integrates methodologies of Human Factors Engineering in software tools. ***ErgoLight*** allows QA experts to identify problems in the user interface design, in the user education and training programs, in the user documentation and in the on-line help system. The problems ***ErgoLight*** identifies are of user conceptual difficulties, using the wrong terminology, orientation problems caused by mode discrepancy and wrong response to user errors. For all problems identified, ***ErgoLight*** provides backtrack to the record of user actions. For repeated difficulties, ***ErgoLight*** provides statistics of the user wasted time, used as a measure of costs of usability deficiencies.

Keywords

usability, testing, validation, human factors, quality, friendliness, ease of use, user productivity, ease of learning, forgiveness, user error, user difficulty, terminology, orientation, mode error, procedure knowledge, education, training, on-line help, user information, user documentation, automation, capture/record, backtrack, error costs, GUI, QA, SQA.

1 INTRODUCTION

1.1 The need to test usability

Software usability

As software companies face ever-increasing competitive pressures throughout the process of development of new applications, they need to define, understand and focus on their core competencies. A main contributor to a company's success is the quality of its product lines. The term Software quality refers to various perspectives, including reliability and usability. In this paper the term "usability" refers to friendliness, ease of use, ease of learning, forgiveness and user productivity.

Usability quality

Software companies prefer to prevent bugs rather than to fix them after they are already introduced to the market. This is because the costs of fixing problems after the product is already installed at the customer site are very high, and the costs of losing the market are immeasurable. This preference applies to all software design, and particularly user interface design.

Windows based applications use Graphical User Interfaces (GUI) that allow application developers to deliver seemingly friendly software products. Apparently, end users often experience difficulties in using software products equipped with these GUIs. Quality Assurance (QA) professionals are challenged by the need to identify design deficiencies that result in reduced usability.

Industry standards are not enough

Traditionally, GUI design relies mainly on designers' style guides. The common rules for user interface design are based on the means provided by popular development tools and on industry standards. Available style guides help designers to deliver applications consistent with other applications. The tools and the standards facilitate becoming acquainted with new software products. An overview of available standards and style guides is presented in Daly-Jones et al. (1997).

Applications that follow style guides are convenient for multi-application users, such as programmers, because the similarity facilitates the learning process. However, they are not necessarily easy to use by many single-application users, because style guides limit usability optimisation. Apparently, the tools and standards do not contribute to the main usability issue, namely, to serve the software features that correspond to a user task, according to the user's expectations. In fact, the most important usability issues are not even within the scope of industry standards.

Usability Barriers

Four major usability barriers are:

- Communication barrier: Designers and users use different terminology. For example, when a novice user of a word processing application looks for how to 'print a letter' and fails to use the corresponding terms used in word processing, such as 'edit document'.

- Interpretation barrier: Designers and users have different knowledge base. For example, when a novice user of a word processing application looks for the procedure for moving a block of text but fails to know the concept of clipboard, how to select text and the cut and paste functions.

- Forgiveness barrier: Designers do not anticipate user errors. For example, when a user unintentionally activates the wrong menu item.

- Orientation barrier: Users fail to identify the operation context. For example, when the user tries to edit when in 'read only' mode.

Usability design vs. usability testing

GUI designers often fail to predict the difficulties that the end users will experience during operation. To enhance usability, more and more companies have begun to consult experts in ergonomics (human factors) as early as at the specification stage. However, even professional user interface designers often overlook many of the deficiencies presented in a software product.

Although the usability of a software product is determined by its design, the only means to verify that the product is indeed usable is by testing. Since user difficulties are hard to predict, GUI validation should rely neither on style guides nor on specification. To identify these problems, usability should be tested for the user's expectation and productivity, by observing the real end user during the interaction with the software application. An overview of methods for designing the user interface is presented in Shneiderman (1992). An overview of methods for evaluating user interfaces is presented in Nielsen (1993).

1.2 The need for tools to test usability

Usability validation in alpha cycles

Formal usability testing is not very common in the industry. In many organisations, the quality of user interfaces is tested within the overall software testing procedures. In these organisations, usability testing relies on observations of experts in software quality assurance during alpha testing and on reports of the end users during Beta testing.

Typically, the persons who test software in alpha testing are of technical background. Traditional testing, by persons of technical background, does not reveal many usability problems, especially those that characterise the first use of a new product. Eventually, alpha testers attribute the reasons for operational difficulties to the end user, rather than to the software design.

Usability validation in usability labs

To overcome these limitations, large software organisations created special departments, Usability labs, operated by experts in Human Factors engineering. In these labs, professionals observe real users, while they use prototypes for completing real tasks. The analysis of usability problems involves various techniques, such as video recording and back tracking, questionnaires and "think aloud". Many usability bugs are found using these means.

Usability labs typically take the approach that procedures of usability QA are valid only if the testing is with representatives of the real end user. In these labs real end users are

observed by human factors experts, their actions and behaviour are video recorded and various techniques are used for obtaining a significant body of data and for deriving conclusions from these data.

The effectiveness of Usability labs

The operation of the techniques used in usablabs is expensive and time consuming. Many developers of high functionality Windows applications, including small and medium software houses, are forced to compromise usability issues, because they cannot pay for the services of usability labs. Because time and budget resources are always limited, usability labs need to focus on the main problems and to ignore others. Consequently, many software products that were examined in usability labs, even including main line products of the leaders in the software industry, suffer from severe usability problems.

Another limitation of the current procedures of usability testing is that they typically do not provide commonly acceptable measures for the costs of a usability problem. For example, it is difficult for these procedures to obtain statistics on the time wasted because of the user's unintentional press of the Caps Lock or the Alt key.

Usability validation in beta cycles

Better still, usability can be validated at the user's site, such as in Beta sites. End users in beta sites provide important information about major operational deficiencies. However, end users identify only a small subset of the problems they encounter. Many usability problems are not identified in beta sites, even if the users are carefully selected to represent the real end users.

Usability problems that end-users do not identify

Typically, end users do not identify deficiencies related to actions that are incompatible to their intention. Examples:

- End 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.

- End 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 end-users do not report

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

First, end 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 end 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. Examples:

- Typically, users hesitate to report problems in understanding the application concepts and terminology and on problems of knowing the procedures, which they need to follow in order to perform a task. This type of usability problems is typical to complex systems, supplemented by lots of user documentation that the end user has no chance to read thoroughly, much less to remember

- Typically, end users do 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 to 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, end users usually consider it to be their own fault and avoid reporting it as a design problem.

Third, end users do not report on a deficiency that they consider being minor. For example, end users typically do not report on problems that they have with the Ins key, which are seemingly negligible. Nobody has ever measured the overall costs of user time waste because of deletion of data when unintentionally editing in Overwrite mode. It is likely that overall, these costs sum up to millions of dollars.

Fourth, whenever the end 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 co-operative 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.

2 USABILITY TESTING TOOLS

Objectives of usability testing tools

Automating usability testing is required to satisfy four main needs:

- The need to reduce the costs of testing procedures
- The need to shorten testing cycles
- The need to identify severe usability problems that are difficult to detect manually
- The need to evaluate the costs of design deficiencies.

SQA tools

Many available Software Quality Assurance (SQA) tools collect data on the application GUI, such as the controls used in the GUI and the user's actions during a testing session. Typically, members of testing teams use these data in alpha cycles.

The main limitation of such tools for usability validation is in their goal, which is to compare implementation to specification, assuming that the specification is an adequate reference for usability validation. These tools do not target important validation features, such as the user's difficulties in understanding the application concepts and terminology, poor system response to user errors and difficulties in perceiving the GUI mode.

Tools available at usability labs

Another approach, commonly adopted by usability labs, is to collect data on the user's behaviour during the interaction, by video recording, by capturing control activation and by human observation. Many usability professionals have tried before to use SQA tools for analysis of the user's behaviour, by collecting data at test time using capture/record techniques. Typically, these professionals found it most difficult to identify the instances of user confusion in the huge, meaningless sequence of user's actions, even when those actions were synchronised with video recording and with observer's notes of the user's operation.

In order to learn about a particular user's difficulty, tools for automated usability validation should identify situations of user confusion and should assess the user's intention at these events.

3 ERGOLIGHT - AN IMPLEMENTATION EXAMPLE

ErgoLight is a set of solutions, used to enhance the usability of Windows applications, namely, to make them more friendly, efficient, reliable, easy to learn and easy to use. Usability enhancement is obtained with an innovative approach, by on-line identification of the user's difficulties, on-line assistance and analysis and by providing to the development team detailed and statistical information, regarding a variety of usability deficiencies. Based on this information ***ErgoLight*** provides reports for changes required in the GUI specification, in the GUI design, in the user education/training programs, in the user documentation and in the on-line Help information.

3.1 ErgoLight technology

Recording the user actions

ErgoLight records the user actions when operating a Windows application, allowing back tracking, as well as history based automated analysis of the user confusion.

Identifying instances of user confusion

ErgoLight provides both manual and automated identification of instances of user confusion. Automated identification of user confusion is based on confusion identifiers, such as user response delay, activation of a Help feature, or invoking a Cancel or Undo feature.

Interpreting the user intention

ErgoLight provides both manual and automated interpretation of user intention. The user intention is interpreted in terms of user task breakdown, which is entered to the ErgoLight™ database at the design phase.

Identifying usability problems

ErgoLight compares the user's recorded actions to the user's intention, analyses the match between them, and identifies usability problems of three types:

Problems typical for users new to the application

Tasks that the user could not accomplish using the software application. For example, the user of a word processor wants to move text but fails to use the Drag and Drop method, or to understand the clipboard concept and how to use it.

Problems typical for experienced users

Sensitivity of the user interface to psychomotoric user errors. For example, the unexpected show up of a dialog box on screen, because of the mouse slip or because of using the wrong shortcut key combination.

Problems that confuse occasional users

Confusion due to customisable features, such as by set-up parameters. For example, when the user fails to print because s/he is not aware of the "print to file" check box.

On line recovery

ErgoLight provides on-line assistance to the end user, based on the results of the problem analysis. This information provided allows the user to resolve certain usability problems, such as mode errors, and to learn how to avoid error prone operations, such as using the wrong key combinations.

Identifying deficiencies in the user information

ErgoLight classifies the records of user confusion by available sources of user information. For example, a designer can extract a report containing all user problems associated with the user's guide.

3.2 *ErgoLight* tools

ErgoLight™ /Spec

A Computer Aided Software Engineering (CASE) tool used at the specification phase, for describing the user interface in terms of user tasks. Using this tools, the product GUI designers enter the User Model, namely, the product functions expressed in terms of the user tasks, broken down to goals, methods and operational procedures

ErgoLight™ /Prototype

A tool for the GUI designers, allowing them to link the specified procedures to the GUI components of the actual prototype, specify the system modes and the indicators of user difficulty in operating the tested product... all just by point and click.

ErgoLight™ /TestSetup

A tool for the test manager, to specify the user classes, to customise the dialog with the end users, to enter a scenario, which the sequence of user tasks in a testing session and to enter a list of standard events.

ErgoLight™ /Operate

An add-on tool for the end user, to enhance the GUI and to collect data on usability deficiencies

ErgoLight™ /Eval

An evaluation tool, used to analyse usability problems, to provide statistical, objective measures of the usability problem identified and to recommend how to change the GUI so that it will be more user friendly

ErgoLight™ /Review

A tool for the members of the development team to review the evaluator's comments and recommendations for changes in the GUI, in the user education, in the user information and the User Model

ErgoLight™ /Desk

A tool for Customer Support centres, used to provide Help Desk information on usability issues

3.3 Full Development Cycle Support

ErgoLight is used throughout the life cycle of application development:

Specification

At the specification phase, ***ErgoLight*** is used to specify the user task breakdown and the modes that affect the way the application responds to the user actions

Prototype

At the prototype phase, ***ErgoLight*** is used to link user tasks and the application modes to the actual GUI components

Beta testing

At the beta-testing phase, ***ErgoLight*** is used to collect data on usability problems

Evaluation

At the evaluation phase, ***ErgoLight*** is used to analyse the data and also to obtain reports on usability problems, on your recommendations for design changes, and on your instructions for the Help Desk personnel

Deployment

At the deployment phase, ***ErgoLight*** is used to provide recovery information for the end users and to provide Help Desk information for support centres.

4 OPERATIONAL PROCEDURE

ErgoLight records the user actions when operating the Windows application, identifies the points of user confusion, and prompts the user to store records of his/her intention at these points. Based on the correlation between the user intention and the actual actions, ***ErgoLight*** extracts information, which is useful for understanding the circumstances of user confusion. The information thus extracted includes reports on problems in the user documentation, problems in the Help system, statistics on user confusion and the user's actions associated to a particular problem, used for backtrack analysis of these circumstances.

ErgoLight test reports include many details of user actions and intentions that are otherwise unavailable. The measures of costs of usability problems used by ***ErgoLight*** are objective and quantitative, in terms of the user's time wasted. Using ***ErgoLight,*** many of the routine details involved in usability problem analysis are obtained automatically. ***ErgoLight*** allows experts in usability quality assurance to focus adding the real value of their expertise.

Using ***ErgoLight***, usability testing is a three-stage procedure:

Stage 1. Definition of the User Interface

Stage 1 is typically conducted at the designer site, by user interface designers, such as system analysts or, preferably, by human factors engineers. At this stage, ***ErgoLight*** is used as a typical CASE tool. The definition stage is typically conducted in 2 steps:

1. User task specification is conducted by system analysts or, preferably, by human factors engineers

2. User operation and problem indicator definition is conducted by the GUI designer, who knows the implementation.

Stage 2: Data Collection

Stage 2 is typically conducted at the user site, such as at Beta sites. At this stage, ***ErgoLight*** is used as a test controller. The data collection stage is typically conducted in 2 steps:

3. Specification of a test set-up, is conducted by the test manager

4. Monitoring of the user's operation, is conducted by the end user.

Stage 3: Concluding

Stage 3 is typically conducted at the designer site, using information gathered at the user site in stage 2. The conclusion stage is typically conducted in 2 steps:

5. Evaluation, is typically conducted by user human factors professionals

6. Implementation, is conducted mainly by the GUI designer.

5 CONCLUSION

Comprehensive detection

Problems of sensitivity of the user interface to user errors are hard to detect, because users typically prefer to work around their errors rather than to report on them. ***ErgoLight*** automatically detects and reports on a wide range of problems that are typically ignored in traditional software testing.

ErgoLight identifies situations of user confusion and initiates a reporting session even before the user is aware of the fact that a usability problem has occurred, or that it should be reported to the application designers.

Problems in procedure knowledge

ErgoLight identifies situations in which the user is not sure about the procedure that should be used to accomplish a task. ***ErgoLight*** identifies such situation by problem indicators, such as the user response delay or the activation of a Help feature.

Sensitivity to unintentional actions

ErgoLight identifies problems that users are likely to ignore, including those relating to the sensitivity of the user interface to user errors. Consider, for example, what happens in case of a key slip. The result may be the activation of an unexpected dialog box, unintentional mode change or, even worse, activation of an undesired feature, such as erasing data. ***ErgoLight*** identifies these instances of user difficulty and reports on them as sensitivity problems.

Backtrack

ErgoLight identifies situations of user confusion that rarely occur. The identification of such situations is provided by on-line detection and backtracking of the user actions.

User control

ErgoLight was designed carefully to provide user control in all operational situations. The main concern is that users might avoid usability testing if it interrupts the fluent operation of the software application. Another concern is the integration with known learning schemes, including learning by reading from the Help screens and learning by “trial and error”. At the learning stage, the problem indicators may be invoked frequently. Users might avoid usability testing at the learning stages to avoid frequent interruption of the fluent learning processes.

Test administration

ErgoLight implements a test set-up, including user groups. A recoof problem indicators and other parameters relevant to test administration, is assigned commonly to all users belonging to the same user group, thus, avoiding duplications in the parameter set-up.

Daly-Jones, O., Bevan, N. and Thomas, C. (1997). *Handbook of User-Centred Design*. D6.2.1 Version 1.1

Nielsen, J. (1993). *Usability Engineering*. Academic Press.

Shneiderman, B. (1992). *Designing the user interface: strategies for effective human-computer interaction*, 2nd edn, Reading, MA: Addison-Wesley.