I-UsE: Integrated Usability Evaluation Environment

Luís Marcelino Vasco Amaral
DI-FCT-UNL
Quinta da Torre, Caparica
{LMarcelino,Vasco.Amaral}@di.fct.unl.pt

Abstract
Software usability evaluation (UE) is a typically neglected phase by software developers. Due to the high costs of properly done formals usability studies, companies easily cut investment, contributing deliberately to the development of biased software with usability problems. Given this panorama, the development of UE tools is becoming increasingly important in order to tackle both the costs associated with the collection of data and the analysis of usability experiments.

This paper presents a practical software tool to support formal studies of UE by combining video recording, live screen capture and analysis features together. This open tool is being developed using standard hardware and free software components, making it a cost effective solution for UE.

Keywords
Usability Evaluation, HCI, Multimedia, User testing, Usability tools.

1. INTRODUCTION

The importance of good software usability is widely recognized and is even subject to quality regulation. For example, the standard ISO 9241 is a document with seventeen parts that specifies requirements of "Ergonomics of Human System Interaction". It is widely accepted that to achieve good usability, the software development process must include evaluation [Gediga02].

Despite the importance of software usability, it is often left to a secondary plan. Companies that assess their software's usability tend to commit few resources to those studies. Frequently, there is one single person responsible for testing the software, from finding bugs to find usability problems. As hiring adequate experts is costly, these tests are typically performed by one of the programmers that developed the software, contributing to biased and wrongly validated software.

For usability evaluations (UE) involving users, the percentage of problems identified increases with user's sample size [Faulkner03]. However, financial and temporal cost of the study also increases with the number of users involved in the study. This may be one of the reasons why companies commit so few resources to assess their software: the expense associated with more formal studies.

The expense in terms of financial and temporal resources of a usability study has two significant contributions: the cost of preparing and conducting the experiment and the cost of analysing the gathered data. The cost associated with the conduction of the experiment may be difficult to minimise, as it depends on the type and number of users involved. However, there are different strategies that may reduce the cost associated with the analysis of a usability experiment. An example is the use of automated tools to analyse collected data.

Our solution combines both recording of data to be collected during the usability experiments and analysis features. The presented solution adopts semi-automated usability evaluation methods, some of which already identified in the literature.

The next section presents some of these usability evaluation methods. Section 3 describes some projects aimed at automating the usability assessment process. The requirements for a tool able to assist generic usability studies are identified in Section 4. Section 5 includes details of the proposed platform and section 6 presents some of the early conclusions and outlines future developments for the presented solution.

2. EVALUATION METHODS

Results of usability evaluations can vary widely when studying the same interface. Molich et al. found that the identification of usability problems is not consistent across usability teams [Molich04]. Furthermore, for studies conducted by the same team with samples of 5 users, Faulkner reported that the same experiment might report from 55% to nearly all usability problems [Faulkner03]. As suggested by Ivory and Hearst, the automation of some aspects of usability evaluation is a solution to achieve systematised results [Ivory01].

Automation on usability testing methods has been used predominantly in two ways: automated capture of use data and automated analysis of log files according to some metric. In a survey about the state of the art in automating usability evaluation, Ivory and Hearst present a taxonomy for usability evaluation models [Ivory01]. This taxonomy of UE classifies five method classes and within these classifies several method types. Two of the method classes are particular relevant to this discussion: testing and enquiry.

The high number of automation approaches of log file analysis is a justification for Hilbert and Redmiles' survey about techniques to extract usability information from Interface Events. [Hilbert00]. They identify high level categories of techniques that emerged from their survey of different approaches:

- Synchronization and searching;
- Transformation;
- Analysis;
- Visualisation;
- Integrated Support.

3. EVALUATION TOOLS

This section reviews some of the tools available to support usability evaluations (UE), sometimes referenced as CAUsE-Tools (Computer-Aided Usability Engineering). The software tools presented may favour particular evaluation methods, some of which are not used in the proposed software solution.

3.1 WCAG Verification tools

There are a several tools to test and validate Web sites according to the Web Content Accessibility Guidelines (WCAG) [Chisholm99]. A list of some of these tools can be found at the Web Accessibility Initiative (WAI) site [Abou-Zahra06].

The tools listed in the WAI site, either in the form of software programs or online services, aim to determine if a web site is accessible to people with disabilities. Some of these tools start to determine the conformance of Web sites to accessibility checks that can be executed automatically. Some also suggest checks that need to be evaluated manually.

Even though most of the listed tools are used to accessibility validation or verification, different tools identify different possible issues. Brajnik compares two of these tools and characterise them according to their completeness, their correctness and their specificity [Brajnik04]. One aspect relevant from this study is the high number of false positive and negative issues reported by these tools. One of the tools reported more than 80% false negative issues generated by automatic or manual tests.

3.2 DRUM

The Diagnostic Recorder for Usability Measurement (DRUM) [Macleod93] is a CAUsE tool from the National Physical Laboratory. It is a tool that supports quantitative evaluation based on observation of task performance and analysis of how successfully people achieve task goals when using a system.

DRUM supports several steps associated with UE. It manages data acquired during the observational evaluation, allows the evaluator to select users and provides facilities for editing and browsing tasks during the analysis stage. DRUM is divided into four modules described next.

The **Evaluation Manager** allows the manipulation and display of data from the different stages of the UE. This manager has eight browser, one for each type of data: Subjects, Tasks, Recording plans, Evaluations, Video sessions, Logs, Measures and Reports.

The **Scheme Manager** enables evaluators to inspect existing events and add or edit custom events,. Custom events may represent sub-tasks or activities to create a hierarchical description of the tasks to be performed. This hierarchy can have up to four levels. Once a task scheme is created, it is stored in the DRUM database.

The **Recording Logger** has two sub-modules that provide the functionality needed to create and edit video-related logs. The Recorder assists log creation in real-time, while the Logger assists "retrospective" logging. To log events of interest and easily locate them, evaluators can specify Markers. The Logger also controls the video recorder, manipulating the tape as needed. If an event is not instantaneous, DRUM offers the possibility to view the associated clip. For that the evaluator only needs to click on that event.

The **Log Processor** performs the calculation, from any log in the DRUM database. The performance measures and performance-based usability metrics include: task time, help and search time, effectiveness, efficiency, etc.

One advantage of DRUM is its ability to control a video recording device and search for meaningful events. According to the authors, the use of time-tamped observed events, may reduce the time required to analyse a video by up to 80% [Macleod93].

DRUM's dependency on specific hardware is an obstacle to the adoption or even further development of this tool.

3.3 KALDI

The Keyboard/Mouse Action Logger and Display Instrument (KALDI) is a CAUsE tool that supports UE of Java-based graphical interfaces [Ivory01]. It is more than an action logger, as it is able to capture usage data and screen shots for Java applications. This software solution eliminates the requirement for specialised video recorder.

KALDI has three components: the Capture Tool, the Monitoring Tool and the Analysis Tool. The first two components are used during the usability test, while the Analysis Tool serves to assist in the evaluation.

The **Capture Cool** logs user events and takes up screenshots using the Java AWT functionality. This collected data is associated to a time stamp and sent to the Monitoring Tool over a network connection.

The **Monitoring Tool** enables the UE expert to structure and store the information collected by the Capture Tool.

The **Analysis Tool** assists the study of recorded events and associated pictures after the execution of the usability test. The analyst may associate several "raw events" to Actions or Tasks, building a meaningful hierarchy. A player is also available to play back screenshots taken during the test.

3.4 UsAGE

The User Action Graphing Effort (UsAGE) [Uehling95] is a usability tool that compares two files: one "expert" with one "novice". These files are generated by a utility called *usage_collect* of TAE Plus, which is a user interface development and management system.

After performing the comparison, UsAGE displays the result in a graph of action nodes. The "expert" series of actions is used as a reference and the "novice" actions are displayed as a comparison. In addition, it displays some metrics about the comparison results, including percentage of matches between expert and novice nodes.

3.5 Mercury TestDirector

The Mercury TestDirector [Mercury06] is a commercial application that supports the testing process: requirements management, planning, scheduling, running tests, issue management, and project status analysis. It uses a Web interface to allow testers and developers to participate and contribute to the testing process.

The testing process starts with the definition of the test requirements. Based on the requirements, testers build the test plan and design the actual tests. The next phase is to test the system as a whole. Tests can be scheduled and dependencies between them may be specified between them. Manual test execution is performed through a browser-based wizard that provides step-by-step guidance to the executer. Once this process is complete, TestDirector generates graphs and reports with the results of the tests.. It also provides an API that enables it to be integrated with custom solutions.

4. REQUIREMENTS

In a survey that presents a taxonomy for usability evaluation models only three from the ten methods used in user testing, have automation support: performance measurement, log file analysis, remote testing [Ivory01]. This may be an indicator of the difficulty to automate some steps in usability evaluation. Automated capture enables the evaluator to collect data for a larger number of users than traditional methods. In addition, automatic generation of usage data may reduce considerably the time required to review recorded testing sessions [Ivory01].

The UsAGE approach to data collection has two weak points: it requires the interface to be developed using TAE Plus and requires an expert execution of the task. The first point prevents it to be used with most commercial applications. The second week point is the notion of "expert" versus "novice" that is not always applicable.

Even after the identification of issues, their presentation may not be consensual. Classification of issues is frequently a subjective activity. It may vary between evaluators or even for on evaluator in different conditions [Brajnik04]. Having customizable ranking in the evaluation software may clearly identify the priorities for problem solving. If circumstances change, a simple modification of the priorities will reorder the issues found in the UE.

The discussion of methods and systems presented above was used to identify the requirements for the proposed integrated usability evaluation environment. The identified requirements are:

- Platform independence and open source;
- Dissociable from the developing environment;
- Capable to record user actions;
- Able to record video and audio streams;
- Support for planning the user evaluation;
- Integrated counting and statistical engine;
- Ability to generate reports;
- Extensible and modular;

5. IMPLEMENTATION

One of the requirements for the proposed tool is to be dissociated from the software to be tested. We believe that the proposed solution has minimal intrusion relative to the user's system.

As shown in Figure 1, our tool integrates three main components and two auxiliary ones, described in the remaining of this section.

The **Experiment Planner** enables the UE expert to develop a priori description of the tasks involved in the evaluation session.

The plan resulting from the Experiment Planner provides guidelines to run the experiment. The evaluator may define the Actions that define the user experiment.

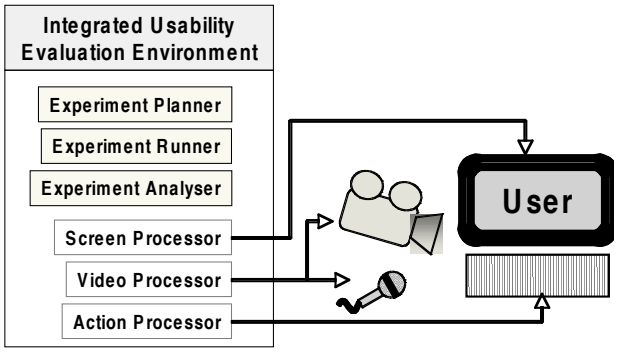

Figure 1: Overview of the I-UsE

The **Experiment Runner** controls the screen, video and action processors. In the beginning of each experiment, it instructs all the processors to start recording.

It is through the Runner that is identified the state of the current stage of the user experiment.

The **Experiment Analyser** enables the UE expert to analyse the recorded data.

The Analyser communicates with the processors to playback the corresponding stream. The analyst may introduce new marks or identify high-level Actions.

The **Screen Processor** enables the evaluator to capture and playback the user's screen. To achieve good screen resolution with the use of off the shelve components, we decided for a software solution.

To capture the user screen we use the free remote control software TightVNC. It is derived from VNC [Richardson98] software and has a Java implementation, making it easy to import as a system module. To capture the user's screen, a VNC server must be running on the user's computer. However, the VNC server does not send keystrokes nor mouse events. Thus, it was necessary to add the Action Processor.

The **Video Processor** manages video and audio streams. It may record live streams or playback existing ones.

To capture live video and audio using general purpose hardware we decided to use the Java Media Framework (JMF). Any video and audio hardware that is recognised by the JMF can be used to capture the live stream. Therefore, it is possible to use regular web cams and microphones as capture devices.

The **Action Processor** is a module in the planning phase. The goal is to manage user events: capture those events from the user machine and make them available to be combined with the remaining captured data.

6. CONCLUSION AND FUTURE WORK

In this paper we have presented the ongoing work concerning our tool designed to integrate data collection and UE analysis features. It was developed to tackle the problem of the lack of efficiency during this UE data study phase that leads to time consumption and consequently expensive studies.

In the future we plan to use our integrated usability evaluation environment in order to assist on the evaluation of a complex commercial system and therefore validate our claims of significant contribution to efficiency of the UE study phase.

We plan to extend our tool by introducing a domain specific (DSL) visual language to allow the usability expert to model UE sessions and generate tests automatically.

The technology used in this project enables to configure the tool for remote usage. This functionality enables a UE expert to guide the testing remotely, with users possibly being physical distant. The potential of this use case to our tool is a promising approach to reduce costs related to mobility and we plan to explore it further.

7. REFERÊNCIAS

[Abou-Zahra06] S. Abou-Zahra, et al., Web Accessibility Evaluation Tools. W3C, March 2006. `www.w3.org/WAI/ER/tools/Overview.html`

[Brajnik04] G. Brajnik. Comparing accessibility evaluation tools: a method for tool effectiveness. Universal Access in the Information Society, Vol. 3, No. 3-4, Springer Verlag, Oct. 2004. pp. 252-263.

[Chisholm99] W. Chisholm, G. Vanderheiden, I. Jacobs. Web Content Accessibility Guidelines. W3C Recommendation, May 1999. `w3.org/TR/1999/WAI-WEBCONTENT-19990505`

[Gediga02] G. Gediga, K. Hamborg, I. Düntsch. Evaluation of Software Systems. in A. Kent, J. Williams (Ed.), Encyclopedia of Library and Information Science, Vol. 72. pp. 166-192. http://citeseer.ist.psu.edu/gediga02evaluation.html

[Faulkner03] L. Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, & Computers*, Psychonomic Society, Inc., 2003, Vol. 35 (3), pp. 379-383.

[Hilbert00] D. Hilbert, D. Redmiles. Extracting Usability Information from User Interface Events. ACM Computing Surveys, Vol. 32, No. 4, December 2000, pp. 384–421.

[Ivory01] M. Ivory, M. Hearst. The State of the Art in Automating Usability Evaluation of User Interfaces. ACM Computing Surveys, Vol. 33, No. 4, December 2001, pp. 470–516.

[Macleod93] M. Macleod, R. Rengger. The Development of DRUM: A Software Tool for Video-assisted Usability Evaluation. BCS Conference on People and Computers VIII, HCI'93, Lougborough, 1993. `www.usabilitynet.org/papers/drum93.pdf`

[Mercury06] Mercury Interactive Corp. TestDirector for Quality Center, Data Sheet `http://www.mercury.com/us/pdf/products/datasheets/DS-1134-0306-testdirector.pdf`

[Molich04] R. Molich, M. Ede, K. Kaasgaard, B. Karyukin. Comparative usability evaluation. Behaviour & Information Technology, Taylor & Francis Ltd, Jan–Feb 2004, Vol. 23, No. 1, pp. 65–74.

[Richardson98] T. Richardson, Q. Stafford-Fraser, K.Wood, A. Hopper. Virtual Network Computing. IEEE Internet Computing. Vol. 2, No. 1, 1998. pp. 33-38.

[Uehling95] D. Uehling, K. Wolf. User Action Graphing Effort (UsAGE). CHI 95 Short Papers, ACM, Colorado, USA. May 1995