QUIP: Quantitative User Interface Profiling

Brian Helfrich
eXperimental Computing Facility
University of California at Berkeley
Berkeley, CA 94720 USA
brian@xcf.berkeley.edu

James A. Landay
Group for User Interface Research, CS Division
University of California at Berkeley
Berkeley, CA 94720 USA
landay@cs.berkeley.edu

ABSTRACT
We introduce a usability evaluation system that provides automatic, quantitative analysis of usage trace data. A problem with existing usability evaluation techniques such as expert evaluation and walk-throughs is that the results are subjective and qualitative. Observational research leads to the most relevant results, but it can be expensive, and analysis of the results is tedious. In the system that we introduce, usage traces are automatically aggregated into a usage profile in which disparities between user and designer conceptual models are exposed graphically.

Keywords
User interface, usability, automated evaluation, profiling

INTRODUCTION

Some problems with usability evaluation techniques such as expert evaluation and walk-throughs are that they do not involve real users, and the results are subjective and qualitative. Objective and quantitative information based on real usage of an interface is better for the purposes of understanding usability defects and planning resources for fixing them. A problem with observational research is that it often requires special equipment, facilities, and lots of time. In addition, the post-experiment analysis of the results is tedious and time-consuming.

Some usability evaluation systems provide automated forms of usage trace analysis and visualization. Guzdial et. al. reviews and introduces several of these systems in [2]. These systems each address important levels of analysis, but none of them gives a full profile of actual usage.

In this paper we introduce a usability evaluation system called QUIP that provides automatic, quantitative analysis of usage trace data obtained from real users running an instrumented version of a target application. The results are presented in the form of a usage profile graph.

For example, the usage profile graph in Figure 1 is an aggregate of the usage traces of two test users and the designer of the UI, all performing the same task. In the profile, each node is a user action, for example, node A might represent a "File|Open..." menu selection, node B a list item selection, node C an "Open" button click, and so on.

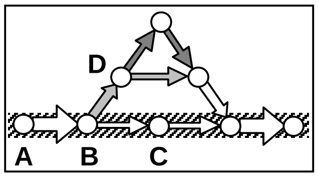

Figure 1 Example usage profile

The directed arcs between the nodes indicate the sequences of actions taken by the users. The path highlighted in green (diagonal shading in the above figure) corresponds to the path taken by the designer of the UI. The width of the arc corresponds to the fraction of test users who performed the subsequence. For example, the thick arc between action A and B indicates that all the users took this path, and the thin arc between action B and C indicates that only the designer of the UI took this path.

The color of the arcs correspond to the average time between the two actions for users taking that path. The color range is a continuous blue gradient from white, for one second or less, to fully saturated blue (dark gray in the above figure), for times exceeding ten seconds. For example, in Figure 1, we can see that all the users quickly performed action B after having completed action A, while there was (on average) some hesitation on the part of the test users before performing action D.

In the following sections, we give an overview of the system with reference to a pilot study and then give some of the results of the pilot study. The pilot was performed at a company developing a sophisticated Java-based marketing application. The test participants comprised five male marketing professionals, who were expert users of the application, and two of the UI developers.

SYSTEM OVERVIEW

To use QUIP, programmers first instrument the target application code. Users are then given tasks to complete, and action data is recorded. Finally, the system computes an aggregate profile graph that designers can use to detect UI defects. Here we describe these steps in more detail.

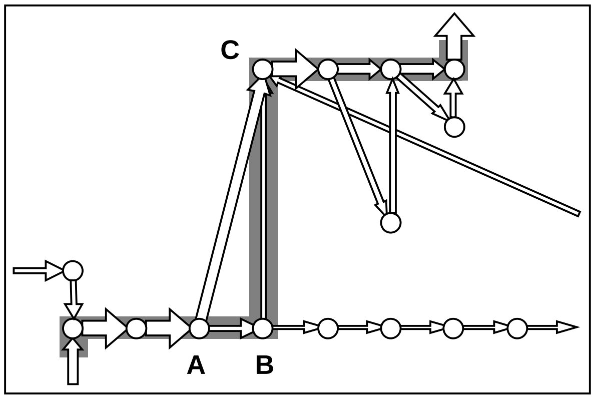

Figure 2. Omission of important action

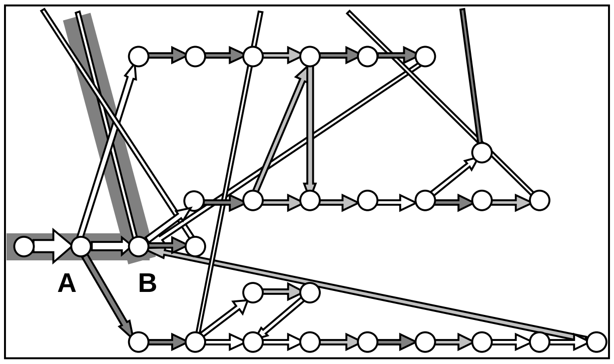

Figure 3. Confused divergence

Instrumenting the user interface

The UI under evaluation is instrumented such that each action that the user performs is recorded. The data recorded for each action includes a test session identifier, a timestamp (in milliseconds), descriptive information about where in the UI the action took place (e.g., file open dialog), the action (e.g., "Open" button clicked), and, if applicable, text entered by the end-user via keyboard. These records are written to a SQL Server™ database. Caching and a separate instrumentation thread are used to minimize effects on the performance of the application.

Recording the data

A high-level task description that exercises the UI being evaluated is prepared. The test users and UI designer(s) then carry out the task on the instrumented system. The *designer trace* represents the expected usage[1].

Profile generation

The usage traces are aggregated into a directed graph such that redundant action sequences are collapsed. For example, in Figure 1, three A→B sequences were collapsed into one, as were two B→D sequences. This technique highlights where usage traces diverge.

The redundant action sequences are determined using the Longest Common Substring (LCS) algorithm [1]. The running time of our algorithm is bound to that of the LCS algorithm, O(n^2), where n is the count of actions per trace. Performance for computing the graph from the pilot (six traces of about 500 actions each) was still acceptable (2 minutes) with completion time dependent primarily on the time spent reading the data from the database.

RESULTS

Each test participant completed the set of 10 tasks in about 20 minutes. The pilot study both confirmed the validity of suspected UI defects in the application and exposed unexpected defects.

Figure 2 shows the segment of the pilot usage profile where users are asked to create an integer field for an online survey. It shows that after naming the survey field (action A), 75% of the users skipped setting its type to Integer (action B), incorrectly leaving it as the default type Text, and continued on to add the next survey field (action C). The type of the field is important for performing calculations on field data. Previously, it was unknown whether even experienced users included the field types in their conceptual model of survey creation.

Figure 3 shows the segment of the pilot usage profile where users are asked to configure an automated e-mail message. This segment exposed an unexpected usability defect. This section of the UI is used to select the type of mail to send. If the mail is Electronic, the user need only select the desired message body (action B) and continue, but if it is physical mail, there are a multitude of options to configure. After setting the type of mail to Electronic (action A), the test users generated several different paths with relatively long inter-action delays. This can be interpreted as confusion over available but inapplicable mail configuration options, leading to unnecessary effort.

FUTURE WORK

An important aspect of interpreting the visualization is relating it back to the UI. Currently, QUIP displays action details in a separate panel when the mouse is moved over the action node. We are developing a more convenient method for viewing this information, along with ways to further examine the information represented by the arcs connecting the actions. Zooming and layout algorithms that help expose trouble areas will also be explored. We are also examining automatic highlighting of patterns such as the divergences illustrated in Figures 2 and 3, as well as patterns yet to be discovered, to speed detection of defects.

REFERENCES

1. Cormen, T.H., et al. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990, 314-319.
2. Guzdial, M., et al. Analyzing and Visualizing Log Files: A Computational Science of Usability. GVU Center TR GIT-GVU-94-8, Georgia Institute of Technology, 1994.

[1] In the pilot study, a single designer trace was composited from different sections in the two designer's traces. The sections correspond to the parts of the UI each designer designed.