WebEyeMapper and WebLogger: Tools for Analyzing Eye Tracking Data Collected in Web-use Studies

3 authors, including:

Peter Pirolli
Palo Alto Research Center
183 PUBLICATIONS   10,669 CITATIONS

SEE PROFILE

Stuart K. Card
Stanford University
194 PUBLICATIONS   22,105 CITATIONS

SEE PROFILE

WebEyeMapper and WebLogger: Tools for Analyzing Eye Tracking Data Collected in Web-use Studies

Robert W. Reeder, Peter Pirolli, Stuart K. Card
Xerox Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304 USA
+1 650 812 4749
rreeder@parc.xerox.com, pirolli@parc.xerox.com, card@parc.xerox.com

ABSTRACT
Eye trackers output a stream of points at which the eye was looking. To give these points meaning, researchers analyzing eye tracking data need to map the points onto the meaningful objects at which the eye was looking. Performing this mapping has proven to be a tedious, time-consuming task. We present a software system that automates this task for Web usabilty studies that incorporate eye tracking.

Keywords
WebEyeMapper, WebLogger, eye tracking, eye tracker, usabilty

INTRODUCTION
Eye trackers have been used in a number of HCI and usability studies [1,2,3]. A basic problem in such studies concerns the mapping of the eye tracking data (*points-of-regard* of the eye) onto graphical elements of the application and content (*elements-of-regard*). The mapping of points-of-regard onto elements-of-regard is called the *points-to-elements mapping*. Often, the points-to-elements mapping is done by painstaking analysis by hand, e.g. [3]. This hand analysis can be speeded up by tools that permit the analyst to rapidly define and label spatial areas that correspond to areas of displayed content, e.g. [2]. A template of defined content areas can be specified for each display state—for example for each and every view of WWW pages visited by a user. As the number of screen changes increases, however, so does the number of templates required for analysis. As a consequence, eye tracking studies are often limited to relatively static displays, and user behaviors, such as scrolling, that change the display rapidly are prohibited. The Web's dynamic nature has made it difficult to effectively analyze eye tracking data collected in Web usability studies. The display changes upon each transition to a new Web page, upon scrolling, and upon changes to the Web browser window. To enable analysis of eye tracking data for the Web, the points-to-elements mapping process needs to be automated. We present the WebEyeMapper system, an automated solution to the points-to-elements mapping problem.

WEB BROWSING PARADIGM
A Web study with eye tracking involves monitoring and eye tracking a user during a *browsing session*. During a browsing session, a user looks at a Web browser window on a computer display. The browser changes the display in response to user actions such as navigating to different Web pages and scrolling. The application-level semantic objects of interest are the Hyper Text Markup Language (HTML) elements that the browser renders on the display – text, images, hyperlinks, buttons, input boxes, etc. These are the elements in the points-to-elements mapping. Because it is computationally expensive, points-to-elements mapping must take place after the browsing session so as not to slow down the Web browser or the collection of eye tracking data. We will call the time of the browsing session *experiment-time* and the time at which we perform points-to-elements mapping *analysis-time*. Points-to-elements mapping requires two tools, one that runs at experiment-time and one that runs at analysis-time. The experiment-time tool must monitor the browsing session and record events that cause a change in the display or that relate to the eye tracker's experiment-time state. The analysis-time tool must convert the record of experiment-time events (the *event log*) and the list of points-of-regard from the eye tracker (the *raw eye tracker data*) into the desired representation of the elements-of-regard. In response to the need for these two tools, we have developed WebLogger, which records experiment-time events, and WebEyeMapper, which actually does points-to-elements mapping.
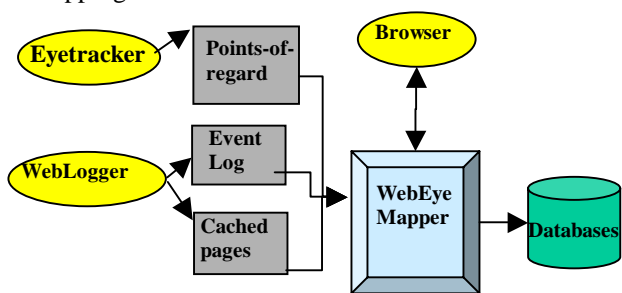

Figure 1. A diagram of the inputs and outputs of WebEyeMapper.

WEBLOGGER

WebLogger is an experiment-time tool that captures and records significant events during the browsing session. WebLogger's purpose is to capture and record all events that WebEyeMapper will later need to perform points-to-elements mapping. In fact, WebLogger is a solution to the more general problem of instrumenting a Web browser [4]. By instrumenting a Web browser, we mean capturing and recording, in real-time, all significant user and browser events. In addition to its role as the experiment-time complement to WebEyeMapper, WebLogger can be used on its own as a tool for collecting data on user actions during a browsing session.

WebLogger was developed for the Windows NT operating system and works with with Microsoft®'s Internet Explorer (IE) Web browser. When WebLogger starts, it simultaneously launches an instance of IE. WebLogger produces a text file, the *event log*, which documents user and application events that occur during a user's interaction with this instance of IE. WebLogger events relevant to points-to-elements mapping include browser events related to the state of the display and eye tracker events related to aligning eye tracking data in space and time with the display state. Logged browser events include events that change the Web page displayed, the portion of the page displayed, or the position or size of the IE window relative to the screen.

WebLogger also has a content-saving feature. It can save the actual Web content (i.e. the text, images, scripts, etc.) at which a user looked during a browsing session with IE. Because of the ever-changing nature of the Web, it is difficult to recall the exact content that a user saw during a browsing session, but WebLogger's content-saving feature makes this possible.

WEBEYEMAPPER

WebEyeMapper is our analysis-time tool for performing points-to-elements mapping. WebEyeMapper, like WebLogger, launches an instance of IE and maintains a pointer to the instance of IE. Using WebLogger event log data, raw eye tracker data, and content from WebLogger's content-saving feature, WebEyeMapper performs points-to-elements mapping (see Figure 1). First, WebEyeMapper does fixation analysis to convert points-of-regard from the eye tracker into fixations. Then it begins a "playback" of a browsing session. WebEyeMapper maintains an analyst-controlled simulation clock to coordinate the replay of WebLogger events and eye fixations. The zero point on the simulation clock corresponds to the beginning of the browsing session. As the simulation clock advances, WebEyeMapper directs IE to load the same Web pages that the user was viewing at the time indicated by the simulation clock, and directs IE to alter its scroll position, window

position, and window size as the user did at experiment-time. In this manner, WebEyeMapper restores the display state of the browser to the same state, moment-by-moment, as the user viewed it at experiment-time. WebEyeMapper can then take eye fixation points, align them in time with the simulation clock, align them in space with the browser window, and determine what is rendered in the browser at the time of each fixation. For each fixation, WebEyeMapper writes the fixation start time and duration, screen, window, and scroll system coordinates, element fixated, and element text fixated, to a database.

CONCLUSION

WebEyeMapper and WebLogger constitute a working system for collecting eye tracking and user event data during a Web-browsing session with Internet Explorer and converting those data into a database of actual elements fixated. The system is a breakthrough in technology for using eye tracking data for Web usability research. It enables researchers to make sense of eye tracking data collected for a user viewing a dynamic display, and automates points-to-elements mapping, which was previously a tedious process. The system will help researchers answer questions such as what type of HTML elements get the most user attention, how positioning of elements on a page aids or hinders users' perception of the page, and what words people read and don't read on a given page.

ACKNOWLEDGMENTS

REFERENCES

1. Aaltonen, A., Hyrskykari, A., and Raiha, K. 101 Spots, or How Do Users Read Menus? In *Human Factors in Computing Systems: CHI 98 Conference Proceedings* (pp. 132-139). ACM Press. 1998.

2. Lewenstein, M., Edwards, G., Tatar, D., and DeVigal, A. Stanford-Poynter Project. See http://www.poynter.org/eyetrack2000/index.htm. 2000.

3. Pirolli, P., Card, S.K., and Van Der Wege, M.M. The Effect of Information Scent on Searching Information Visualizations of Large Tree Structures. *AVI 2000*. 2000.

4. Reeder, R.W., Pirolli, P., Card, S.K. WebLogger: A Data Collection Tool for Web-use Studies. Technical report number UIR-R-2000-06 online at http://www.parc.xerox.com/istl/projects/uir/pubs/default .html. 2000.