Behavior Research Methods, Instruments, & Computers
1998, 30 (4), 573-582

Listener: A tool for client-side investigation of hypermedia navigation behavior

R. DARIN ELLIS, THOMAS B. JANKOWSKI, JARROD E. JASPER, and BALAJI S. THARUVAI
Wayne State University, Detroit, Michigan

Behavioral researchers have employed hypermedia-based software applications in their experiments for some time. More recently, interest in the World-Wide Web has developed among researchers in the social sciences, and popular use of this new medium continues to grow at an incredible rate. This paper describes Listener, a tool developed to log users' hypermedia and World-Wide Web navigation behavior using Apple Macintosh computers in a laboratory setting. Listener is able to capture navigation actions through cached documents, overcoming some of the problems associated with analyzing standard web server logs.

The growth of widespread use of the World-Wide Web (WWW) in recent years has spurred researchers in human–computer interaction to focus on usability issues regarding hypermedia (Cockburn & Jones, 1996). Important usability issues arise when one considers how people will navigate through this immense pool of hypermedia-based information. To help investigate such usability issues, we developed Listener to collect information about hypermedia navigation activity in a nonobtrusive fashion. Listener was designed for use in a laboratory environment to keep track of the user's step-by-step navigation through hypermedia spaces at the information-request level. Listener currently works on the Macintosh Operating System 7.5.1 (MacOS) with one family of browsers, specifically those produced by Netscape. Rather than being used to focus on the features of different browsers, however, Listener is designed to evaluate hypermedia navigation using embedded navigation objects (i.e., links). By performing research at the more standard level of the hypermedia itself, results will generalize beyond the context of a particular browser's use to hypermedia navigation behavior in general.

Hypermedia navigation is inherently sequential. Sequential data have long been used to investigate human–machine interaction, and thus other tools have been developed to accomplish this goal. These methods have included both direct external observation and background electronic monitoring. Many methods of direct observation and analysis of human–machine interaction behavior, collectively termed *exploratory sequential data analysis* (ESDA) by Sanderson and Fisher (1994), have recently grown in popularity with advances in the use of computerized tools such as MacSHAPA (Sanderson, McNeese, & Zaff, 1994; Sanderson, Scott, et al., 1994). This technique is capable of integrating video data, accompanying off-line codes and annotations, verbal protocol data, and activity logs, and has successfully been used in applications such as air traffic control (Vortec, Edwards, & Manning, 1994) and the design of student advising software (Sanderson, Iozzo, Buberel, & Au, 1995).

Although video- and verbal protocol-based data are very rich and extremely useful in performing system design and analysis, such protocols are not a very efficient means for collecting data on the user-action level of human–machine interaction. Analysis overhead, which is the amount of time spent analyzing data expressed as a ratio of the timespan recorded in the data, is estimated to be between 10:1 and 100:1 by many investigators. User-action level data recording is better accomplished through various electronic forms, in which data is automatically collected in a suitable form for processing. For example, Bourgeois and del Castillo (1993) investigated information search strategies in the review of legal trial transcripts using a HyperCard stack. After parsing the information into investigator-defined elements of analysis, they were able to record both the sequence of information acquisition and the time spent with each chunk of information. Salomon (1995) described the evaluation of an information kiosk that was set up at a major human–computer interaction conference in 1989. User navigation logs for the kiosk were obtained post hoc and analyzed to determine which sections of the information system were used most frequently.

Other researchers have extended this approach by applying it to investigate the popular, and relatively new, WWW browsing setting. To date, this research has required a high degree of expertise in programming and access to browser application source code. For example, both Catledge and Pitkow (1995) and Tauscher and Greenberg (1997a, 1997b) reprogrammed XMosaic to capture user interaction events and create an analyzable log.

Listener is designed to operate in a similar manner without the need for custom programming or browser source code modification, thus offering this capability to a much broader array of investigators. Listener has the

Correspondence should be addressed to R.D. Ellis, Institute of Gerontology, Wayne State University, 87 E. Ferry Street, 226 Knapp Building, Detroit, MI 48202 (e-mail: ellis@iog.wayne.edu).

added ability of being able to place Netscape into "kiosk mode," where all of the standard browser navigation controls are hidden. In kiosk mode, the user must navigate solely through links embedded in the hypermedia. This feature facilitates rigorous investigation of hypermedia formatting and navigation in a manner that generalizes beyond the specifics of particular browsers. It also allows investigators in the behavioral sciences to design hypermedia-based experiments that can be run using common browser software and simple hypertext markup language (HTML) documents instead of the more complex programming required with other hypermedia methods. By monitoring and recording detailed information on the hypermedia navigation behavior of an individual, Listener can be used to help answer questions regarding WWW design and navigation and to facilitate other types of hypermedia-based behavioral experiments in an easy and inexpensive manner, either alone or in a complementary capacity to data obtained through video-based techniques such as MacSHAPA.

TECHNICAL BACKGROUND

Listener uses the AppleScript scripting language to address functions available in Netscape's Macintosh Application Program Interface (API). These functions, described fully below, enable collection of information about each WWW page viewed through the Netscape browser. Listener interfaces with Netscape each time a new page element is downloaded into the browser. These elements, called hypertext transfer protocol (HTTP) objects, may represent a page definition document or an inline page element such as an image or a frame. For each "hit" or HTTP object downloaded, Listener records information on the sequence of the hit and the page in which it is placed, the time that the requested object finished downloading, the location (URL, or uniform resource locator) of the object, the title of the browser page in which the object appears, the type of object downloaded according to the Multipurpose Internet Mail Extensions (MIME) standard, and the URL of the page that referred the hit. Although Listener is capable of recording information for each element in every WWW page, the investigator may configure Listener to disable image logging, which simplifies the content of the data file and avoids potential problems outlined in the "Limitations" section below. Listener can be used with Netscape operating in its default configuration, but it also gives the investigator the option of running the browser in kiosk mode. Kiosk mode is a special setting in Netscape browsers, unavailable through the standard user interface, which hides almost all of the browser's controls normally available to the user, including all navigation menus and buttons and other controls. This allows Netscape to function as a general purpose hypermedia viewer and is useful in settings in which the investigator wants to observe users navigating a hypermedia space in the absence of user interface controls specific to the Netscape browser. Use of kiosk mode also prevents problems, discussed in the "Limitations" section, associated with recording certain navigation actions through Netscape's user interface.

Apple Events: The Enabling Technology

Listener relies on technology available in the AppleScript system-level scripting language that allows the development of simple programs to automate and customize applications running on a Macintosh computer. The MacOS communicates with applications through Apple Events, which are small packets of commands and information sent between programs. For example, the MacOS Finder launches an application on the computer by sending the Apple Event *open* to the application when the user double-clicks its icon. In a similar way, when a user double-clicks the icon of a document created by the application, the Finder reads the document's attributes, including information on the application that created it, and passes the *open* Apple Event to that application with the added request to open a new window containing the document. Hence, Apple Events are the primary means of communication between applications on the MacOS.

All applications created for MacOS must support a basic set of Apple Events (e.g., *open* to launch the application and *print* to print a document) as part of the Open Scripting Architecture (OSA) specification that Apple supplies to application developers. Many applications contain additional commands available to the user either through AppleScript or a scripting element inside the program itself. AppleScript allows the user to create applications that can send, receive, and process Apple Events from other applications and the Finder.

Scripting With Netscape Browsers

The Netscape family of browsers such as Navigator and Communicator support a number of additional Apple Events. Information about these is available through the application's Apple Events "dictionary." All applications that are AppleScript compliant contain this dictionary information, which is viewable through the AppleScript Script Editor. More detailed information on the Netscape API is available in the standards documentation that Netscape Communications Corporation provides through its WWW server (http://home.netscape.com/newsref/std/mac-remote-control.html). The event names used in this article are taken directly from the Netscape Navigator dictionary and are indicated by italics.

Apple Events have distinct properties that are used in sending and receiving information. For Netscape these properties include an event ID, an argument keyword and type, and, in the case of events that return feedback, a returned value and value type. The additional events that Netscape supports are divided into four categories: miscellaneous WWW-related events, events that pertain to application windows, events that allow Netscape to converse with registered applications, and events sent by Netscape to other registered applications on the Macintosh. The miscellaneous WWW-related events supported

by Netscape are *OpenURL, ShowFile, parse anchor, cancel progress*, and *find URL*. These events can be sent from an AppleScript application to Netscape to control various aspects of browsing activity. For example, sending the event *OpenURL* (http://www.iog.wayne.edu/homepage.html) tells Netscape to retrieve the document "homepage.html" from the WWW server "www.iog.wayne.edu" using the HTTP protocol. This action will cause the browser to download the page and all of its accompanying elements into the active browser window. Listener uses the *OpenURL* event to load a default page when the application is launched.

The second set of events deals with the windows viewed in Netscape. These are *webActivate, list windows,* and *get window info*. Listener uses the *list windows* event to get information about the currently open windows in Netscape. It also invokes the *get window info* event to get information about the active window when documenting individual downloads. The *webActivate* event is used to make and keep the browser the frontmost active application window even in cases in which Listener is actually the active application. This ensures that the display remains stable and Listener remains in the background.

The third set of events deals with registering external applications to communicate with Netscape. These include *register URL echo*, *register viewer*, *register protocol*, *register window close*, and the corresponding *unregister* events. Registering an application with Netscape means that the application will be referred to whenever Netscape processes an activity related to the registered application. For example, an e-mail program can be set to automatically process any e-mail address links in web pages by registering the e-mail package as the "handler" of the e-mail protocol with the *register protocol* event. Listener uses the *register URL echo* event to receive "echoes" of the URL of each HTTP object downloaded in Netscape.

The last events are those sent by Netscape to registered applications. These are *View doc file*, *Begin progress*, *Set progress range*, *Making progress*, *End progress*, *Query viewer*, *Echo URL*, and *window closed*. Listener receives the *Echo URL* event at the completion of each downloaded HTTP object, including inline images.

PROCEDURE FOR USING LISTENER

The investigator may open the Netscape browser application first and then launch Listener, or alternatively, launch Listener first. When Listener is launched first, it opens the designated browser automatically. In this case, care must be taken to set Netscape to open with only a browser window; launching Listener while Netscape has a mail or news window or the component bar in Communicator open will interfere with the proper operation of Listener, as described in the "Limitations" section below. Listener is launched either by double-clicking on the application icon in the finder or by highlighting the application icon and selecting Open from the file menu. Either action will cause the System Finder to issue an *open* Apple Event to launch the Listener application.

The first time that Listener is launched after installation, a dialogue box appears that allows the investigator to locate the desired Netscape browser. This is because Listener uses a generic application addressing scheme that allows for the use of any version of Netscape located on any network drive accessible from the Macintosh desktop. Once this initial selection is made, Listener will automatically open the designated browser whenever it is launched.

Listener features several options that may be edited by the investigator through the user interface. When it is launched, it presents a dialogue box that outlines the current state of these settings. The first option is to run Netscape in either normal or kiosk mode. If kiosk mode is chosen, Netscape hides all of the navigation and directory control options from the directory menu as well as from the toolbar located at the top of the browser window. If normal mode is chosen, the Netscape browser presents the participant with all of the default navigation and directory controls. The second option allows the investigator to choose whether or not images that are downloaded as part of a page request will be logged. When image logging is turned off, the log file is smaller and consists mainly of "text/html" MIME types. In most instances it is not necessary to log images, and in these cases it is recommended that image logging be turned off. (Note, however, that if the user navigates to a URL that contains only an image, that page will not be logged under these circumstances.)

The final options involve the log file-naming scheme. The log file name has two components: A log file base name and a log sequence number. The default base name is "data.log," and the sequence number is a three-digit number that is appended to the base name either before the first period in the name or, if no extension is employed, after the base name. For example, the first default file name will be "data001.log," followed by "data002.log," and so on. This feature was implemented so that successive Listener sessions can be run quickly with minimal input by the investigator. The settings dialogue box presented by Listener upon launch allows the investigator to accept all settings as is, to reset the file name sequence to "001" only, or to edit all settings. If the investigator chooses to edit all settings, the file name sequence is reset and a series of dialogue boxes is presented that enables the browser mode, image logging status, and log file name base to be changed. Finally, a standard "save file" dialogue is presented with the default log file name in the text box, so that the session can be started by merely clicking the Save button or hitting the return key.

After the log file is created, participants are presented with a dialogue box informing them that their session is being recorded and asking that they consent to being observed in this manner, similar to the procedure used by

Catledge and Pitkow (1995). If the participant clicks the Quit Now button, Listener quits, and the log file indicates that the participant refused. If the participant clicks the I Agree button, the browser becomes the top-most application in the desktop view, and a default home page is loaded in the browser window by Listener. The default home page must always be an HTML document entitled "default.html" and placed in a subdirectory named "docs," which is located in the directory holding the active copy of Listener. The investigator may replace the "default.html" page with any valid HTML document as long as the name and location of the document remain the same.

After participant consent is secured and the default home page loads in the browser window, the remainder of the session appears just like any other browser session. From this point on, Listener displays no dialogue boxes or windows. The only indication that Listener is running (i.e., that user behavior is being logged) is that Listener will appear in the list of currently active applications. At the end of the session, the log file is closed by selecting Listener from the list of active applications and then selecting Quit from the File menu or using the command-Q keystroke combination.

NAVIGATION LOG DESCRIPTION

The user's navigation is logged into a data file that is specified during the Listener start-up sequence. The data is stored as tab-delimited text and thus is amenable to direct translation into spreadsheets, database software, and statistical packages. Each data record is composed of seven variables retrieved from the browser as a direct or indirect result of a user-initiated navigation action. There are some limitations in the Netscape API that prevent certain navigation actions from being logged, and those will be discussed in detail below.

Each navigation action passes event parameters to Listener, which are recorded in the following fields: a sequential page number, a sequential hit number, the time that the requested information finished downloading, the URL of the object downloaded, the page title, the MIME type, and the referring page. See Table 1 for a definition of each field.

These parameters are used to reconstruct the navigation activity of users both in a controlled information space (e.g., a tutorial or experiment developed on the researcher's local hard drive) and in naturalistic browsing situations on the Internet. Hits are differentiated from pages for important reasons. A page is composed of many HTTP objects, one of which is an HTML text document holding the page layout definitions and most of the page text. The other HTTP objects are multimedia elements, most commonly inline images. Each of these objects has its own event associated with downloading it, and thus is logged as an individual hit. The page represents the information target of interest in most research, with hits logged as HTTP objects representing elements of that composite target. As noted, the user may opt not to include downloaded images in the log file.

Table 1
Parameters Recorded in the Listener Navigation Log

Data Record Parameter	Description
Page	A running count of pages, which are defined as a group of HTTP objects collectively displayed in a unique browser window. Pages are determined using the URL string parsed from the results of the *get window info* event, which is invoked each time an *Echo URL* event occurs.
Hit	A running count of the number of HTTP objects that have been downloaded by the browser, either through user navigation action or by the requested page in the case of inline elements such as images and frames. The hit is calculated as a count of *Echo URL* events.
Time	The time at the completion of the HTTP object download, returned by the *Echo URL* event.
URL	The URL for each hit downloaded by the browser, returned by the *Echo URL* event each time an HTTP object completes downloading.
Page title	The page title is parsed from information returned by the *get window info* event. It usually represents the contents of the <TITLE> tag in a standard HTML page header, and it matches the text displayed after the "Netscape:" prefix in the title bar of the active browser window.
MIME type	MIME type of the HTTP object downloaded, returned by the *Echo URL* event.
Referring page	The URL of the page from which the current downloaded HTTP object was requested, returned by the *Echo URL* event. The referring page may represent the page from which the user selected a link or the page on which an inline element is displayed.

Time is recorded, but the total time that a user spends on a page can only be roughly estimated with this tool because the browser alerts Listener at the completion of a download request. That is, the record is time-stamped when the download is complete, not when the user clicks on a link. Thus, intervals calculated by subtracting one hit's time from the next in sequence include (1) time spent with a completely downloaded page and (2) time spent viewing the loaded elements of an incompletely loaded page. This differs from more rigorously controlled paradigms in cognitive psychology—for example, where stimulus presentation with tachistoscopes is nearly instantaneous and timing begins at stimulus onset. Unfortunately, stimulus presentation over the WWW is not nearly this precise, particularly when stimuli are de-

livered over the Internet from a remote server. Hosting stimuli on a local area network (LAN) server or the workstation's local hard drive can dramatically reduce this problem. However, Listener is primarily intended for situations in which the experimenter is interested in investigating information retrieval behavior in which variables such as the direction and number of steps taken in a path to an information goal are more important than the precise amount of time spent on a given page.

Usage Example: Different Paths to the Same Information Target

Consider a hypothetical user who has been instructed to retrieve a given information target on a WWW site that includes both categories for browsing and a search engine (Figure 1). In Figure 1, boxes represent web pages, and arrows represent navigation actions. The specific action taken is noted in the circles. The gray lines represent paths that are available but do not lead to the target.

Listener provides information with which the investigator can distinguish between users of the two different valid navigation paths represented in black in the figure. Both paths start out on a home page that provides instructions on the navigation options, as well as navigation controls with which to choose between the two paths. The navigation controls include a list of categories and a Browse button, on the one hand, and search string text box and a Search button, on the other. In the path to the left, the user chose from a list of categories that represented available navigation actions. The navigation action in this case was to select the category and click the Browse button. This led the user to the relevant category page, which contains links to WWW pages that are members of that category. The user then clicked the link to the target WWW page from the list of relevant category members, leading the user to the target WWW page. The log in this example would contain three hits, all of which would contain the information presented in Table 1: (1) a record of the instructions page, (2) a record of the relevant category page, and (3) a record of the target WWW page. In the path on the right side of Figure 1, the user typed in a search term. The navigation action in this case was to enter the term and click the Search button. This led the user to a page of search results, which contains links to WWW pages that contain the search term. The user then clicked the link to the target WWW page from the list of search results, leading the user to the target WWW page. The log in this example would also contain records on three pages, all of which would contain the information presented in Table 1: (1) a record of hitting the instructions page, (2) a record of a search results page, and (3) a record of the target WWW page. Note that both of these examples demonstrate paths that

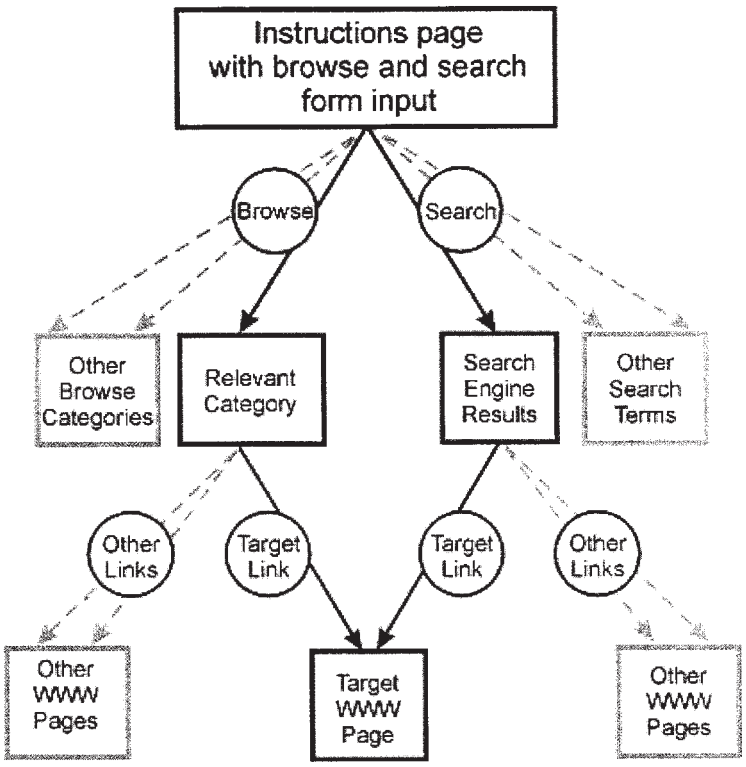

Figure 1. An example hypermedia space, with multiple navigation paths to a given information target. See text for description of terminology.

are efficient routes to the target information; that is, there are no errors in navigation that would lead the user off the path to the target information. The experimenter could manipulate the variables in this example and observe the results in the navigation log. For instance, factors that could be varied include the user's semantic familiarity with the target information and the perceptual saliency of the web site search capability (e.g., a prominent graphical button labeled Search versus a small bit of hyperlinked text in the middle of a paragraph).

The gray lines in Figure 1 demonstrate alternative paths that would be less efficient in reaching the target information, forcing the user to backtrack to reach the target. These missteps could include choosing the wrong category from the Browse list, typing in the wrong search term, or choosing a link from a search results list that did not contain the target information. The Listener log records all user actions, not just goal-relevant actions; the log allows the investigator to analyze all navigation events, including navigation errors such as backtracking. In the example above, the user's hypothetical return path from the target information to the instructions page can be explicitly determined by the information requests that are made. This is possible because the log contains information not available in a server-side log. In the example, the user could return to the instructions page via either a series of Back button clicks (revisiting previous pages in reverse sequence) or use of the History list (direct return to the top level). With each of these methods, the revisited pages are logged as "internal/parser" HTTP objects, although images and other inline page elements served from the browser cache are not logged. Capturing data on navigation of cached information is an important feature of Listener; problems with users' mental models of browser-based history controls was the focus of criticism in Cockburn and Jones's (1996) usability analysis of WWW browser applications.

OPTIONS FOR LISTENER LOG DATA ANALYSIS

Listener itself provides no postcollection log processing support. However, in addition to the ESDA technique discussed in the introduction, there are numerous quantitative techniques reported in the literature for analyzing sequential data. These include computational user modeling, advanced statistical and mathematical techniques, and even standard statistical analysis.

Log data collected by Listener could also be useful for computational user modeling techniques such as GOMS (Goals, Operators, Methods, and Selection techniques) modeling (Card, Moran, & Newell, 1980, 1983). The application of GOMS to the modeling of hypertext browsing has been accomplished with other software programs, such as HyperCard (Carmel, Crawford, & Chen, 1992). Although GOMS and other formal user modeling techniques usually operate at the keystroke level, there are times when the unit of analysis is at the more abstract, functional level of action. In fact, John and Vera (1992) modeled a highly interactive computer task and found that the functional level model predicted performance nearly perfectly, while a keystroke model was only about 60% accurate. This mirrored earlier work in text editing (Card et al., 1983). In WWW browsing, the functional-level model is defined by *which information* is retrieved in *what sequence*, whereas the keystroke model is defined by which menu selections and button clicks result in the retrieval of given information.

There are also numerous, although not particularly popular, advanced statistical and mathematical techniques for modeling sequential data. Cuomo and Sharit (1989) studied sequential behavior using an architectural computer-aided design system. They analyzed usage with numerous methods, including Markov-chain analysis, and discussed in detail the theoretical and applied benefits of each technique. Rauterberg (1993) implemented a method to automatically extract user mental models from database navigation patterns using a combination of Markov-chain analysis and Petri nets. Olson, Herbsleb, and Reuter (1994) discussed the use of log-linear modeling and lag-sequential analysis in modeling transitions between tasks in a computer-supported collaborative work environment.

Finally, researchers are deriving aggregate measures of behavior that are amenable to standard statistical modeling techniques such as regression and analysis of variance. For example, Catledge and Pitkow (1995) analyzed the average path length and the average number of visits to a given site across browsing sessions, finding evidence for a three-category model of navigation behavior proposed by Cove and Walsh (1988). Reed and Geissler (1995) used correlational and nonparametric methods to evaluate the effect of prior computer experience on dependent measures such as the frequency and percentage of linear and nonlinear steps in hypertext navigation. Carmel et al. (1992) analyzed, among other things, the effect of task and system expertise on dependent variables derived from users' browse paths, search behaviors, and topic selections. Mohageg (1992) analyzed the effect of hypertext linking structure on task completion time, number of errors, deviations from optimal path, and an uncertainty measure derived from the between-subjects variation in the paths chosen.

BENEFITS, LIMITATIONS, AND SETTINGS FOR USAGE

Benefits

Most HTTP servers log access to the server, but there are several benefits to using the client-side Listener software. First, Listener can be used to observe browsing behavior on the WWW, even when the investigator does not have access to the logs of the servers being visited. This allows for ecological validity not previously avail-

able without using more cumbersome methods of monitoring browsing behavior, such as video recording or verbal protocol analysis.

Second, even if the server access log is available to the investigator, client-side logging allows browser activity that is not logged by the server to be recorded. Specifically, most server logs do not indicate a hit on a URL unless it is actually downloaded by the browser client. However, the browser stores downloaded documents in a cache to avoid redundant downloading in the event that the browser's navigation commands (e.g., Back, Forward) are used to revisit a URL. Thus, when the Back button is used on the client, the server does not log a hit, but Listener records an *Echo URL* event with a MIME type of "internal/parser" for the HTML document that defines the page. Although the log cannot directly distinguish between the browser commands used to reach a cached document (e.g., Back button vs. History list selection), this capability provides a much clearer picture of the actual browsing behavior than that provided by server-side logs.

Third, Listener will even work without an actual HTTP server connection. The *Echo URL* event notifies Listener of completed requests for all URLs, regardless of protocol. Downloads using URLs that refer to files residing on the hard drive, floppy drive, or mounted network volumes are also recorded. Researchers can host their own experimental hypermedia or simulate a WWW site on a local drive, investigating browsing behavior on a workstation without an Internet connection. This is an important feature of Listener for investigators who do not have access to WWW servers or workstations with fast network connections.

Finally, although not implemented in this version of Listener, there is the capability to automate other aspects of the data collection for a given investigation. For those investigators who are familiar with AppleScript, the script can be edited to administer a questionnaire to the participant regarding computer-related background and attitudes, demographics, and so on, and to include his/her responses in the output file. Expanding Listener's capabilities in this manner could further decrease analysis overhead relative to the observation techniques discussed earlier.

Limitations

Listener's ability to log some types of requests is limited by constraints in the Netscape API and the *Echo URL* event specification. In particular, requests made in the absence of a referring page are not registered by the *Echo URL* event. This can occur when the home page specified in Netscape's General Preferences is loaded or when a user types a URL directly into the Location field, types a URL in an Open dialogue box, or uses a Bookmark. It can also occur when the participant uses the navigation buttons, makes a selection from the Go menu, or uses the History dialogue to revisit a hypermedia page that was originally reached through one of the first three navigation methods. In all of these cases, the primary part of the page (generally denoted as "text/html" in the MIME type field of the log) will not be registered by Listener, but hits on embedded HTTP objects such as inline images will. The referring page for these objects will be the page that was not logged. Thus, although inconvenient, it is possible to infer navigation with these browser user interface controls.

To avoid such problems at the beginning of a session, Listener was designed to load a default home page that is assigned a referring URL in the browser window, and this is always the first page logged. For that reason, when Listener is launched, Netscape should be set to have only a browser window open and all other types of windows, such as the mail or news window or Communicator's component bar, closed. Starting Netscape with an improper window configuration will interfere with the loading of the default page and may affect the logging of subsequent hits. *Echo URL* event limitations later in the session can be prevented by using investigator-controlled hypermedia in which all browser navigation choices can be accommodated with embedded links. By then running Listener with Netscape in kiosk mode, none of the problematic navigation actions or user controls are available to the participant.

Also, Listener is not capable of accurately recording hypermedia objects that have not been allowed to download completely in a given page. For example, if a user clicked very quickly through the links that connected a series of pages, particularly when the pages are downloaded from a slow remote server, objects that were not completely downloaded would not be logged, or would be logged with an incorrect page title. This is true of both text and image objects. After an *Echo URL* event is registered, Listener employs the *get window info* event to extract the page URL and title from the active window. If an HTTP object that has been requested as part of a page completes downloading after the user has moved on to the next page, the *get window info* event will return the wrong URL and title for that object's page. Although the problem is not as apparent when images are not being logged, it can still cause the page count to be inaccurate in those cases. This problem is minimized when the process of downloading occurs very rapidly, as is the case when the requested objects reside on a local or network hard drive or a WWW server on a LAN. It can also be minimized in experimental settings by using stimuli containing a limited amount of inline image content. It can be prevented, of course, by allowing all elements of a page to load completely before another navigation action is taken.

A similar situation in which a hypermedia request will not be logged properly is when the user interrupts the transfer with actions such as clicking the Stop or Back

buttons. Running Netscape in kiosk mode can ameliorate this problem, as it can other problems arising from the use of the browser's navigation controls. In addition, as one would expect, requested hypermedia objects will not be recorded in the log if the user quits the browser application or Listener itself before they are finished downloading.

When a URL is irregularly formatted, such as a string of variables passed to a CGI, it is sometimes difficult to determine the MIME type of a downloaded object. Listener relies on Netscape's interpretation of the HTTP object; if a page contains a link to a CGI that returns a text string, Netscape will interpret the CGI as MIME type "text/html" and Listener will log it as such, regardless of the actual MIME type of the object. Internal links followed within a document will not be recorded in the log when they are initially hit, but they may appear as referring pages on subsequent hits. The URL is the only information in the log to discriminate between these two types of page requests: The URL for referring pages that have been navigated using internal document links includes the HTML "name" reference (e.g., http://www.example.site/examplepage.html#name).

Listener is subject to inherent limits on the size of its log file. If the log file grows beyond this limit, the script will crash. The maximum log file size is around 32K due to limitations in the MacOS object specification. In practice, this would not be likely to occur, since it would represent several hundred URL requests. Careful design of experimental protocols should work around all of these problems.

Another limitation is the timing mechanism. Currently, the timing information resolution is accurate only to the second, leaving nearly a 2-sec potential for error. For a page of information that is typically viewed for 20 sec or less, this is a 10% or greater margin of error, and may be unacceptable to some investigators for whom timing information is critical. This limitation is inherent in the specification of the date class objects in the AppleScript architecture. Another timing limitation, as noted, is that the time intervals reflected in the Listener log include server download time for the subsequent object as well as time spent on the current page. For controlled experiments that are designed to use time information as a performance measure, care should be taken to design hypermedia that downloads quickly. For investigators who wish to design their own stimulus information for this purpose, we suggest that pages be limited in the amount of text and images they contain. Further, these stimulus web pages should be placed on the local hard drive of the testing machine and referenced with "file:///" URLs to minimize access time.

Limitations in the Netscape API give rise to a few functional limitations in Listener that should be pointed out. Listener records navigation through an information space defined by HTML pages and logs only requests for information. Thus, Listener records neither within-page navigation (scrolling or within-page hyperlinks) nor browser user interface commands. When a user requests one piece of information in succession after another, Listener is not capable of distinguishing between a History list selection, a toolbar button click, or a Go menu selection. However, as noted, Listener is not intended to be a tool for the evaluation of the browser user interface. Following Shneiderman's (1992) syntactic-semantic model of user knowledge, the limitations are in recording syntax-level interaction. These limitations are not relevant if (1) the object of investigation is at the semantic, goal-oriented level of interaction, and (2) the information being navigated is parsed down to the level of one information chunk per page.

Setting for Usage

Given the benefits and limitations listed above, the investigator can empirically control two distinct dimensions on which Listener's effectiveness varies: interface control and content control (Table 2). There are primarily two levels of interface control: normal mode and kiosk mode.

Table 2
Settings for Using Listener

Interface Control	Content Control: High	Content Control: Low
High	Lab Setting I Navigation of experimenter-created content with experimenter-defined links. Content accessed locally (e.g., hard drive or AppleTalk network). See Mohageg (1992), Bourgeois and del Castillo (1993).	Kiosk Setting Navigation of existing hypermedia content via limited navigation controls (i.e., hyperlinks embedded in the content). See Salomon (1995).
Low	Lab Setting II Navigation of experimenter-created content with standard browser controls, including bookmarks, forward and back buttons, and direct URL entry. Content accessed locally (e.g., hard drive or AppleTalk network). See Cockburn and Jones (1996).	Natural Setting Navigation of online WWW-content with standard browser controls, including bookmarks, forward and back buttons, and direct URL entry. See Catledge and Pitkow (1995), Tauscher and Greenberg (1997a, 1997b).

Other, more intermediate levels of interface control could be attempted with instructions to participants (e.g., "Don't use the bookmarks") and close monitoring for compliance. Content control varies from locally hosted, investigator-created content to real-world online WWW content. Listener's limitations along these dimensions are best described in the context of the existing literature on hypermedia navigation behavior. Listener is best equipped to provide accurate depictions of user navigation behavior in a high-interface-control–high-content-control setting (Lab Setting I cell of Table 2). Investigations of this nature would be similar to those described by Bourgeois and del Castillo (1993) and Mohageg (1992). Listener is well equipped for usage in the Kiosk Setting and Lab Setting II, although some of the limitations discussed above have a chance of occurring. In Lab Setting II, the experimenter determines the content while allowing the use of all navigation controls. This setting would not prevent the referring-page issue (described above) from arising; however, with a small and well-defined information space, missing log entries arising from lack of a referring page could be diagnosed and inferred post hoc. Coupled with other data collection techniques, this setting is well suited for investigation of various navigation tools and the effect they have on behavior (see Cockburn & Jones, 1996). In the Kiosk Setting, the limitations of Listener caused by the various browser menu commands are avoided, and information can be gained on circumscribed user behavior (see Salomon, 1995, for an investigation of kiosk browsing). However, browsing online WWW content in kiosk mode presents a different problem: Content is often designed with dead-ends—that is, pages that required the user to "click the Back button to return to the home page." Finally, Listener may be used to study natural, online Web-browsing situations with standard Navigator settings. It is under these conditions that Listener is most limited. Although Listener offers advantages over server-side logs, it should be used only in this setting when better techniques (i.e., reprogrammed browsers, e.g. Tauscher & Greenberg, 1997b) are unavailable.

FUTURE DIRECTIONS, SYSTEM REQUIREMENTS, AND AVAILABILITY

Many researchers use the Macintosh platform for experimentation, but there is no reason that Listener cannot be implemented on other platforms as well. For example, the *Echo URL* Apple event has an analogous command (referred to as Topics) in the Windows DDE implementation of the Netscape API, *WWW_URLEcho*. Developing Listener on the Windows platform would take advantage of the vast installed base of such machines. Similarly, Listener could be modified to work with other browser software on the MacOS and Windows platforms, such as Microsoft Internet Explorer (MSIE). Since the Apple Events dictionary of MSIE differs from that of Netscape, Listener in its current form will not work with MSIE.

Listener requires MacOS 7.5.1 (or greater) and the AppleScript system extension. Our testing indicates that Listener works well with Navigator Versions 2.0, 2.02, and 3.0; Navigator Gold; and Version 4.0 beta 5 of Communicator. Future official releases of Netscape browser products will be evaluated with Listener when they are released. Any changes to the Netscape API in these future versions will likely necessitate modification in future versions of Listener.

Listener is available free of charge in the compiled form or as AppleScript source code from the authors via e-mail at giw@iog.wayne.edu. All correspondence regarding Listener, including bug reports, questions, and suggestions, should be sent to the same address.

REFERENCES

Bourgeois, M. J., & del Castillo, D. (1993). A HyperCard program to track information-search strategies. *Behavior Research Methods, Instruments, & Computers*, **25**, 396-399.

Card, S., Moran, T. P., & Newell, A. (1980). The keystroke level model for user-performance with interactive systems. *Communications of the ACM*, **23**, 396-410.

Card, S., Moran, T. P., & Newell, A. (1983). *The psychology of human–computer interaction*. Hillsdale, NJ: Erlbaum.

Carmel, E., Crawford, S., & Chen, H. (1992). Browsing in hypertext: A cognitive study. *IEEE Transactions on Systems, Man, & Cybernetics*, **22**, 865-883.

Catledge, L. D., & Pitkow J. E. (1995). Characterizing browsing strategies in the world-wide-web. *Computer Networks and ISDN Systems*, **27**, 1065-1073.

Cockburn, A., & Jones, S. (1996). Which way now? Analyzing and easing inadequacies in WWW navigation. *International Journal of Human–Computer Studies*, **45**, 105-129.

Cove, J. F., & Walsh, B. C. (1988). Online text retrieval via browsing. *Information Processing & Management*, **24**, 31-37.

Cuomo, D. L., & Sharit, J. (1989). A study of human performance in computer-aided architectural design. *International Journal of Human–Computer Interaction*, **1**, 69-108.

John, B. E., & Vera, A. H. (1992). A GOMS analysis of a graphic, machine-paced, highly interactive task. In *CHI '92 Conference Proceedings: Striking a Balance* (pp. 251-258). New York: ACM.

Mohageg, M. F. (1992). The influence of hypertext linking structures on the efficiency of information retrieval. *Human Factors*, **34**, 351-367.

Olson, G. M., Herbsleb, J. D., & Reuter, H. H. (1994). Characterizing the sequential structure of interactive behaviors through statistical and grammatical techniques. *Human–Computer Interaction*, **9**, 427-472.

Rauterberg, M. (1993). AMME: An automatic mental model evaluation to analyse user behaviour traced in a finite, discrete state space. *Ergonomics*, **36**, 1369-1380.

Reed, W. M., & Geissler, S. F. (1995). Prior computer-related experiences and hypermedia cognition. *Computers in Human Behavior*, **11**, 581-600.

Salomon, G. (1995). A case study in interface design: The CHI '89 information kiosk. In R. M. Baecker, J. Grudin, W. A. S. Buxton, & S. Greenberg (Eds.), *Readings in human–computer interaction: Toward the year 2000* (pp. 25-34). San Francisco: Morgan Kaufmann.

Sanderson, P. M., & Fisher, C. (1994). Exploratory sequential data analysis: Foundations. *Human–Computer Interaction*, **9**, 251-317.

Sanderson, P. M., Iozzo, N., Buberel, J., & Au, I. (1995). The advising workbench: Participation-based development of a software environment to support student advising. In *Proceedings of the Human Factors and Ergonomics Society 39th Annual Meeting*, (pp. 1180-1184). Santa Monica, CA; HFES.

Sanderson, P. M., McNeese, M., & Zaff, B. (1994). Handling complex real-world data with two cognitive engineering tools: COGENT and MacSHAPA. *Behavior Research Methods, Instruments, & Computers*, **26**, 117-124.

Sanderson, P. M., Scott, J. P., Johnston, T., Mainzer, J., Watanabe, L. M., & James, J. M. (1994). MacSHAPA and the enterprise of exploratory sequential data analysis (ESDA). *International Journal of Human–Computer Studies*, **41**, 633-668.

Shneiderman, B. (1992). *Designing the user interface: Strategies for effective human–computer interaction*. Reading, MA: Addison-Wesley.

Tauscher, L., & Greenberg, S. (1997a). How people revisit web pages: Empirical findings and implications for the design of history systems. *International Journal of Human Computer Studies*, **47**, 97-138.

Tauscher, L., & Greenberg, S. (1997b, March). Revisitation patterns in World Wide Web navigation. In *Proceedings of the ACM SIGCHI'97 Conference on Human Factors in Computing Systems* (pp. 399-406). Atlanta: ACM.

Vortec, O. U., Edwards, M. B., & Manning, C. A. (1994). Sequences of actions for individual and teams of air traffic controllers. Special Issue: Exploratory sequential data analysis. *Human–Computer Interaction*, **9**, 319-343.

(Manuscript received December 12, 1996;
revision accepted for publication October 9, 1997.)