The Tangled Web We Wove: A Taskonomy of WWW Use

Michael D. Byrne[1], Bonnie E. John[2], Neil S. Wehrle[3], David C. Crow[4]

[1]Department of Psychology
[2]Human-Computer Interaction Institute
[3]School of Design
Carnegie Mellon University
Pittsburgh, PA 15213

[4]Trilogy Development Group
6034 West Courtyard Dr.
Austin, TX 78730

byrne@acm.org, bej@cs.cmu.edu, nsw+@andrew.cmu.edu, david.crow@acm.org

ABSTRACT

A prerequisite to the effective design of user interfaces is an understanding of the tasks for which that interface will actually be used. Surprisingly little task analysis has appeared for one of the most discussed and fastest-growing computer applications, browsing the World-Wide Web (WWW). Based on naturally-collected verbal protocol data, we present a taxonomy of tasks undertaken on the WWW. The data reveal that several previous claims about browsing behavior are questionable, and suggests that that widget-centered approaches to interface design and evaluation may be incomplete with respect to good user interfaces for the Web.

Keywords

World-Wide Web, task analysis, video protocols

INTRODUCTION

A great deal of public and research interest has been devoted to the World-Wide Web (or WWW) in recent years. Most of this research effort has focused on the technical aspects of the Web and the application of new Web-based technologies. Our concern, however, is with the usability of the Web, in particular the use of browsers. In order to perform any kind of sensible usability evaluation, be it empirical, analytical, or heuristic, it is necessary to first understand the tasks users engage in while browsing. That is, in order to determine which HCI techniques and/or approaches are most likely to aid in Web usability, it is first necessary to understand what is that users actually do with their time while using the Web.

While there has been valuable research on patterns of Web use (e.g. navigation patterns [1, 7]), these "click-studies" provide little information about the task contexts in which the users' actions occurred. For example, a click study can provide information about how often pages are visited and links are traversed, but not the tasks in which users were engaged while doing so. The focus of the present research is to gain a clearer understanding of the tasks users engage in while browsing the Web and the time spent doing those tasks. This will enable rating the relative importance of various interface analysis methods—there is little to be gained by analyzing tasks which users rarely perform, or which cost users very little time.

Further, we wanted to observe the tasks users normally perform in their daily Web use, rather than giving them artificial tasks. While observing users doing specific tasks can be useful, it is possible that the task or tasks used in the study might not reflect the tasks that users do when left to make their own decisions about how their time is allocated. Other naturalistic studies of user behavior [e.g. 2] have found that undirected user behavior is much more complex and interleaved than directed behavior.

THE STUDY

Participants were asked to browse the WWW as they would on a normal work day and provide verbal protocols describing what they were doing as they browsed. A video camera was set up to record both the protocol and the user's screen as they browsed, for one day of browsing. The participants were eight volunteers from the university community, all of whom were experienced Web users. We attempted to collect a reasonable cross-section of users and had faculty, students, secretarial staff, and research staff included in the sample. In order to encourage the participants to engage in the kind of browsing they would do normally, they were videotaped in their offices (or home in the case of one student participant) using their normal workstation. While it is possible that participants altered their browsing patterns due to the presence of the video camera, we explicitly discouraged this. A variety of platforms and browsers were used, with Netscape Navigator running on a Power Macintosh being the most frequent choice. Approximately 5.75 hours of videotape was

Appears in *Human Factors in Computing Systems: Proceedings of CHI 99,* pp. 544–551. New York: Addison Wesley.

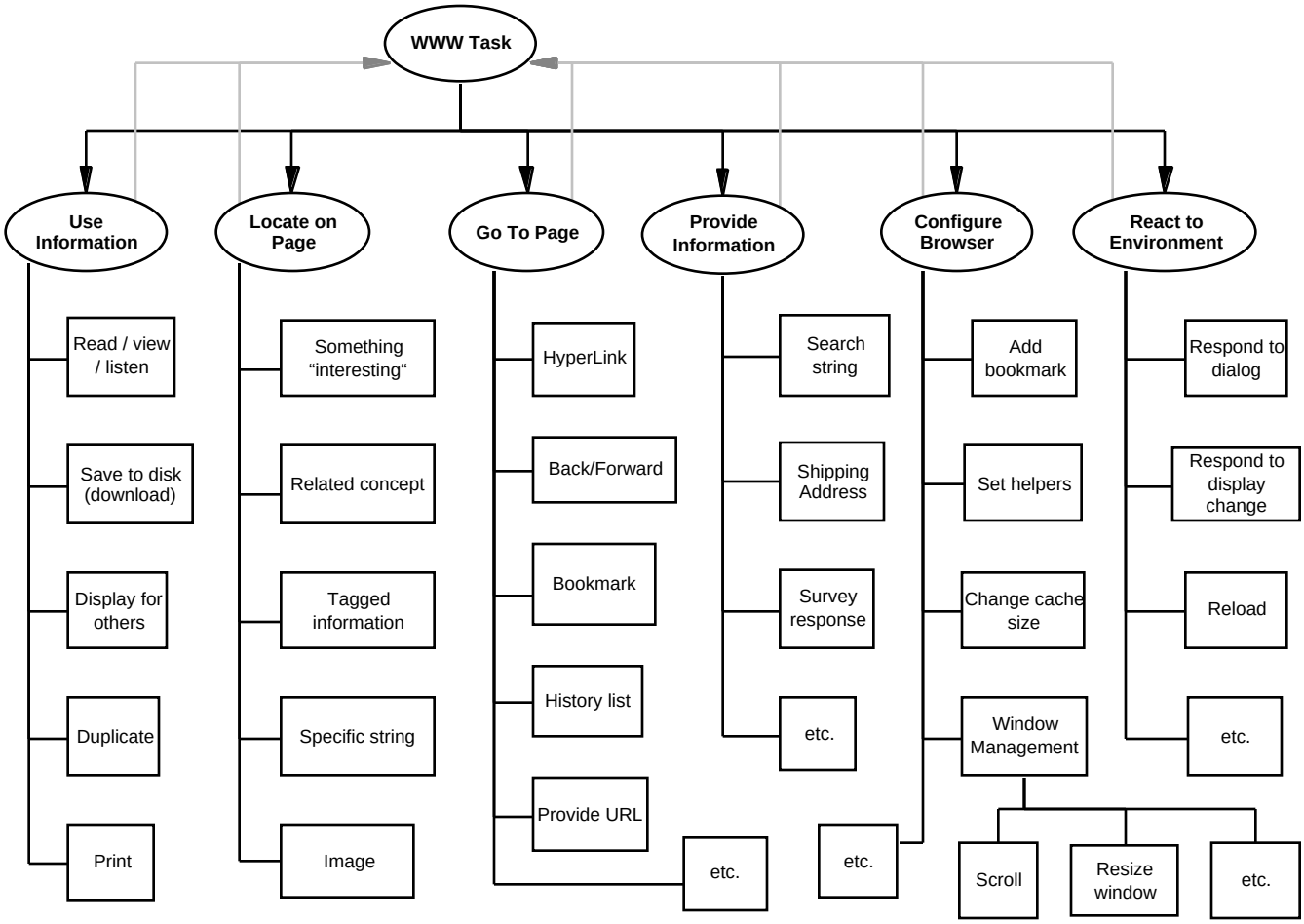

Figure 1. Taxonomy of WWW user tasks

collected and analyzed, of which 5 hours was WWW browsing. The amount of data generated and analyzed for each participant ranged from approximately 15 minutes to over 74 minutes. One of two analysts coded each videotape.

TASKONOMY

We constructed a taxonomy of tasks (a "taskonomy", shown in Figure 1) through a combination of our prior research into information-finding, analysis of the the capabilities of the Web and Web browsers, personal introspection, and observing our first three users. We observed six general classes of Web tasks: Use Information, Locate on Page, Go To Page, Provide Information, Configure Browser, and React to Environment.

As an example of how we constructed this taskonomy, consider the task Locate on Page. Our prior research in the use of a textual on-line help system [6] indicated that when attempting to locate information, users could search for a specific string or a related concept. Since web pages also include images, we included a category for locating an image. Introspecting, we added a category for locating something "interesting." Finally, we observed one of our first three users searching for information that would be tagged with a specific string (explained below).

While we had to add one subcategory to fully cover the five additional protocols (Go To Page using the history list that pops up when the Back button is depressed in newer versions of Netscape), the six main categories remained unchanged. That is, the taskonomy was virtually unchanged after analyzing the first three users, and then successfully covered the remaining five protocols (only two events out of 892 new events fell outside the original taskonomy, both of which were the unforeseen type of Goto). More details about the coding scheme can be found at: <http://act.psy.cmu.edu/ACT/people/byrne/webtask/guide.html>.

We chose to code the protocols at this level because we believed this to be a useful first pass at understanding how users allocated their time in terms of tasks and behaviors. Other levels of abstraction are both possible and potentially useful and should be pursued in follow-up research.

Use Information

Use Information describes any activity (or series of activities) in which the user was attempting to use a piece of information from the WWW. The Use Information subcategories were based on our observations of what the first three users did with the information they obtained from the Web. Information on the Web can serve a variety of purposes: it can be read, listened to, viewed or watched (e.g. images, animations, layouts), duplicated (e.g. copy and paste), downloaded to a local disk, displayed for others, printed. Most activities done while browsing the Web are in service of a Use Information task.

A Use Information task began whenever the user initiated a new activity with the goal of making use of some piece of information. The task was considered complete whenever the desired use had been made or the user explicitly gave up. For example, if a user had a Use Information(print) task, it would be considered started as soon as the user did anything to find that piece of information and ended when the Print dialog had finished.

Locate

Frequently, using a piece of information or going to a URL requires finding that information or link on a Web page, which typically requires some visual search. We called these activities Locate tasks. Users could search for a specific word, which we called Locate String search. Users also searched for particular images (e.g. graphic links), coded as Locate Image. They could be looking for something not necessarily a particular word or image but anything related to a concept (e.g. "I'm looking for 'photography' or 'cameras' or something like that"), which we termed Locate Related. Another class of searches can be best described as Locate Interesting, in which a user is seeking no specific word or concept, but is simply looking for something that might catch their interest. The most difficult kind of search to explain, but one which was observed, was what we called Locate Tagged. When a user was looking for a particular piece of information and did not know what it was that they were looking for, but knew some tag that would identify it as the piece of information they wanted, it was coded as Locate Tagged. For example, one user wanted to know the resolution of a printer he was considering purchasing. He did not know the number of dots per inch for the printer, but knew that the number he wanted would be tagged with something like "resolution" or "DPI" or the like. This is distinct from Locate Related in that it is not the concept that the user is searching for, but a value pointed to by some tag matching a concept or word.

Locate tasks were coded as beginning either as soon as the relevant page was visually available to be searched (usually after loading) or the user's protocol gave evidence they were searching. Locates were typically considered complete when the user explicitly indicated they had found the item or when a mouse movement was made to the target item. Alternately, a Locate task could be coded as finished when the user gave up, either explicitly or by navigating to a page not linked to the current one (e.g. clicking "back").

Note that our use of Locate does not mean "Locate a page on the Web" but is more like "Locate an item on a page." Locating a particular page on the WWW can require one simple GoTo (e.g. if the page is bookmarked) or a series of Locates and GoTos (e.g. following a series of hyperlinks).

Go To

Any activity which caused the browser to display a particular URL we considered a Go To. Most browsers support a wide array of ways in which a browser can be directed to a URL, including the back/forward button, bookmarks, hyperlinks, typing in a URL, history menus, a Home button, and others. Our subcategories of GoTo were based on an analysis of the methods supported by the browsers used. Go To tasks are typically fairly rapid, but they can be time-consuming, such as when typing a long URL is involved, the network response is slow, or the browser takes a long time to render the page.

GoTos were coded as starting as soon as the command that caused the browser to change pages was initiated (for example, as soon as users pointed at the URL field in the toolbar) and were coded as complete as soon as the destination page was displayed with enough content that it was possible for the user to interact with it.

Provide Information

Users not only use the Web to get information, but to send it as well. They provide product selections, authentication information, shipping addresses, search criteria, and so on. These activities were all classified as Provide Info tasks. Provide info tasks were coded as beginning as soon as the user began the mouse move or typing that supplied the information (usually in a form) and ended as soon as there was confirmation that the information had been received (typically by the display of the response page). There are a potentially infinite number of kinds of information users could be providing, so we made no strong commitments to particular subcategorization.

Configure

There is a wide variety of browser state information that is user-configurable, and changing the state of the browser (other than which URL to view) we termed Configure tasks. The kinds of Configure tasks available to the user depended on the number of user-configurable options provided by the browsing software. The most obvious aspect of a browser that users can (and frequently do) change is the state of the window or windows. Users can change the size, location,

order, scroll position, and number of browser windows (among other things). There are other things about the browser that users can change, however, such as bookmarks and assorted other preferences like cache size.

Configure tasks were coded as beginning as soon as a mouse move or keystroke involved in changing whatever aspect of the browser state change began, and ended whenever the final state at the end of the task had been reached.

React

While most browsing activities are user-driven rather than browser-driven, there are times when the browser demands something of the user. We classified these situations as React tasks. These are typically in the form of a responding to a dialog box (e.g. where to save a file, can't find a DNS entry, etc.), but can take other forms. One common other form is the use of the Reload button—the user is reacting to some problem with a page display. Many React tasks have Configure tasks as subgoals. For example, when a page is loaded that has a fixed-width table in it that is wider than the current window, this often causes the user to react with a Configure task to change the window width.

React tasks were coded as beginning whenever a dialog or extraneous window appeared, or whenever the mouse movement to the control (e.g. the "Reload" button) required to react to the situation started. React tasks were considered complete when the dialog or window had been dismissed or when the action initiated by the React task completed (e.g. the page had reloaded).

Subtask Sequencing

In general, these tasks cascaded a great deal and had subtasks. For example, one user wanted to download a paper written by a colleague. Thus, the top-level goal was to Use Information (download). The user decided to use a search engine to find the colleague's page, which generated a Go To Page (bookmark) task to get to the engine. Once there, the user engaged in a Provide Information task to tell the search engine what to look for, followed by a Locate on Page task to find the appropriate link. This was followed by another Go To Page (hyperlink) task to the relevant page, then another Locate on Page to find a link to the paper itself. The entire episode counts as a single Use Information task, with several subtasks performed in sequence:

UseInfo(download)
- GoTo(bookmark)
- ProvideInfo(search criterion)
- Locate (related)
- GoTo(hyperLink)
- Locate(related)

This episode generated six task instances. Note that the duration of the top-level Use Information task would include the time taken for all the subtasks—the task covers the time beginning when the user begins their attempt to download the file until the download is complete.

Use Information tasks are not the only kinds of tasks that can have subtasks. In fact, all of the task types can (and did) have subtasks. Locate tasks often have Configure subtasks, such as scrolling the window. Provide Information tasks can generate Use Information tasks (often Duplicate) to provide form fill-in values. Configure tasks rarely have subtasks, but do occasionally (such as a Use Information subtask to determine what it is that a particular preference does). React tasks, as previously mentioned, often have Configure subtasks. Furthermore, tasks at *any* level could generate subtasks—this did not occur only at the top level. Since each task type can generate one or more of the other types as a subtask at any level, there is very little *a priori* hierarchy that can be imposed on the taxonomy.

RESULTS

We originally expected the tasks to form a hierarchy, but we discovered that any one of these general classes of tasks can generate any other type of task as a subgoal, thus, the "hierarchy" is tangled and nearly flat—not so much a strict hierarchy.

Top-level Categories

We found that our six top-level tasks (Use Information, Go To, Locate, Provide Information, Configure, and React) did an excellent job of capturing the types of behavior engaged in by our users. All episodes in the protocols fit into these six categories. Not surprisingly, some tasks were more frequent than other and some took both more total time and more time per task. Results for the top-level tasks are presented in Figures 2 and 3.

Note that the most common (in terms of raw number) class of events are actually Configure events. Users often needed to scroll the page in the window, and each time a user scrolled, this created a Configure task (frequently as a subtask of Locate). This finding conflicts with other reports that users are reluctant to scroll [4].

However, in terms of total time it is clear that the tasks that dominated our users' browsing was Use Information. This is hardly surprising since the widespread dissemination of information cheaply and quickly was the original purpose of the WWW. The next most time-consuming activity was Locate. Locate often had Configure as a subtask, because users often needed to scroll to locate the item for which they were looking.

The GoTo class of tasks occupied our users for a fair amount of time as well. What is striking about this number is that most of the time taken to perform a GoTo is time

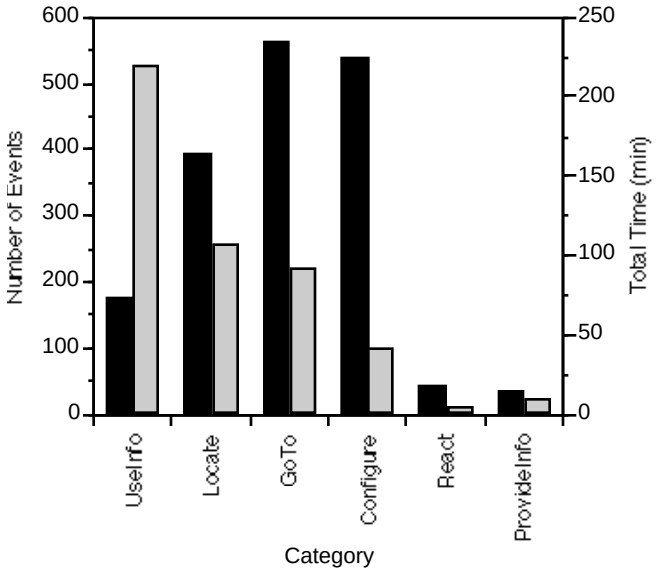

Figure 2. Number of events (black bars) and total time in minutes (gray bars) for each type of top-level task

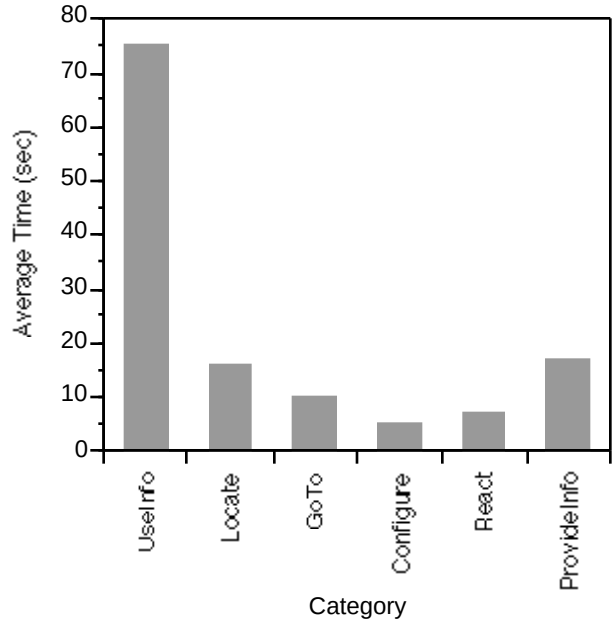

Figure 3. Average time (in seconds) spent on each top-level task category

waiting for the page to load. Over 50% of that GoTo time is, in fact, simply users waiting for page loads.

Note also that the time spent sums to more total time than we had videotape. This is because tasks nest within one another, as previously discussed.

Average times are revealing as well. The average Use Information task took our users over a minute. There are two reasons these tasks are so much longer than the other task types. First, these tasks include reading. Despite claims to the contrary [5], some users actually do spend time reading, rather than merely scanning, Web pages. Second, Use Information tasks typically had more subtasks. The information to be used often had to be found using a series of Locates and Gotos.

Provide Info tasks were the next longest on average. These tasks typically involve at least some typing, and can sometimes require a large number of clicks and keystrokes. Provide Info tasks also have system response time included, as a Provide Info task was not scored as being completed until the response page was displayed.

Several of the task categories have interesting divisions. In particular, Use Information, Locate, and GoTo have useful subcategories. Provide Informations and Reacts were neither especially frequent or particularly time-consuming, so those will not be considered in greater detail. Configure tasks were frequent, but the bulk of them (477 out of 538, taking up 33 minutes total time) were scrolling.

Use Information

Use Information was the dominant category in terms of both total and average time. While this is hardly surprising, this does raise the question of what is it that users want to do with the information they get from the Web—why are they browsing in the first place.

The breakdown of Use Information tasks by subcategories is presented in Figures 4 and 5, again by total number, of tasks observed, total time spent, and average time per task. These data are clearly dominated by Read tasks in terms of

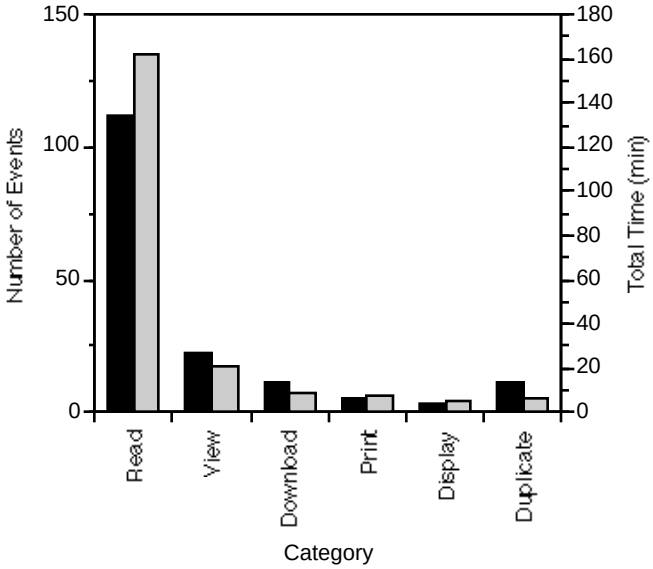

Figure 4. Number of events (black bars) and total time in minutes (gray bars) for each type of Use Information task

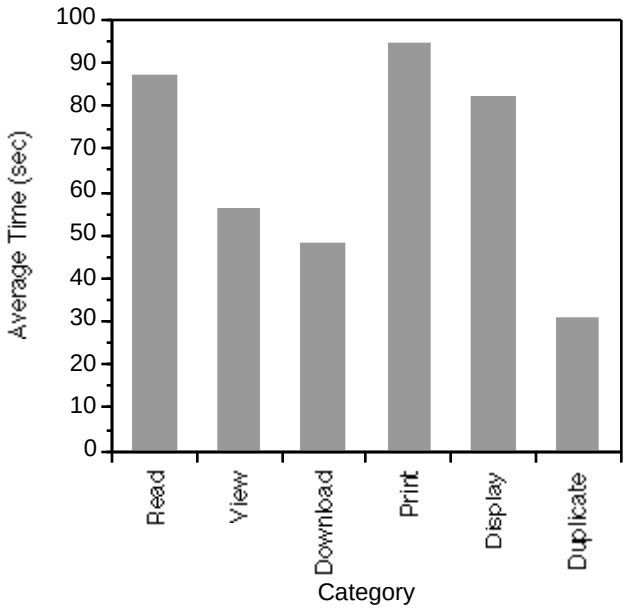

Figure 5. Average time spent on each type of Use Information task (in seconds)

number and total time. With respect to average time, only the Read subcategory has sufficient number of events to provide a stable estimate.

Locate

When looking for an item on a page, there are various levels of specificity users have in mind. These were categorized into one of four types: String (or Image), Interesting, Related, and Tagged. Breakdowns by type for number of observations, total time, and average time are presented in Figures 6 and 7.

It is noteworthy that the Tagged search was the least common in terms of both frequency and total time. Tagged searches tended to occur only at the leaf nodes of multi-page searches, which were guided primarily by other types of search.

String searches, which are the most specific type of search, were the most rapid on average. Searching for specific strings was not as common as the more general types of searches and tended to be a result of users looking for a specific text-based anchor which they knew was already present on a page. Thus, these searches were probably aided both by the users' spatial memory for the page and the fact that revisitation of links is common. Revisitation often meant the sought-after link had been visited recently, and most browsers display recently-visited text in a different color than non-anchor, non-visited text. (Visual search for a distinct color is typically very rapid. [8])

The relative frequency of searches for something Interesting is also noteworthy. In typical laboratory studies of WWW use [e.g. 3], users are given specific search goals. However, we intentionally did not give users such targets, which likely resulted in a greater number of less-directed searches for things that just appeared "interesting" to the user.

GoTo

While most Web browsers support a wide array of methods for changing the URL being viewed, actual usage patterns suggest that users tend to rely mostly on a small number of methods. Figures 8 and 9 present the number, total time, and average time of the GoTo tasks performed by our users. GoTo task times also include time that most users would rather not spend, time waiting for pages to load. (The GoTo operation is not considered complete until the new URL is

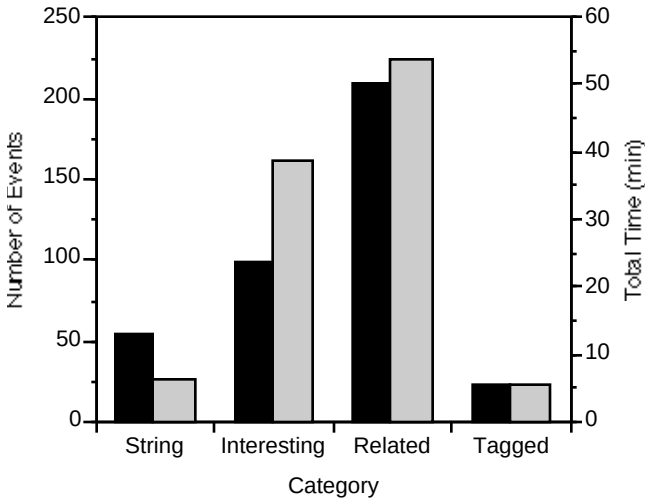

Figure 6. Number of events (black bars) and total time in minutes (gray bars) for each type of Locate task

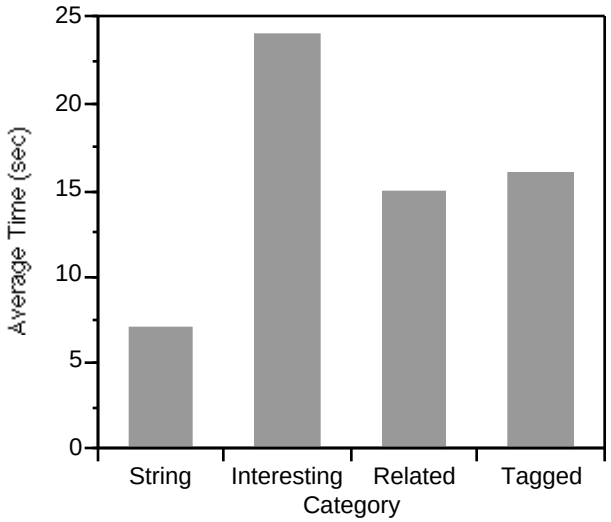

Figure 7. Average time (in seconds) spent on each type of Locate task

displayed.) Overall, our users spent over 47 minutes waiting (of the 5 hours total time spent browsing), and nearly all of this waiting was time spent waiting for pages to load. This number is probably a significant underestimation of the proportion of time average users spend waiting, as all but one of our participants had high-speed ethernet-based network connections (the Carnegie Mellon campus is served by a T3). Furthermore, one of the most experienced users—also an experienced programmer—had an aggressive multi-window browsing strategy clearly motivated by the desire to get something else done while waiting for slow pages to load. Thus, this is likely a very conservative estimate of the amount of time wasted by waiting.

Following hyperlinks was the most common way to change the URL being viewed, with the next most common method being the "back" button. Notice that, on average, the "back" button is much faster than following hyperlinks. This is almost certainly due to the fact that the page accessed by the "back" button is usually cached by the browser. This suggests that the gains that could by had by better caching algorithms and higher network bandwidth may be considerable.

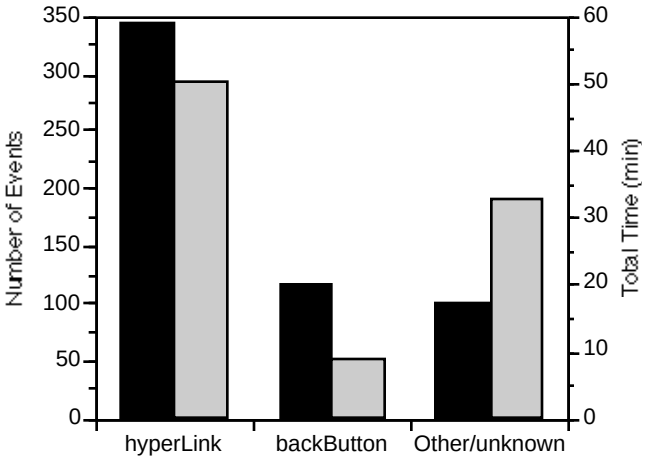

Figure 8. Number of events (black bars) and total time in minutes (gray bars) for each type of Goto task

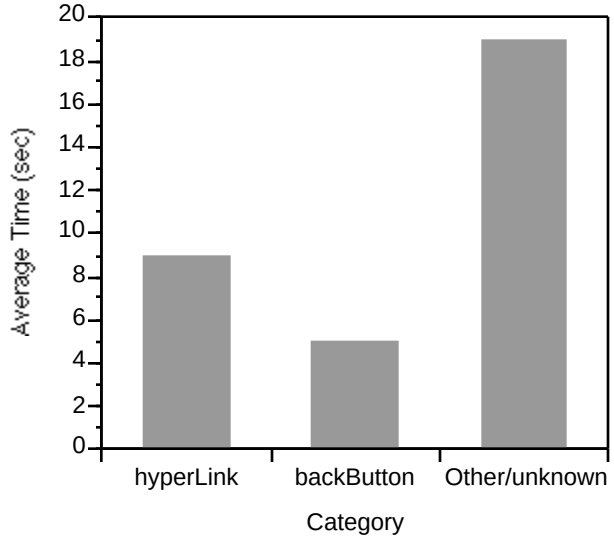

Figure 9. Average time (in seconds) spent on each type of Goto task

Most of the "other" GoTo's involve typing in a URL, and thus these types of GoTo's require a great deal more time. URLs appear to be particularly difficult for users to type because of the unusual punctuation and preponderance of nonwords. Other types of navigation, such as the use of history menus, were quite infrequent. This may be because the user interfaces of most history systems are less than optimally matched to the way users think about Web navigation [7]. Alternatively, it may simply be that users rarely back up more than a page or two at a time.

DISCUSSION

Implications for WWW Browser Design

What these data suggest is that time spent worrying about things like button layouts and history menus may not have much impact on normal Web browsing. Users do not spend a great deal of time interacting with the GUI widgets of their browsers relative to the amount of time they spend engaged in things like reading, visual search, and waiting. On the other hand, this may well be because the functionality or interface provided to users to support their tasks are poor. It is not clear whether users would spend more time interacting with GUI widgets if they were better designed. For example, we observed little use of the history system. This may be because the history system is poorly designed, as suggested by [7]. However, it might also be the case that users would make little use of history systems no matter how implemented.

An obvious case where widget design could make a difference is scrolling. Users spend a great deal of time scrolling (approximately 40 minutes in our 5-hour sample was spent scrolling), and advances which reduce the latency of scroll operations (such as wheeled mice like the Microsoft IntelliMouse) have the potential to save users considerable time. Whether such devices actually do save users time is still an open question, but the potential is clearly there.

Because users spend so much time waiting, improving the performance of the caching and rendering algorithms in browsers should clearly be a high priority as it could potentially save users considerable time. Improving system performance to reduce waiting time is hardly a new suggestion in HCI; however, this appears particularly salient in the case of the Web. Even pages that clearly

should have been cached (e.g. those loaded by the "back" button) took an average of approximately five seconds to be fully loaded and rendered.

Implications for Page Design

Users are willing to scroll through and read long passages, despite claims to the contrary [5] based on "classic directed tasks." In undirected situations, if users find essays or articles that are of interest to them, they do read them. This suggests that long, textual Web pages are not necessarily a bad idea but should be designed for readability. On the other hand, users do spend a great deal of time searching pages for items related to a target concept, and there may be tradeoffs between readability and "scanability" of a page. These data suggest that the tradeoffs should be carefully evaluated. For some pages it may indeed be worthwhile to sacrifice readability for searchability—but for other pages this may only distract and annoy users.

Some of the initial decisions made in designing browsers defaults were excellent. For example, most Web browsers underline and color links, which can be a tremendous aid to the visual search process—visual search for a target that can be discriminated on the basis of color alone are typically very rapid [8]. However, HTML now allows designers to override this and make link colors different than the defaults. Most page design guidelines advise against this, and our data is in agreement with this guideline—anything that slows visual search is likely to cost users time.

Overall, the clearest point that these data make is that WWW browsing is a complex mixture of a variety of behaviors, and any attempt to improve the interface to the Web needs to be sensitive to this variety.

FUTURE WORK

Although the summary data presented here give a high-level view of what people are doing when they browse the Web, the verbal protocols hold a wealth of detail. The current analyses are clearly limited. Future analyses can and should include analyses at higher levels of abstraction (e.g. strategies and patterns of behaviors), and analysis of the contents of the tasks in which users engage rather than just the behaviors. For example, the current analyses did not consider whether or not a given Locate was successful or not, or what it was that was being Located, but merely that the user was trying to locate something on a page. Integration with click studies, which can provide more detailed information about the exact contents of the Web pages being browsed (e.g. "what percentage of the links on a given page are visited?"), is also likely to provide further insight into browsing behavior.

Furthermore, the sample of users and environments is also clearly limited. A wider sampling of users, browsers, and network environments would not only improve the generality of the results, but allow for more careful consideration of individual differences. We expect that more detailed analysis of naturalistic studies such as this one will provide considerable design guidance.

ACKNOWLEDGEMENTS

This research was sponsored by the National Science Foundation (NSF), Award #IRI-9457628 and by and by the National Institute for Mental Health (NIMH), fellowship #2732-MH19102. It was also supported by generous contributions from the Xerox corporation. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Xerox, the NSF, the NIMH, the U.S. Government, or any other organization.

REFERENCES

1. Catledge, L. D., & Pitkow, J. E. (1995). Characterizing browsing strategies in the World-Wide Web. In Proceedings of the Third International World Wide Web Conference, http://www.igd.fhg.de/www/www95/papers/, Darmstadt, Germany.

2. Cypher, A. (1986) The structure of users' activities. In Norman, D.A. and Draper, S.W., (eds.) *User Centered System Design*, pp. 243-263.

3. Morkes, J., & Nielsen, J. (1997). Concise, SCANNABLE, and Objective: How to Write for the Web. http://www.useit.com/papers/webwriting/writing.html

4. Nielsen, J. (1996). Top Ten Mistakes in Web Design. http://www.sun.com/columns/alertbox/9605.html

5. Nielsen, J. (1997). How Users Read on the Web. http://www.useit.com/alertbox/9710a.html

6. Peck, V. A. & John, B. E. (1992) Browser-Soar: A cognitive model of a highly interactive task. In Human Factors in Computing Systems: Proceedings of CHI 92 (pp. 165-172). New York: ACM Press.

7. Tauscher, L., & Greenberg, S. (1997). Revisitation patterns in World Wide Web navigation. In *Human Factors in Computing Systems: Proceedings of CHI 97* (pp. 399–406). New York: ACM Press.

8. Triesman, A., & Gelade, G. (1980). A feature-integration theory of attention. *Cognitive Psychology, 12,* 97–136.