CHAPTER

Automated data collection methods 12

12.1 INTRODUCTION

Data are the building blocks of research. As the recorded output of research efforts, data are the raw materials that must be processed, analyzed, and interpreted to provide answers to research questions. Data collection is therefore a critical phase in any research effort.

Data collection is also often one of the most challenging aspects of research. Timing user task completion with a stopwatch, furiously writing notes describing user interactions with software systems, coding notes from ethnographic observations, and many other tasks are laborious, time consuming, and often—as a result—error-prone.

Fortunately, human-computer interaction (HCI) researchers can use the computers that are the subject of our research as powerful data collection tools. Software tools can be used to collect vast amounts of user interaction data, often with little or no direct effort on the part of the researcher administering the study. Interaction logging software tracking keystrokes and mouse clicks, special-purpose instrumented software designed to track use of specific features in tools, web site access logs, and home-grown customized tools for tracking what users do and when can simplify data collection, increase consistency, and decrease error.

Approaches to automated data collection can generally be placed on a spectrum of ease of use and flexibility (Figure 12.1). Existing software tools such as website access log analyzers can often be easily used or adapted for research purposes, but capabilities might be limited. System observation and logging software may be somewhat more powerful, but installation and configuration issues can be challenging. Custom-built or modified software can be crafted to meet the precise research needs, but the development effort can be substantial.

All of these automated methods for computerized data collection are capable of producing voluminous data sets. This can pose a substantial problem for researchers: while generating data is easy, deciding which data to collect and how best to analyze that data can be challenging. Although many projects involving automated data collection (including data collected from human subjects—see Chapter 13) follow a familiar arc of planning, data collection, cleaning, analysis, and iteration to refine methods and techniques, details vary between projects due to technological differences in data acquisition methods and analytic differences associated with varying research questions. Some of these issues will be discussed here and in Chapter 13,

Research Methods in Human-Computer Interaction. http://dx.doi.org/10.1016/B978-0-12-805390-4.00012-1

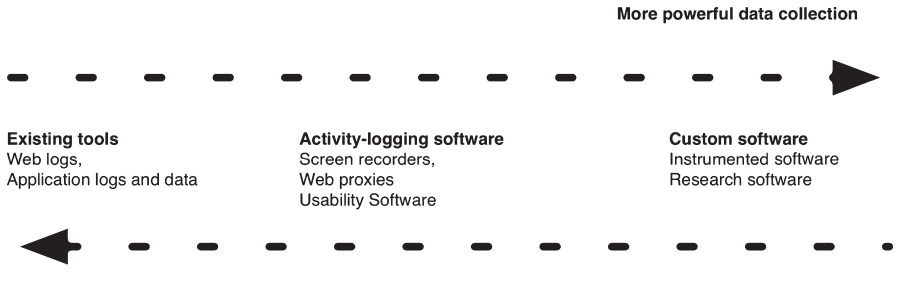


FIGURE 12.1

Computerized data collection systems present a trade-off between power and ease of implementation and use.

but you may need to digger deeper in the literature to find similar work for more specific guidance on appropriate data granularity, data cleaning, and analytic techniques.

In this chapter, we focus on log and data capture. This is certainly not the full story of the use of automated data capture in HCI. Newer technologies such as smartphones and a huge variety of inexpensive sensors provide rich troves of data suitable for understanding how we interact with computers in a wide variety of environments. These applications will be discussed in Chapter 13 on Human Data collection and Chapter 14 on online and ubiquitous HCI research.

12.2 EXISTING TOOLS

Many commonly used software tools collect and store data that can be used in HCI research. These tools have the obvious appeal of relative simplicity: although some effort may be required for analysis, data collection tools may be readily available. For some widely analyzed data sources—such as web server logs—commercial and freely available tools can provide substantial assistance in interpretation.

These advantages do not come without a cost. Using unmodified, commodity software is likely to limit you to data that is collected by default. If your research questions require additional data, you may be out of luck. This is often not a real barrier—many successful research projects have been based on analysis of data from available software. A sound strategy might be to start with these tools, pushing them to see how far they can take your research efforts and moving toward more complex measures if needed.

12.2.1 WEB LOGS

Web servers, email servers, and database servers all generate log files that store records of requests and activity. As a sequential listing of all of the requests made to a server, a log file provides a record of how the server has been used and when. This detailed information can be useful for evaluating system performance, debugging problems, and recovering from crashes.

Web logs have also proven to be a potent tool in HCI research. Given a website and a log file, researchers can often analyze entries to determine where users went and when. When combined with an understanding of the architecture of a site, this information can be used to assess the usability of a site. Timing data in web logs also presents opportunities for empirical studies. Although log data is not perfect, and often presents analytic challenges, appropriate analysis can often yield useful insights.

12.2.1.1 Web log contents

Although web servers can be configured to store a variety of data fields along with each request, most log files store data that can identify a request and its source. Some log files also contain fields that are generally less useful. The useful data includes:

- *Host*: The Internet protocol address of the remote computer that made the request. As many people access the Internet via networks that use firewalls or proxy hosts that forward requests from internal machines, a host address might not correspond directly to a specific user's computer.
- *Timestamp*: When the request occurred, usually including a date and a time code. Times may be given relative to Greenwich Mean Time.
- *Request:* The HTTP request sent by the client to the server. The request has several fields that may be of interest:
  - *HTTP Method*: The type of request being made—usually "GET" or "POST" (Fielding and Reschke, 2014).
  - *Resource*: The file, script, or other resource requested from the server.
  - *Protocol:* The version of the HTTP protocol used.
- *Status Code:* A numeric response from the server, indicating success (200–299), redirection (300–399), client error (400–499), or server error (500–599) (Fielding and Reschke, 2014).

Several other potentially useful fields may be available:

- *Size*: The size—in number of bytes—of the item returned to the client.
- *Referrer*: The web page that "referred" the client to the requested resource. If a user on http://yourhost/index.html clicks on the "search.html" link, the request indicates that "http://yourhost/index" was the referrer. Some requests, such as those that come via an address typed in to a browser, do not arrive via a link and have a dash ("-") in the referrer field.
- *User Agent:* The make and model of the web browser that made the request. As this is self-reported, it may or may not be accurate.

Figures 12.2 and 12.3 give some example log entries.

Most web servers use the common log format (World Wide Web Consortium, 1995) or similar formats as the basis for formatting log files. Customization facilities provided by most web services allow for the inclusion of specific fields. This can be very useful for adapting your logs to fit the needs of each project. If you are running a study involving users who are particularly sensitive to privacy concerns, you might configure your server to remove the client IP number from the log files. Similar

```
10.55.10.14 - - [13/Jul/2007:13:42:10 -0400] "GET /homepage/classes/spring07/686/index.html HTTP/1.1" 200 8623

10.55.10.14 - - [13/Jul/2007:13:48:32 -0400] "GET /homepage/classes/spring07/686/schedule.html HTTP/1.1" 200 16095

10.55.10.14 - - [13/Jul/2007:13:48:33 -0400] "GET /homepage/classes/spring07/686/readings.html HTTP/1.1" 200 14652
```

FIGURE 12.2

Log file entries, containing host IP address, timestamp, request, status code, and number of bytes.

```
10.55.10.14 %t "GET /homepage/classes/spring07/686/readings.html HTTP/1.1" 200 14652 "http://10.55.10.128/homepage/classes/spring07/686/schedule.html" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8) Gecko/20051202 Fedora/1.5-0.fc4 Firefox/1.5"
```

FIGURE 12.3

A detailed version of the last entry from Figure 12.2, including the referrer and the user agent.

changes can be made regarding the recording of the referrer, the user agent, or other fields. For many studies, it may be useful to create a special-purpose log in parallel with a traditional access log. The customized log file provides the information needed for your study, without interfering with access logs that might be used for ongoing website maintenance. Customized log file formats may require customization of the web server software or of the log analysis tools, but this is generally not hard to do.

Most web servers generate error logs in addition to access logs. The list of requests that generated server errors can be useful for identifying problems with a site design, such as links to nonexistent pages or resources. Check your server documentation for details.

As web logs can become quite voluminous, proper care and handling is very important. Numerous software tools extract information from log files for static reports or interactive analysis: several approaches to this analysis are described in this chapter.

Logs from publicly accessible sites may include regular and repeated visits from web robots, tools used by search engines and other tools to retrieve web pages, follow links, and analyze web content. Before using the logs of your publicly accessible site for research purposes, you might consider using the robot exclusion protocol (Koster, 2007) to discourage these automated tools. This protocol is very straightforward: all you need to do is to place one simple file in the root directory of your server. Polite bots will not make further requests once they see this file. As a result, the proportion of your log entries generated by these crawlers will be reduced, leaving you with more of the good stuff—visits from human users. As this step may have the (possibly undesirable) effect of reducing your site's visibility to search engines, you may wish to exclude robots for short periods of time while you collect data. Once your data collection is complete, you can disable your robot exclusion measures, thus allowing search engines to index your site and maintain your visibility.

Note that web requests—and therefore web logs—are not limited solely to recording clicks in web browsers. Many web sites provide Application Programming

Interfaces (APIs), often following the Representational State Transfer (REST) conventions (Fielding and Taylor, 2002). Essentially, REST (and similar) APIs define structured web requests to return data suitable for extraction and manipulation by third-party web sites and other programs. As these APIs can be used to provide data to mobile apps and stand-alone desktop programs—as well as web pages—any such accesses will be included in web logs, and can therefore be analyzed to track usage patterns.

12.2.1.2 Web usability/design research

By telling us which pages were accessed and when, web access logs can provide valuable information for usability evaluations and understanding of usage patterns. Relatively simple page access counts tell us which pages are accessed frequently, and which are not. When coupled with an understanding of page layout information, this can help identify opportunities for improving usability. Aggregate counts of timestamps, referrers, and user agents can be used to understand when a site is being used, how people are getting to links within a site (external referrers are particularly interesting in this regard), and which browsers they are using—all potentially useful information in the context of evaluating a site design. Interactive visualizations of this data at multiple granularities—particularly when coordinated with views of the site—can provide guidance for improving site design (Hochheiser and Shneiderman, 2001). For example, if important areas of the site are infrequently accessed, links might be moved to more prominent locations or be made more visually distinctive. Postmodification analysis can be used to evaluate the success (or lack thereof) of such measures.

Web access logs also provide the intriguing possibility of extracting information about the actions of specific users as they navigate a website. This information can be very useful for understanding which path users take through a site and where they might run into problems.

As each entry in an access log can contain an Internet address, a timestamp, the requested URL, the referring URL, and a user agent, we might be tempted to combine this information with knowledge of a site's layout to infer the path of specific users through a site. If we see that an access to “index.html” is soon followed by a request for “help.html,” with both requests originating from the same network address, we might think that these requests came from the same user. Matching user agents and an entry indicating that the referrer page for the “index.html” page was the “help.html” page might increase our confidence in this theory. Judiciously used web cookies can provide additional useful information.

Unfortunately, things are not necessarily that simple. Firewalls and other network address schemes may make requests that come from multiple users appear as if they all come from the same machine. Web browsers can easily be configured to provide misleading information for the user agent fields and referrer fields. Web redirects may create misleading requests, appearing as if a user intended to visit a site, when they had no such interest. Cookies may be disabled by some users and browsers.

Despite the problems, access logs can be used to generate useful models of user paths (Pirolli and Pitkow, 1999). Augmenting these records with additional information

including keywords extracted from visited web pages or URLs and page view time can provide increased accuracy in characterizing user sessions (Heer and Chi, 2002).

As a stand-alone tool, web log analysis is limited by a lack of contextual knowledge about user goals and actions. Even if we are able to extract individual user paths from log files, these paths do not tell us how the path taken relates to the user's goals. In some cases, we might be able to make educated guesses: a path consisting of repeated cycling between "help" and "search" pages is most likely an indication of a task not successfully completed. Other session paths may be more ambiguous: long intervals between page requests might indicate that the user was carefully reading web content, but they can also arise from distractions and other activity not related to the website under consideration. Additional information, such as direct observation through controlled studies or interviews, may be necessary to provide appropriate context (Hochheiser and Shneiderman, 2001).

Complex web applications can be designed to generate and store additional data that may be useful for understanding user activity. Database-driven websites can track views of various pages, along with other actions such as user comments, blog posts, or searches. Web applications that store this additional data are very similar to "instrumented" applications—programs designed to capture detailed records of user interactions and other relevant activities (Section 12.4.1).

The analysis of web log information presents some privacy challenges that must be handled appropriately. IP numbers that identify computers can be used to track web requests to a specific computer, which may be used by a single person. Analyses that track blog posts, comments, purchases, or other activity associated with a user login can also be used to collect a great deal of potentially sensitive information. Before collecting any such data, you should make sure that your websites have privacy policies and other information explaining the data that you are collecting and how you will use it. Additional steps that you might take to protect user privacy include taking careful control of the logs and other repositories of this data, reporting information only in aggregate form (instead of in a form that could identify individuals), and destroying the data when your analysis is complete. As these privacy questions may raise concerns regarding informed consent and appropriate treatment of research participants, some web log analyses might require approval from your institutional review board (see Chapter 15).

Web server logs have been the subject of many research studies over the years. The development of visualization tools to interpret these logs has been a recurring theme since the 1990s and continuing on to more recent work (Pirolli and Pitkow, 1999; Hochheiser and Shneiderman, 2001; Malik and Koh, 2016). Web search logs, particularly from search engines, have proven to be a particularly fruitful data source for studying how users conduct searches and interpret results (White, 2013; White and Hassan, 2014), particularly for specific tasks such as searching for medical information (White and Horvitz, 2009). For more on the use of web search logs to study user behavior, see Chapter 14. As is often the case, web log analysis studies often use multiple complementary datasets to confirm and complement log data. A study of the social network Google+ combined log analysis with surveys and interviews to

understand how users choose to share different types of content with different people on the network (Kairam et al., 2012).

12.2.1.3 Empirical studies

Empirical studies of task performance times require some means of capturing timing data. Although hand-held stopwatches can do this job admirably, software that measures and records elapsed times between starting events and task completion is usually more reliable and easier to work with. As described later in this chapter, this approach has been used extensively in special-purpose software built specifically for HCI studies.

For experimental tasks involving selections that can be presented as links on web pages, web servers and their logs present an ideal platform for gathering empirical task performance data. In this model a web server is run on the same machine that is used to perform the experimental tasks. This eliminates any delays associated with requesting materials over a network connection. The selection of a link from a starting page indicates the beginning of the task, with subsequent link selections indicating intermediate steps. Eventually, a link indicating successful task completion is selected. The elapsed interval between the selection of the start and completion links is the task completion time, with access records of intermediate requests indicating steps that were taken to complete the task and the elapsed time for each subtask.

This method is not without drawbacks. Extraction of the relevant information from logs may require manual interpretation or implementation of special-purpose log analysis software. Timestamps in server log files time events by the second, so this approach is not suitable for studies that require finer task-time resolution.

Web browser caches may cause additional problems. These caches store local copies of pages that have been recently accessed. If a user requests a page that is in the cache, the browser returns the copy that has been stored locally, instead of making a new request from the web server. This may cause problems if you are trying to track every user request, as requests for cached pages might not generate web server log entries.[1] You may want to turn off caching facilities before using a particular browser to run an experiment.

One helpful strategy for keeping data clean and clear is to start each session with an empty log file. After the session is complete, the file can be moved to a separate directory containing all of the data for the given subject. This simplifies analysis and prevents any problem associated with disentangling multiple participants from a longer log file.

In practice, these drawbacks usually do not create serious problems. The "Simultaneous vs. Sequential Menus" sidebar describes a study that used server logs to compare alternative web menu designs.

[1] Then again, they might. It all depends on the server configuration. However, it is best to be defensive about such matters: assume that they will not and take appropriate steps.

SIMULTANEOUS VS. SEQUENTIAL MENUS

Computer interfaces may be designed to present choices in a hierarchical, sequential manner, even if the items in the menu are not necessarily hierarchical. A restaurant selection tool for a city might allow users to select a neighborhood, followed by a price range, and finally a type of cuisine, but this is not the only possibility order. A simultaneous menu scheme would allow selections to be made in each of these three criteria at any time.

A comparison of the strictly sequential menu approach versus the simultaneous menu approach used a locally hosted web server to present alternative menu structures for the same underlying data set (Hochheiser and Shneiderman, 2000): US Census Bureau economic data for counties in the state

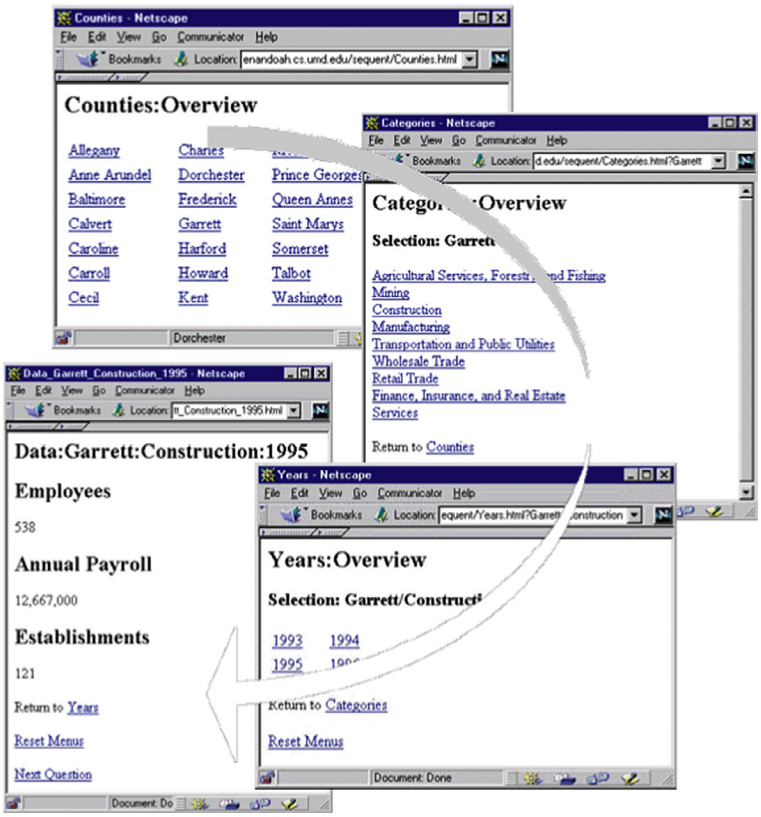


FIGURE 12.4

Sequential menus: users choose first from counties, then from categories, and finally from years, in order to get to a detail page.

From Hochheiser, H., Shneiderman, B., 2000. Performance benefits of simultaneous over sequential menus as task complexity increases. International Journal of Human–Computer Interaction 12 (2), 173–192.

of Maryland. A sequential menu allowed users to select a county, followed by a business category, and then a year (Figure 12.4). Simultaneous menus allowed for selection in any one of these criteria at any time, with detail displays showing data based on values for the three attributes selected (Figure 12.5). Each task in each menu structure began with the selection of a "start" link, and ended with the selection of a link that led to the correct answer.

Pages were presented on web pages, loaded onto a single machine, and accessed directly from that machine to minimize network delays. All menu selections were implemented as standard web links and captured in a standard log files. Logs were analyzed to extract the difference in time between the event signifying the start of the task and the corresponding event indicating task completion, using timestamps from log file entries.

This study found that sequential menus fared well for simple tasks, but simultaneous menus were preferable for more complex tasks (Hochheiser and Shneiderman, 2000).

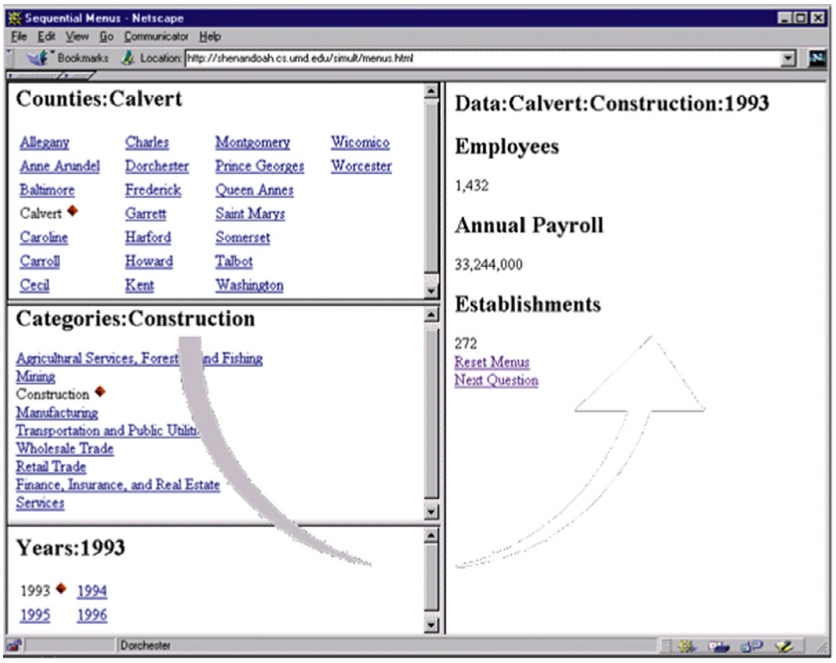


FIGURE 12.5

Simultaneous menus: once a user has selected a value for each of the three variables, details are shown on the right.

From Hochheiser, H., Shneiderman, B., 2000. Performance benefits of simultaneous over sequential menus as task complexity increases. International Journal of Human–Computer Interaction 12 (2), 173–192.

12.2.2 STORED APPLICATION DATA

As we use computers, we leave traces that provide valuable information about how we interact with applications and store and manage information. The tools that we use collect substantial data trails that implicitly and explicitly describe user activities. Examples include (but are not limited to):

- File systems: The files and folders that we create and use present a model of how we organize information. Do we separate work activities from home? Do we have many folders, each containing a small number of files, or only a few folders, each with many files?
- Graphical user interface (GUI) desktops: Some people have dozens of icons on their desktops, while others have only a few. Does this say anything about their organizational preferences?
- Email programs: Many people use an email "inbox" as a todo list, reminding them of tasks that must be completed. Some users make extensive use of filing and filtering capabilities, while others leave all messages in one folder.
- Web bookmarks can also be more or less organized.
- Social networking tools such as Facebook or LinkedIn provide detailed perspectives on how people connect to each other and why.

Each of these domains (and others) can be (and have been) studied to understand usage patterns and to potentially inform new designs. This research is a form of HCI archeology—digging through artifacts to understand complex behavior patterns.

There are attractive aspects to using existing data that is stored by tools that users work with on a daily basis. Interference with the user's work or habits is minimal. Users do not have to participate in experimental sessions to be part of the study and no training is necessary.

The generality of this approach is limited by the tools involved and the data that they collect. The example tools given earlier (file explorers, email clients, web browser bookmark tools, GUI desktops, etc.) all provide tools that can be used to manipulate and maintain organizations of information. As a result, they can be used to identify which structures exist, which categories' items might be placed in, etc. As more transient activities—such as selections of menu items—are generally not recorded, this approach is not well suited for the study of specific implementations. Instead, this approach to data analysis is best suited for the study of long-term patterns of ongoing tasks such as those described earlier.

As the analyses may involve exploration of potentially sensitive matters such as email messages, file system content, and web bookmarks, investigators using these approaches should be sensitive to privacy concerns. In addition to properly informing participants of the privacy risk (see Chapter 15), researchers should exercise discretion when examining potentially sensitive data. Investigations should be limited to only the data that is strictly necessary. An exploration of email communication patterns might reduce privacy risks by examining message headers, instead of message bodies. If this is not sufficient, anonymizing the content to simply indicates that A had an email conversation with B can provide further privacy protection.

A final limitation of this approach involves the challenge of extracting data. Converting these computational artifacts from their native form to a representation suitable for analysis can be challenging. You may need to write special-purpose software to extract data from these tools. In some cases, this may require interpreting (or reverse engineering) nonstandard file formats.

Although log files and implicitly stored data may prove useful for the analysis of many important tasks and activities, these approaches have some very real limitations. These tools are often limited in the granularity of the data that are collected. A web log that provides detailed information about the paths followed by various users in the course of completing some tasks does not contain any information about the users' activities while they were on a given site. Similarly, an email client may provide information regarding the structure of nested mailboxes, but information about intermediate states—such as the names of mailboxes that were created and later deleted—may not be captured.

Numerous studies have looked at email use from a variety of perspectives, including understanding how users "refind" old emails (Whittaker et al., 2011), using content to personalize search results (Teevan et al., 2005), and understanding how batching and work practices influence productivity and stress associated with email (Mark et al., 2016).

12.3 ACTIVITY-LOGGING SOFTWARE

Software tools for logging and recording user activity can provide rich data for usability studies. Tools that capture mouse actions (movements and clicks), keystrokes, and other interactions can help us identify common sequences, understand actions used to complete tasks, and often to gather information about transitions between different tools. Unlike the web server logs described earlier, these data collection tools can be applied to many different applications, providing the possibility of insight into the use of email, office productivity tools, and core operating system features.

These tools generally fall into two main categories: proxies and interaction recording tools. Proxies intercept user actions and record appropriate data points before passing the actions on to the original software (Figure 12.6). Data returned from the application can also be intercepted and modified before being returned to the user. Both user interaction data and application response data can be stored in a log file.

Interaction recording tools generally capture screen video and potentially microphone audio, providing a record of what happened and when. The resulting video and audio streams provide context and details not possible with simple proxies, allowing us to know not just that the user was working with a word processor, for example, but what she was typing and often why. Some usability tools use a combination of recording tools and proxies to capture both raw events and video, providing a rich mix that puts recorded actions in context.

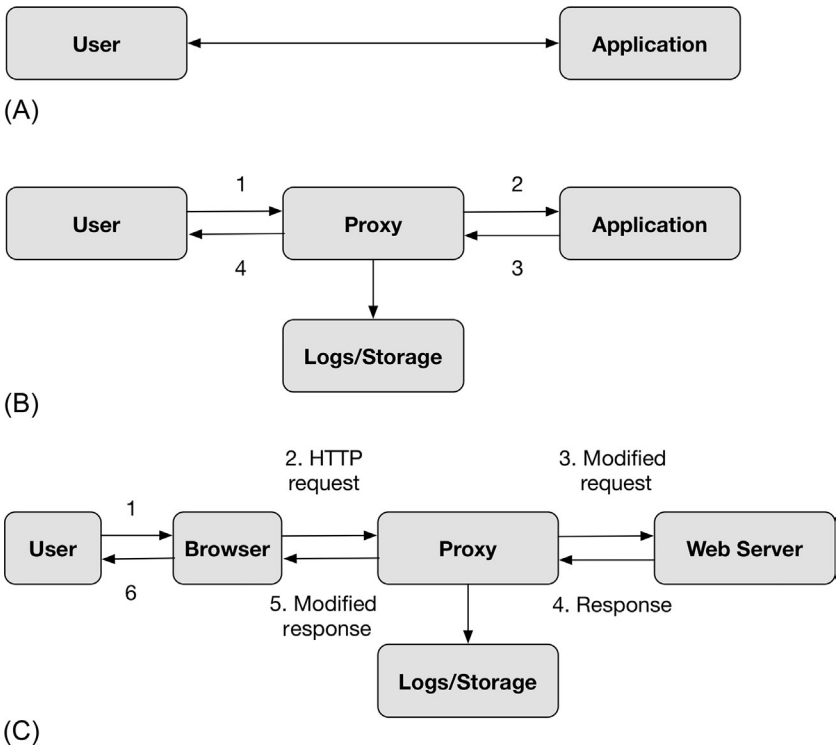


FIGURE 12.6

User actions: (A) passed directly to an application, (B) intercepted by a proxy, and (C) managed through a web proxy.

Proxies and interaction tools lie in between web logs and custom software in terms of flexibility and utility. Although they capture more data and can be more flexible than web logs and existing software (Section 12.2.2), they require more work in configuration. However, the tools described in this section are significantly less challenging than custom software.

12.3.1 WEB PROXIES AND INTERACTION LOGGERS

Web proxies were initially designed as tools for optimizing web browsing: users in an organization would forward all of their web requests to a specified proxy server, which would store copies of requested sites in a local cache. If an outgoing request asked for a page that was in the cache, the proxy server would simply return the contents of the cache entry, thus saving the cost of making a new request to the remote server and processing the response.

Web proxies for HCI research involve a more general approach to this model. In essence, the proxy becomes an intermediary that receives all web requests from

a group of users, retrieves the requested materials, and returns them to the users. As all requests from a group of users are handled by the proxy server, it can collect complete session data for all users. This provides a broader picture of user activities than standard server logs, which only contain records for requests from a single site.

Web proxies can intercept (and modify) user requests before sending them on to the server. Proxies can also modify the responses from the remote servers before the resulting web pages are displayed by the client software. Specifically, pages can be modified to include content necessary for the collection of additional interaction data (Atterer et al., 2006).

The first step in using a web proxy—for any purpose, including HCI research—is selecting an appropriate computing environment. As the computational demands of handling web requests for a large group of users can be substantial, you probably want to dedicate resources (computers, disk space, and network bandwidth) specifically for this purpose. If your proxy server is not able to process web requests quickly and efficiently, users will notice delays in their web browsing. This may cause some users to change their browsing habits, while others may simply refuse to use the proxy server. Ideally, the proxy server should not impose any performance penalties on end users.

Many open-source shareware, and commercial proxy servers are available for all major computing platforms. The Squid proxy server (http://www.squid-cache.org) is widely used on Linux and Unix systems. The popular Apache web server (http://httpd.apache.org) can also be configured to act as a proxy server. The choice of platform and software is likely to be dictated by your specific computing needs.

Once installed, proxy software must be appropriately configured and secured. You need to consider who may use your proxy server—you can limit access to users only from certain Internet domains or numbers—which sites you will allow access to, and what sorts of information you might want to store in the logs. As configuration options differ widely from one proxy package to the next, you should carefully study your software documentation and related resources.

Web browsers must be configured to use general-purpose web proxies. The configuration process tells the browser to contact the appropriate proxy host for all web requests. The most straightforward approach is to specify the proxy server settings directly in a web browser configuration dialog (Figure 12.7), but this requires manual configuration of every browser. Alternatives include proxies at the level of the Internet gateway or router—many organizations and companies use proxies or similar intermediate processors to filter web content, for purposes such as blocking adult content. This approach might be possible in some organizations, but would likely require working with your IT support teams.

Once the proxy server and web browser have been configured, users can continue to browse the web as before. Web requests are handled transparently by the proxy server and noted in the log files (Figure 12.8).

The resulting log files contain information on all sites visited by all users of the proxy. This is a major difference between proxy servers and web logs (Section 12.2.1). Whereas web logs maintain access requests for a single site, proxy servers track all

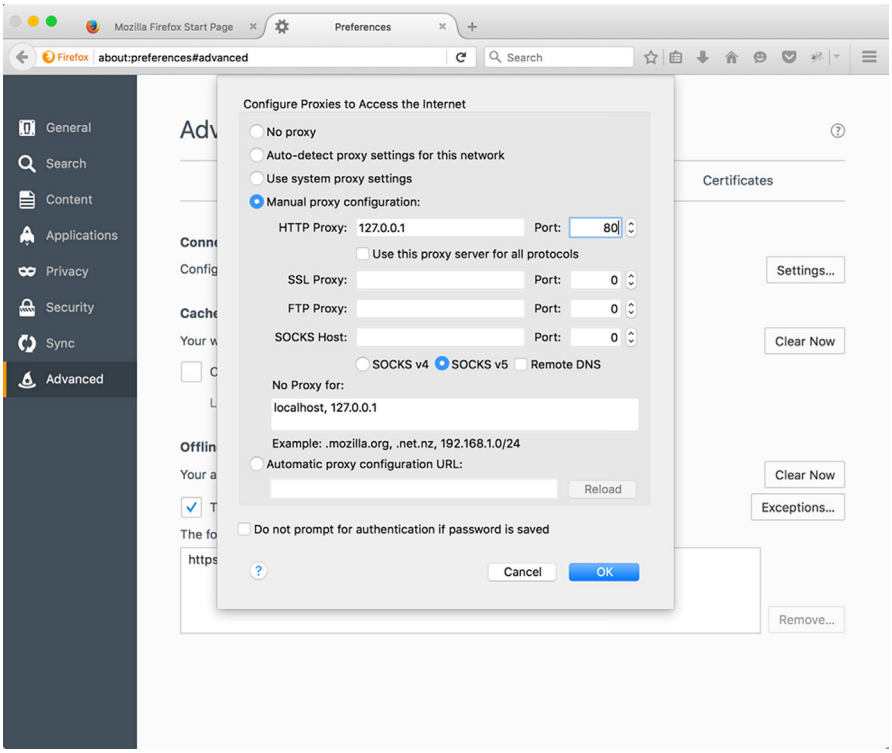


FIGURE 12.7

Firefox proxy configuration dialog for the web browser: the computer at address 127.0.0.1 will act as a proxy server on port 80.

```
- 127.0.0.1 - - [26/Sep/2007:10:22:19 -0400] "GET http://triton.towson.edu/~hhochhei/ HTTP/1.1" 304 -
- 127.0.0.1 - - [26/Sep/2007:10:22:19 -0400] "GET http://triton.towson.edu/~hhochhei/hhstyle.css HTTP/1.1" 304 -
- 127.0.0.1 - - [26/Sep/2007:10:22:19 -0400] "GET http://triton.towson.edu/~hhochhei/hh.jpg HTTP/1.1" 304 -
- 127.0.0.1 - - [26/Sep/2007:10:22:19 -0400] "GET http://triton.towson.edu/~hhochhei/towson-header.gif HTTP/1.1" 304 -
- 127.0.0.1 - - [26/Sep/2007:10:22:19 -0400] "GET http://triton.towson.edu/~hhochhei/arrow.gif HTTP/1.1" 304 –
```

FIGURE 12.8

Log entries from a proxy server. Note that each request contains a full URL for a web resource, as opposed to a local path, as in Figure 12.2.

requests from all users—capturing information on the use of many web sites not controlled by the team managing the proxy and collecting the data. Since proxy access is configured through the browsers or routers, users might not even be aware that the proxy use—or the data collection—is happening.

Although this information can be used to provide a rich picture of user browsing habits over time, care and discretion should be used when studying this data. As users may forget that they are using a proxy that logs their requests, they might visit

sites that are potentially embarrassing or inappropriate. Records of this sort should be treated carefully, including detailed and clear explanations in any consent forms. Possible approaches for avoiding embarrassment include anonymization of users or websites.

Logging of web requests is only the tip of the proxy iceberg. Several researchers have made extensive—and creative—use of web proxies to collect web usage data. WebQuilt (Hong et al., 2001) was a proxy server specifically designed to aid in the collection of usability data. WebQuilt combines logging facilities with an engine for transforming log file entries into inferred user actions, a tool for aggregating log files into graph structures, and visualization components for the display of graphs displaying user paths through a site.

Unlike general-purpose web proxies, WebQuilt was designed to be used to collect data on a site-specific basis. To run a usability test for a given website, the experimenter asked users to visit a URL specifically designed to support proxy-based access to the site under investigation. The WebQuilt proxy handled all requests for the site, including the modification of page content to route subsequent requests for that site through the proxy. As a result, WebQuilt did not require any configuration of the browser software.

Proxy servers can be quite powerful, but they present numerous technical challenges. Installing, configuring, and managing a proxy server can be difficult. Your proxy server must have the processing power and network bandwidth for effective operation. If your study involves only a small set of users for a short time frame, a single machine might be sufficient. Large-scale studies involving multiple users for extended time periods might need a more robust solution, involving many machines and more bandwidth. Cutting corners on proxy capabilities might jeopardize your study: if users find that the proxy is too slow for effective web use, they might temporarily or permanently stop using the proxy, effectively removing their data from your study.

Instrumentation software can often provide an attractive alternative to proxies. MouseTracks (Arroyo et al., 2006), UsaProxy (Atterer et al., 2006), and other similar systems (Kiciman and Livshits, 2010; Carta et al., 2011a,b; Huang et al., 2011, 2012) modified pages with JavaScript code that recorded low-level interaction data including mouse movements. This data was sent to the server for logging and visualization. Similar approaches have also been used to track touch interactions on mobile devices (Buschek et al., 2015). Although somewhat less flexible than JavaScript, which can be delivered solely from the server hosting relevant web pages, browser plugins can also be used to record detailed user interactions (Guo and Agichtein, 2009).

Selection of appropriate tools for tracking web interactions will likely require tradeoffs between expressive power and complexity. Proxies are relatively easy to configure. An organizational proxy can transparently collect and log access information for multiple web sites, without requiring any changes to those sites. Capture of more fine-grained data, through JavaScript, plugins, or any of the research idea tools

discussed earlier, will require additional programming and configuration, possibly including changes to the sites under question. Although potentially labor-intensive, these changes may provide access to otherwise unavailable data regarding user interactions with your sites.

12.3.2 KEYSTROKE AND ACTIVITY LOGGERS

Modern GUI windowing systems that support multitasking and concurrent use of multiple related tools present intriguing possibilities for HCI research. How often do users change applications? How many tasks do people work on at any given time? How long do users generally work within any given window before switching to another? What fraction of time is spent on overhead such as resizing windows, moving windows, or adjusting system controls? Answering these and other related questions requires data collected at the level of the operating environment.

Activity-logging software runs invisibly in the background, recording mouse movements, keyboard input, and windowing systems' interactions including window movement, resizing, opening, and closing. These tools act as proxies for user interaction events, recording events as they happen, before they are passed along to applications or the operating environment. Keyloggers are a special subclass of activity-logging software, focusing only on keyboard input. Activity-logging software has achieved a fair amount of notoriety in recent years, as these tools have been used as "spyware," to surreptitiously record user interactions in the hopes of stealing passwords, finding evidence of criminal behavior, or collecting evidence of spousal infidelity.

Commercial activity-logging products are often marketed as being tools for employers and parents to track inappropriate computer use by employees and children, respectively. Although some of these tools might be appropriate for data collection for research purposes, some antispyware programs may defeat or remove activity loggers. You may want to test the logging software on relevant computers and disable antispyware measures before trying to use these tools to collect data.

The disruption and recovery tracker (DART) (Iqbal and Horvitz, 2007) logged window positions and sizes, window actions, user activities, and alerts from various systems. DART's design presents an example of the responsible use of these tools for conducting legitimate research while remaining sensitive to privacy concerns. Keyboard logging was limited to a subset of possible choices, including menu shortcuts and some punctuation, and only a portion of each window title was collected. The resulting data therefore did not include file names, email addresses or subject lines, or web page titles. The analysis of more than 2200 hours of activity data collected from the main computers of 27 people over a two-week period generated numerous insights into time lost due to email or instant-messaging alerts, and how users respond to and recover from those interruptions (Iqbal and Horvitz, 2007). Logging studies can also be used to collect data on effective use of devices, as in a study that captured mouse movements of

adults with disabilities as they completed everyday computer tasks, in the hopes of building tools that might adapt to better suit the needs of these users (Hurst et al., 2013).

Mobile devices present additional possibilities for activity tracking, providing not only the opportunity to record which keys and controls were activated and when, but also detailed information regarding users' geographic location. HCI researchers have used logs of user location to predict short-term motion of individuals in crowds in a city environment (Fan et al., 2015), to infer movement characteristics associated with depression (Canzian and Musolesi, 2015), and undoubtedly for countless other interesting questions. See Chapter 14 for further discussion of the possible uses of mobile and ubiquitous computing for tracking user activity.

12.3.3 INTERACTION RECORDING TOOLS

Think-aloud studies and contextual inquiries involving direct observation and recording of user activities can be invaluable means of identifying and understanding usability problems, but they can also be a challenge to interpret. Recording what goes on as a user executes a series of actions to complete a complex task can be time consuming and error-prone, and subsequent discussions of the activity details and motivating context can be hard to capture. Similarly, although studies of log files can indicate what happened and when, the why of the observed interactions is often harder to gauge.

Screen capture and audio recording tools can provide invaluable assistance in these situations. Full-screen video with mouse pointers and accompanying audio can provide rich detail suitable for detailed analysis down to the mouse click. Audio can capture user comments vital for interpreting outcomes of think-aloud studies or other usability inquiries. Screen capture can also be very useful for exploring the use of computational tools as work as being conducted, as unobtrusive recording might capture interactions with greater realism than possible in lab settings. This approach has been used to study contexts including work in law offices (Cangiano and Hollan, 2009) and the use of electronic medical records during patient visits with physicians (see the "LAB-IN-A-BOX" sidebar, Chapter 13).

Recording tools generally come in one of two flavors. Commodity tools adopted for research provide a simple and cost-effective, yet limited solution. Screen recorders generally used to capture demonstrations will capture some or all of the screen, along with audio, providing a video in a standard format such as MP4. Some real-time web conferencing services include similar recording facilities, providing an excellent option for studies involving users in remote locations.

For more functionality at a likely higher cost, many researchers choose to use dedicated usability study software packages. These tools augment basic screen and audio capture with linked and integrated loggers for mouse and keyboard action, often with additional data streams including webcam images of the users at work.

Although potentially expensive and requiring some effort in configuration and management, tools in this class—led by TechSmith's Morae software—have been widely adopted by HCI researchers.

12.4 CUSTOM SOFTWARE

Modern computing applications are complex: word processors, spreadsheets, email programs, and web browsers may have dozens, if not hundreds of toolbar buttons and menu items. Which of these items are used and which are not? How would modifications to the interface change usage patterns for various functions? These questions are of interest both to researchers, who might be interested in understanding the efficacy of various strategies for grouping and rearranging controls for complex operations, and for product developers interested in comparing the effectiveness of proposed interface changes.

There are several approaches to collecting detailed data on the usage of complex interfaces. User observations, interviews, video recordings, and other strategies described elsewhere in this book can and have been used effectively for these purposes. However, these approaches are all laborious and expensive, requiring many hours spent observing users, asking questions, or coding events on videotape.

In many cases, a more attractive alternative is to have the software collect data on its own usage. A program designed for this sort of data collection would store every important user action—menu choices, toolbar button selections, key presses, and more, in a log file or database. Storage of these events in chronological order, including a timestamp, would provide a complete history of which options were selected and when. Analysis of this data might help developers understand which commands are used frequently, rarely, or usually in close combination with other commands (such as "cut" followed by "paste").

12.4.1 INSTRUMENTED SOFTWARE

The practice of adding measurement and recording tools to software is known as *instrumenting*. Constructing instrumented software may require a fair amount of technical expertise, as code for handling user interactions must be substantially modified. For many commercial or "closed source" software products, this level of access to the source code may be available only to the vendor or the developer of the product. However, macro and extension facilities in some products have been successfully used to write instrumentation code for interface evaluation purposes. A third possibility involves open-source software. Researchers interested in studying the usage of the interfaces of open-source projects might produce their own, instrumented versions of popular programs for use in research data collection. Detailed versions of each of these approaches are described in the Instrumented Software for HCI Data Collection sidebar.

INSTRUMENTED SOFTWARE FOR HCI DATA COLLECTION

Instrumented software has been used to collect usage data in support of widely used commercial products, research prototypes, and open-source tools. These examples are representative of some of the possibilities.

Microsoft Office 2003

Microsoft's Customer Experience Improvement Program let users opt in to having usage data collected anonymously. Data collected includes menu selections, keyboard shortcuts, and artifacts of user customization, including the number of mail folders and any modifications or customizations. This broad-ranging data collection was open ended, rather than hypothesis driven: "In short, we collect anything we think might be interesting and useful as long as it doesn't compromise a user's privacy" (Harris, 2005).

The large data set (over 13 billion user sessions) provided substantial insight that informed the redesign of the Office interface for the Office 2007 release (Harris, 2005). Even though the Paste command—the most popular, with more than 11% of all command usage in Word—was frequently accessed via shortcuts, the Paste button was the most frequently clicked button on the toolbar. This led Microsoft's UI team to place the Paste button prominently in the revised interface for Word (Harris, 2006).

This study also confirmed that Word users frequently use a small subset of features while rarely using other features (McGrenere and Moore, 2000). The top five commands in Word accounted for more than 32% of all command usage, with frequencies declining quickly after the top 10 (Harris, 2006).

Personalized Versions of Application Interfaces

Noting the potential difficulties associated with complex interfaces for desktop applications, McGrenere et al. set out to investigate the possible utility of a simplified user interface containing only items selected by the user. Using the scripting tools in Microsoft Word 2000, they built an extension to Word that would allow users to work with this simplified interface. Tools for adding items to their personalized interface were included, along with a control that could be used to switch between the simplified interface and the full interface as desired. In a field study with 20 users, this software was installed along with a logging tool for capturing usage and a program that would upload usage logs to an Internet server. Usage data collected included histograms of function usage frequency. This data indicated that only a small number of commands were used very frequently and that the users added almost all of those commands to their personalized interfaces. A series of questionnaires indicated that users preferred the personalized interfaces in terms of navigation and ease of learning (McGrenere et al., 2002).

(Continued)

INSTRUMENTED SOFTWARE FOR HCI DATA COLLECTION—CONT'D

Instrumenting Open-Source Software

As few commercial products offer customization tools comparable to those found in Microsoft Office, instrumentation of open-source software has proven to be a fruitful alternative. One study of web navigation patterns used an instrumented version of the Firefox web browser to collect data on the use of browser features such as the "back" button, history views, and bookmarks. This relatively small study (25 users) combined instrumented software with web proxies in order to identify new patterns in web browser feature usage and browsing behavior, some of which may have been related to the rise in tabbed browsing and other relatively new browser features (Obendorf et al., 2007).

Ingimp provides an example of broader use of instrumented open-source software for HCI data collection. Short for "instrumented GIMP," ingimp was an instrumented version of the Gnu Image Manipulation Program, a powerful open-source tool for photo editing and image processing. Created by a group from the University of Waterloo, ingimp was widely publicized in the hope of motivating users to participate in the study.

Ingimp collected a variety of data, including usage timing, the number of windows and layers open at a time, command usage, and task-switching details. Instrumenting GIMP to collect this data required modifying the open-source program to record appropriate events and transmit them to a central server. Interaction data is transmitted at the end of each session. If the software crashes before a log is transmitted, the incomplete log is detected and sent to the server when the program is next used.

The ingimp instrumentation approach involved several privacy protection measures. Although mouse events and key press events are recorded, specific details—which key was pressed or where the mouse was moved—are not recorded. A dialog box on startup provides users with the option of disabling event logging for the current session. As GIMP is an open-source project, the developers of ingimp made all the source code available. Knowledgeable users can investigate "patches"—descriptions of the differences between the original GIMP and ingimp. These differences reveal where the logging code has been added and what details it logs. Although few (if any) users are likely to take the trouble to do this, this does represent a thorough attempt at full disclosure.

Ingimp's developers used this information to improve the usability of GIMP and other free or open-source software tools (Terry et al., 2008).

Whichever approach you select for implementing data collection instruments, you should think carefully about the data that you are collecting. Although you may be tempted to collect as much data as possible, doing so may not be beneficial. Instrumenting every possible interaction in a complex application may require a great deal of effort, and as the amount of data collected may increase with the

extent and granularity of the instrumentation, you might consider exactly what you need to capture. In some cases, individual mouse movements and key strokes might be needed, while other studies might need only higher-level user actions, such as selection of the "paste" operation. Careful attention to the relationship between your experimental hypotheses—what do you hope to learn?—and the data collected may help increase your chances of success. Another successful strategy involves associating each recorded action with one or more categories, allowing ease of processing and filtering by criteria appropriate to an analysis. This approach might allow comparison of keyboard and mouse-movement records to higher-level task indicators.

12.4.2 RESEARCH SOFTWARE

Another class of custom software tools for automatic data collection involves software that is explicitly created for the sole purpose of running an experiment. These tools generally present users with a series of tasks to be completed and record data regarding task completion time, errors, and whatever other data may be necessary. The Fitts' Law, Children, and Mouse Control sidebar discusses an example of a custom software package developed for a study of how well young children use computer mice. Researchers interested in studying how well young children use a mouse built a tool that tracked task completion time as well as the trajectory of mouse movements in tasks that involved moving between two targets. This study found that younger children were much less accurate mouse users than adults (Hourcade et al., 2004).

FITTS' LAW, CHILDREN, AND MOUSE CONTROL

Full-size computer keyboards, keypads on phone and small devices, mice, trackballs, jog wheels, and joysticks are familiar controls for computers and other electronic devices, but familiarity does not necessarily imply understanding. How do we use these tools? How efficient are we? What sort of mistakes do we make? What are the factors that determine task completion time, accuracy, and error rate? Although researchers—in cognitive psychology and more recently in HCI—have been asking these and similar questions for more than 50 years, detailed study of the human use of these devices can still lead us to valuable insights.

Target selection is an important task in this area. Given multiple targets that a user might want to select—keys on a keyboard or buttons on a graphical user interface—what determines how quickly and accurately a user can move from one to another? Studies of target selection performance guide the size and selection of graphical icons, placement of buttons on a cell-phone keypad, and many other aspects of interface design.

(Continued)

FITTS' LAW, CHILDREN, AND MOUSE CONTROL—CONT'D

Paul M. Fitts conducted pioneering experiments in this area in the 1950s, leading to the development of Fitts' law, a frequently cited result in HCI research.[2] Originally intended as investigations of the theoretical limits of human performance in performing tasks of differing amplitudes of movement, Fitts' experiments involved asking participants to move between two targets separated by a distance. Fitts found that the information content of the task was determined by the distance between the targets and the inverse of the width of the targets (Fitts, 1954). This result was later generalized to expressive movement time as being a function of the logarithm of the ratio of the movement amplitude to the target width (MacKenzie, 1992).

Fitts' law tells us that as the distance between targets increases, or the size of the targets decreases, the time required to move between them increases. This has a certain intuitive appeal: it is harder to reach small targets than it is to reach larger targets, just as we can cover short distances more quickly than we can cover long distances. As much of our interaction with computers involves target selection, Fitts' law can help us understand the impact of design decisions regarding the placement and sizing of icons on a screen or keys on a keyboard.

Fitts' law is important enough to have spawned follow-on works, with researchers examining a wide variety of variations on the original task (MacKenzie, 1992; MacKenzie and Buxton, 1992). Extensions and novel applications of have confirmed the relevance of Fitts' law to the use of mobile devices while walking (Lin et al., 2007); developed models for "two-thumb" text entry on small keyboards (Clarkson et al., 2007); proposed extensions for nonrectangular targets (Grossman et al., 2007); explored implications for novel input modalities including multitouch devices (Nguyen et al., 2014) and flexible displays (Burstyn et al., 2016); added an extra dimension for virtual reality and 3D displays (Lubos et al., 2014; Teather and Stuerzlinger, 2014; Janzen et al., 2016); and even extended Fitts' law to foot input for under-desk devices (Velloso et al., 2015).

Juan-Pablo Hourcade and his colleagues at the University of Maryland faced this problem in the course of their work with young children. Faced with 5-year-old children who had difficulty clicking on computer icons, they set out to understand how preschool children differed from young adults in their ability to complete target-selection tasks (Hourcade et al., 2004). Although several researchers had conducted Fitts' law research with young children, none had specifically addressed the question of whether performance differences justified the effort required to build interfaces specifically for this class of young users.

[2] Authors of books on HCI research are contractually obligated to refer to Fitts' law at least once.

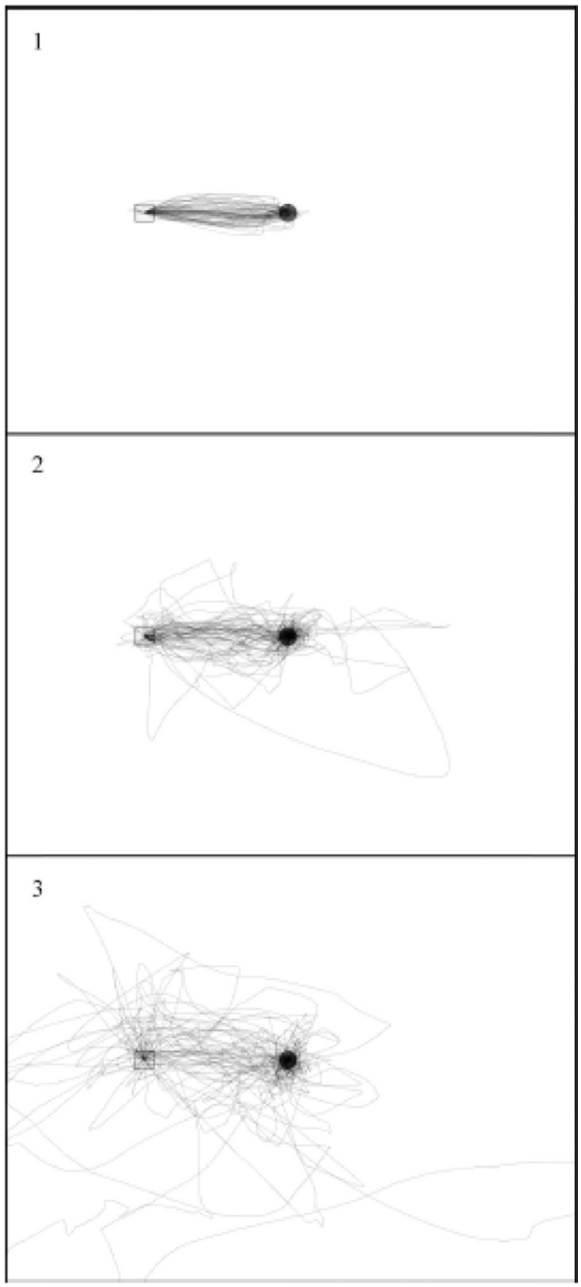


FIGURE 12.9

Aggregate mouse controls for all users show striking differences in the paths covered by different groups: (1) adults, (2) 5-year olds, and (3) 4-year olds (Hourcade et al., 2004).

From Hourcade, J.P., Bederson, B.B., Druin, A., Guimbretière, F., 2004. Differences in pointing task performance between preschool children and adults using mice. ACM Transactions on Computer-Human Interaction 11 (4), 357–386.

(Continued)

FITTS' LAW, CHILDREN, AND MOUSE CONTROL—CONT'D

Their study involved 13 4-year-old children, 13 5-year-old children, and 13 adults (between 19 and 22 years old). Participants were asked to move the mouse from a home area to a target to the right. Targets had three diameters—16, 32, or 64 pixels—with three distances between start and target—128, 256, or 512 pixels. Participants completed 45 tasks in roughly 15 minutes—about the limit of the attention span of 4- to 5-year-old children. Data collected measured accuracy (did they press the button inside the target), time, and measures of reentry (leaving and reentering the target). Software developed for conducting the experiments also collected mouse motion data sufficient for reconstructing mouse movement paths.

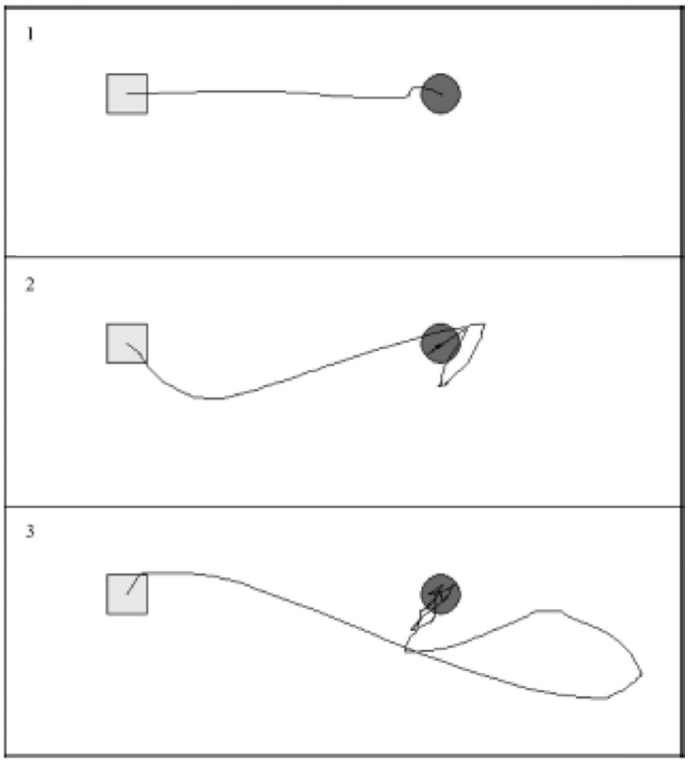


FIGURE 12.10

Typical paths illustrate greater reentry rates for children: (1) adult, (2) 5-year-old, and (3) 4-year-old (Hourcade et al., 2004).

From Hourcade, J.P., Bederson, B.B., Druin, A., Guimbretière, F., 2004. Differences in pointing task performance between preschool children and adults using mice. ACM Transactions on Computer-Human Interaction 11 (4), 357–386.

Comparison of the data from children and adults generated several important insights. Children were slower and less accurate than adults. Children also tended to hover over targets, reentering much more frequently than adults. Hourcade and colleagues found that while Fitts' law does apply to children, the model is more accurate for adults.

Mouse motion paths (or "trails") provide striking illustrations of the differences between children and adults. While adults move accurately and quickly between targets, 5-year-old children went far afield, often overshooting targets. Four-year-old children were even worse, moving all over the screen (Figure 12.9). Children were also much more likely to repeatedly enter, leave, and reenter targets (Figure 12.10).

Based on these differences in performance profiles, Hourcade et al. suggested several possible approaches to designing interfaces for young children. Possible solutions include larger icons, smaller mice, expanding targets, slower or accelerated mouse movement, and constrained motion between selections (using directional arrows and a selection button) (Hourcade et al., 2004).

12.5 HYBRID DATA COLLECTION METHODS

If one source of HCI data is good, then two must be better. Multiple channels of data collection can be combined to overcome the shortcomings of any one approach. Logging user interactions with a word processor may be a good start toward understanding how various controls are used, but log files provide little, if any, contextual information that might prove invaluable for interpretation and analysis. Video recordings or direct observation of users at work can help researchers understand users' goals, frustrations, state of mind, and satisfaction (or lack thereof) with the system being used. Taken together, these data sources provide a much more detailed and complete picture of user activity than either would on its own.

Combining outputs from log files and video sessions may require specialized software support. Ideally, we might want to identify an event in a log file and jump right to the video recording that displays what the user was doing at that moment. HCI researchers have developed tools that provide this synchronized access (Hammontree et al., 1992; Crabtree et al., 2006; Fouse et al., 2011). These features may also be available in professional tools for usability studies, such as the interaction recording tools discussed in Section 12.3.3.

Similar approaches can be used for evaluation of web-based systems. Web log files and web proxies are limited in their ability to capture details of user interactions—and possibly in their ability to identify distinct users. However, they are relatively easy to configure and deploy. An instrumented web browser can provide vastly greater detail regarding specific user events, but it requires installation of unfamiliar software on end-user computers, potentially limiting the number of participants. One

way to resolve this dilemma is to use both approaches in the same study. Specifically, data would be collected through both proxies and instrumented browsers (Obendorf et al., 2007). Although this approach might be more expensive and time consuming than either approach used independently, the resulting data may be of higher quality.

Another form of hybrid might combine automated data capture and analysis with observation or other qualitative approaches (Chapter 11). As mentioned, log files from web activities or instrumented software are limited in their ability to describe the context of work. It is often difficult to go from the fine-grained detail of individual actions in a log file to a broader understanding of a user's goals and motivations. If we combine log data with active observation by a human researcher, we stand a better chance of understanding not just what the user was doing, but why she was doing it. The observer might sit behind the subject, watching her activities and making notes in real time, creating a log of observations that can be synchronized with the events in the server log. Alternatively, video recordings allow for annotation and observation at some later time. Log analysis studies involving remote users or those not involved in a formal study (see the discussion of "A/B" testing in Chapter 14) might be accompanied by an optional survey at the end of a session, asking users to complete questions relating to their satisfaction with the system (see Chapter 5 for more discussion of surveys). In any case, appropriate software can be used to view individual user events alongside observer annotations and content, thus providing a more detailed and informative picture than either source would give on its own. Combinations of multiple log approaches with observer annotations can provide even greater detail.

12.6 DATA MANAGEMENT AND ANALYSIS

12.6.1 HANDLING STORED DATA

Whenever you write or modify software to track user activities, you need to decide how to manage the data. Two approaches are commonly used: log files and databases. Log files are plain text files that indicate what happened, when it happened, and other details—such as the user ID—that might help when interpreting data. Log files are easy to write, but may require additional tools for interpretation. The comments from Section 12.2 are generally applicable to any application logs, with one important exception. As commonly available software tools that parse and interpret standard web log formats may not be immediately applicable to logs that you might develop for your software, you may need to dig in and develop custom tools for parsing these log files.

Databases can be very useful for storing user activity information. Carefully designed relational databases can be used to store each action of interest in one or more database tables, along with all other relevant information. Powerful query languages, such as SQL, can then be used to develop flexible queries and reports for interpretation of the data. This approach may be most useful when working with an application that already connects to a relational database. When your tool uses a relational

database to store application data, additional user activity information can often be added without much effort. This is often the case for database-driven web applications. If, however, your tool does not interact with a database, developing tools to parse log files might be easier than adding a database to the application.

12.6.2 ANALYZING LOG FILES

Having collected some log files, you will want to do something with them. Although log files for web servers, proxies, keystroke trackers, and custom-instrumented software might all have different formats and contents, the general approach toward instrumentation is roughly the same: in each case, you have one line in the file for each event of interest. Each line is likely to have some text indicating the time and date of the event (otherwise known as the timestamp), a description of what happened (such as the URL that was requested), and other related details.

How you proceed in your analysis is largely determined by your goals. If you are simply interested in trying to count certain events—for example, how many people pressed the Print button—you might be able to read through the file, classifying each event into one or more counters of various types. A single event in a log file might be classified according to the page that was requested, the day of the week, the time of day, and the type of web browser that made the request.

Reading through the file to extract the various pieces of information known about each event is an example of a common computing practice known as *parsing*. Often written in scripting languages, such as Perl and Python, log-file-parsing programs read one line at a time, breaking the entry for each event into constituent pieces and then updating data structures that keep counts and statistics of different types of event, as needed. Once the parser has read all of the relevant events and tallied up the numbers, results can be displayed graphically or in tabular form.

Countless programs for parsing and analyzing web log data have been developed since the web first came onto the scene in the 1990s. These tools range from freely available, open-source (but still highly functional) offerings to high-end commercial products, providing a variety of ways to slice-and-dice data. Many of these tools work on data from proxy servers as well.

For publicly available websites, many operators rely on the detailed querying and visualization tools provided by Google Analytics. Using a small bit of code inserted to every page on the site, Google Analytics collects data and sends it to Google, where it is stored for analysis via Google's tools. Google Analytics is a popular and powerful tool for understanding website usage patterns, but as it is not intended for supporting usability studies, you might want to try a test run before using it for a full study. Furthermore, as Analytics only works on public sites, it is not appropriate for studies using locally hosted material.

Data from nonweb applications might prove a bit more challenging to analyze. Keystroke loggers and activity loggers may come with their own log-parsing and analysis packages, but you are likely to be on your own if you write software instrumentation to collect data. One approach in this case might be to design your log files

to match the formats used by web servers. This mimicry would make your data amenable to analysis by web-log analysis tools. Another possibility is to create parsing and analysis software: if you can instrument your user interface to collect interaction information, you will probably find this to be a reasonably manageable task.

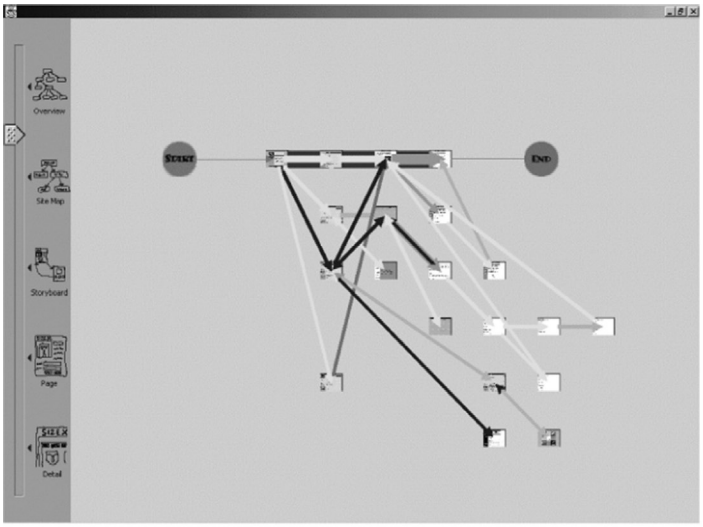
More sophisticated questions might require fancier footwork. One common goal is to study the sequence of events. Do users click Print before Save more frequently than they click Save before Print? Similar challenges are found when trying to infer the structure of interaction from web logs, leading to a variety of strategies that have been used to pick out user "sessions" (Heer and Chi, 2002).

Another approach might be to visualize log files. Highly interactive visualizations might show each event in a log file as a point on the screen, while providing tools for filtering and displaying data based on different criteria. As with other approaches for analyzing log files, visualization has been most widely used for web logs. WebQuilt (Hong et al., 2001) displays pages and links between them as nodes and links in a graph. Links are drawn as arrows, with thicker arrows indicating more heavily used links and shading indicating the amount of time spent on a page before selection of a link (Figure 12.11A). Users can zoom into a node to directly examine the page in question (Figure 12.11B).

Other visualizations include the use of two-dimensional "starfield" displays for viewing individual requests by date, time, and other attributes (Hochheiser and Shneiderman, 2001) and finer-grained visualizations of mouse events on individual pages (Arroyo et al., 2006; Atterer et al., 2006).

As with any other analysis, understanding your goals and planning your data acquisition and analysis appropriately is key to effective use of these detailed logs. Ben-Naim et al. describe the use of log analysis for an adaptive learning program, including an explicit list of the questions involved and a description of the approaches used to answer those questions (Ben-Naim et al., 2008). Appropriate storage of log data can also facilitate analysis, with some researchers using business-oriented online analytical processing (OLAP) tools to drill down into relevant details (Mavrikis et al., 2015).

Data mining and machine learning techniques can be well suited for the extracting patterns from log files. Relatively simple techniques such as association rules (Agrawal et al., 1993) might be used to determine patterns of frequently cooccurring accesses—for example, "sitemap" and "search" page accesses might frequently be associated with clicks on a "contact us" page. Data mining approaches have been used to inform site design and usage characterization from a variety of perspectives, including personalization of content (Srivastava et al., 2000; Eirinaki and Vazirgiannis, 2003) with results familiar to any web users who have seen web pages including advertisements matching search terms that they have recently used. Clustering techniques might also be used to develop models useful for clustering users into groups based on their usage patterns or predicting desirable outcomes such as purchases. Although fascinating, and also relevant to the processing of physiological data (Chapter 13) and ubiquitous computing data (Chapter 14), data mining is largely beyond the scope of this book—for more information, see one of the many online courses or textbooks on machine learning and data mining.



(A)

(B)

FIGURE 12.11

WebQuilt visualizations of log file data: (A) visualization of paths between two endpoints and (B) a page in context (Hong et al., 2001).

From Hong, J.I., Heer, J., Waterson, S., Landay, J.A., 2001. WebQuilt: a proxy-based approach to remote web usability testing. ACM Transactions on Information Systems 19 (3), 263–285. Copyright ACM.

12.7 AUTOMATED INTERFACE EVALUATION

If using computers to collect data for HCI research is good, why not go further? Perhaps we can build software that automatically tests and evaluates user interfaces, generating data on usability issues or potential task performance times.

Automated inspection methods involve the analysis of aspects of user interfaces including layout, content, and language, in order to determine how well they conform to design guidelines (Ivory and Hearst, 2001). Generally, these tools provide reports indicating the extent to which an interface complies (or fails to comply) with guidelines. These reports can help designers understand where users might run into problems, and how an interface might be improved. The evaluations provided by these tools are generally based on empirical evidence, accepted design practices, and other accumulated experience. Automated inspection tools have been widely used in assessing the accessibility of websites. These tools examine web pages in search of images without explanatory text (<alt> tags), embedded scripts that might not be interpretable by screen readers, lack of navigation support, and other problems that may cause difficulties for users with disabilities. Dozens of web accessibility evaluation tools—ranging from free websites to expensive commercial software—have been developed. See Chapter 10 for more information.

Although these tools may provide some useful advice, the utility of any particular tool may be limited by the validity and scope of the underlying guidelines: analyses that are based on broad, well-supported guidelines are likely to be more appropriate than those that examine a narrower range of concerns. The use of multiple inspection methods to test interfaces from varying perspectives might be helpful. Ideally, these tools should be seen as companions to—not replacements for—traditional design reviews and user testing.

A variety of approaches to automated testing have been explored. Systems that focus on the use of modeling or simulation to predict task performance times and other quantitative and qualitative characteristics of interface usage are appealing, but they may be difficult to construct and limited in utility (Ivory and Hearst, 2001). Other efforts the use of modeling techniques that carry through the design and development phase into support for automated testing (Humayoun et al., 2012; Wollner et al., 2015).

12.8 CHALLENGES OF COMPUTERIZED DATA COLLECTION

Automated software for HCI data collection has many advantages. The use of software to record user interaction events along with timestamps can ease data collection for structured experiments, simplifying work that previously would have been done with a stopwatch and paper records. Logs from web servers and other activity tracking tools document interaction events in unstructured, "natural" activities with far less difficulty than earlier techniques of observing or videotaping users.

However, effective use of these tools requires addressing several important challenges. As with any method for data collection, automated methods work best if their use is carefully considered in the context of the specific situation that is being studied and the research questions that are being asked. Before collecting data—or designing a system to collect data—you should ask yourself what you hope to learn from your research and how the data that you collect will help you answer those questions. Collecting too much data, too little data, or the wrong data will not be particularly helpful.

Computer use can be considered over a wide range of time scales, with vastly different interpretations. At one end of the spectrum, individual keyboard and mouse interactions can take place as frequently as 10 times per second. At the other extreme, repeated uses of information resources and tools in the context of ongoing projects may occur over the course of years (Hilbert and Redmiles, 2000). Successful experiments must be designed to collect data that is appropriate for the questions being asked. If you want to understand usage patterns that occur over months and years, you probably do not want to collect every mouse event and key click; the volume of data will be simply overwhelming. Similarly, understanding dynamics of menu choices with specific applications requires more detailed information than simply which applications were used and when.

The amount of data collected is generally referred to as the granularity or resolution of the data. Fine-grained, high-resolution data involves every possible user interaction event; coarse-grained, low-resolution data contains fewer events, perhaps involving specific menu items, selection of specific buttons, or interaction with specific dialog boxes.

The specificity of the questions that you are asking may help determine the granularity of data that you need to collect. Many experiments involve structured questions regarding closed tasks on specific interfaces: which version of a web-based menu layout is better? To support these studies, automated data collection tools must collect data indicating which links are clicked, and when (see the Simultaneous vs Sequential Menus sidebar for an example). Web server logs are very well suited for such studies.

Open-ended studies aimed at understanding patterns of user interactions may pose greater challenges. To study how someone works with a word processor, we may need to determine which functions are used and when. Activity loggers that track individual mouse movements and key presses may help us understand some user actions, but they do not record structured, higher-order details that may be necessary to help understand how these individual actions come together to involve the completion of meaningful tasks. Put another way, if we want to understand sequences of important operations in word-processing tasks, we do not necessarily want a list of keystrokes and mouse clicks. Instead, we would like to know that the user formatted text, inserted a table, and then viewed a print preview. Still higher-level concepts are even harder to track: how do we know when a user has completed a task?

Researchers have tried a variety of approaches for inferring higher-level tasks from low-level interaction events. Generally, these approaches involve combining

domain knowledge of both the software being studied and user behavior to identify patterns that would be representative of defined tasks (Hammontree et al., 1992; Ivory and Hearst, 2001). These inferential efforts face many challenges. For example, applications that provide multiple methods for accessing given functionality (such as both a menu choice and a toolbar button for Print) may generate log files that contain all of these methods. However, log entry analysis approaches may not recognize these multiple paths as leading to a common goal. Establishing appropriate contextual information may also be difficult: log file entries that indicate a button was pressed are less informative than those that indicate which button was pressed (Hilbert and Redmiles, 2000).

Analysis challenges are particularly pronounced in the analysis of web server logs, which may contain interleaved requests from dozens of different users. Statistical analyses and visualization tools have been used to try to identify individual user sessions from log files (Pirolli and Pitkow, 1999; Hochheiser and Shneiderman, 2001; Heer and Chi, 2002), but these tools are imperfect at best. If a web browser coming from a given Internet address accesses a page on your site and then accesses a second page 10 minutes later, does that count as one session or two? Your log file cannot tell you if the user was reading the page between those two requests or if she was talking on the telephone. Those requests may not have come from the same person—for all you know, it is a shared computer in a library or classroom that is used by dozens of individuals on any given day.

Custom-built or instrumented software may alleviate some data-granularity problems by providing you with complete control over the data that is collected, at the expense of the time and effort required to develop the data collection tools. If you are willing and able to commit the resources necessary for software customization, you can configure the software to capture all of the data that you think might be interesting: nothing more, nothing less.

Unfortunately, matters are rarely so clean-cut. There may be a vast difference between what you think you need before you start large-scale data collection and what you may wish you had collected once you begin analyzing the data. The expense of running experiments—particularly those that involve substantial effort in participant recruitment—creates a tendency toward collecting as much data as possible. “It's easy to collect this information,” the thinking goes, “so we may as well. After all, storage is inexpensive, and these details may prove useful later on.”

Although there is a certain logic to the defensive approach of collecting as much data as possible, there are some limits to this approach. As anyone who has sifted through megabytes of event logs can tell you, collecting lots of data may simply leave you with lots of uninformative data junk to sift through. Even with software tools, the identification of meaningful patterns (as opposed to random coincidences) can be difficult. Lower resolution data may be somewhat easier to analyze.

If your data collection tools can clearly distinguish between coarse-grained and fine-grained events, you might be able to have your cake and eat it too. Data collection tools might mark each event with an indication of the level of granularity that

lets you fine-tune your analysis—looking only at high-level events, such as menu selections; only at low-level events, such as mouse movements; or perhaps some hybrid approach that examines low-level events that precede or follow interesting high-level events.

As with any HCI research, proper attention to pilot testing can be important. Pilot testing of both the data collection and data analysis pieces of the experiment can help you verify that the data you are collecting actually tells you what you want it to. Analyzing the pilot data may help you verify that you are collecting data of the appropriate granularity.

All of the approaches to automatic data collection raise potential security concerns. Logs of web browser activity can say a good deal about a person browsing the web. This information might be used to infer sensitive or embarrassing details about a person's habits, interests, or medical concerns. Although the potential harm from the logs of any single website may be relatively minimal, proxy servers can be configured to capture all of the interactions with every website visited by a given computer. Indirect (and sometimes nonexistent) links between people and computers make matters even worse in this regard. Web logs track the identity (in terms of the IP number) of the computer that makes each request. A number in a web server log may correspond to the computer on your desk, but this does not mean that you were the person that was at the computer when the browser visited embarrassing websites.

Activity loggers and keystroke loggers make matters even worse. By tracking every input action, these tools collect enough data to reconstruct documents, emails, calendars, and other damaging evidence. These tools have been surreptitiously used in criminal investigations and divorce proceedings. Regardless of your views on the appropriateness of using secretive software to spy on family members, you should take care to ensure that your data collection tools do not gather data that others would find sensitive, damaging, or otherwise private. Some approaches include customizing your tools to avoid potentially problematic data, such as specific keys that are pressed (as opposed to simply noting that a key was pressed) and window titles (which may contain document titles).

12.9 SUMMARY

Automated data collection systems give researchers the ability to easily collect detailed user interaction information. Appropriately configured software tools can be used to replace labor-intensive approaches such as manual observation or coding of events on video. The result is a qualitative, as well as quantitative difference: not only can more data be collected, but the increased ease of data collection allows researchers to conduct experiments that otherwise would be too difficult or expensive. These strengths make automated data collection a clear first choice for many HCI research efforts.

There are three broad categories of question that might benefit from automated methods of data collection:

- Retrospective analyses of information management behavior: These studies look at artifacts of computer use, including location of documents, email folders, and other structures created during the course of using and managing information, in order to understand how people use these tools.
- Controlled experiments: Web server logs and completely customized software can be used to collect timing and related data for experiments. As web logs contain entries for each link selection event, they are most useful for cases involving the study of selection of web links. With proper design, web links can be used to model menu layouts and related topics. Fully customized software may be needed if additional data (such as mouse movements) is required, but hybrids may be useful. For example, JavaScript embedded in a web page might be used to record mouse movements and translate those movements into events stored in a log file alongside the basic server logs.
- Usability studies and other explorations of how users work with tools: Web server logs, proxy server logs, keystroke loggers, and activity loggers record user interaction events with one or more websites, applications, or operating environments. The interactions can be used to examine which features of a tool a user used and when. With appropriate analysis, this data can be used to find interaction problems and identify opportunities for usability improvements.

Successful use of any of these approaches requires careful consideration of the appropriate granularity of data to be collected and the tools to be used for data analysis. As with other data collection approaches, the key is to precisely identify the data that is needed and collect only that data.

Tools that collect data on user activities have potential privacy implications. This is particularly important when the goal is to study how users work with tools to complete real tasks: providing artificial tasks in the hopes of reducing privacy concerns may decrease the realism of the data. Experiments involving this set of data should be carefully designed, in consultation with appropriate institutional review boards (see Chapter 15), to avoid violations of participant privacy and trust.

DISCUSSION QUESTIONS

1. Online spreadsheets, word processors, and other office productivity tools blur the line between websites and traditional software. In doing so, they provide both opportunities and challenges for HCI researchers. As the software infrastructure for online tools resides completely on the hosting server, researchers can easily modify and redeploy interfaces without having to update individual computers. As with traditional web interfaces, requests for content from the server can be logged and resulting files can be analyzed. However, as client-side interactions (usually executed through JavaScript code)

do not generate server requests, additional data recording measures may be necessary. What other challenges or opportunities can you see in such dynamic applications? Pick an online word processor, spreadsheet, or presentation tool: how would you design a system to study its usability?

2. A hybrid system for automated computer data collection might involve a combination of web server logs—ideally from a proxy that would track all of a participant's interactions—and one or more software packages instrumented to collect data of interest. What would be the pros and cons of such a system relative to a full-scale activity logger that would track all user interactions?

3. Legitimate concerns about user privacy have led some researchers to be very cautious about the data that they collect with keyboard or activity loggers. This appropriate concern for user privacy does not come without a cost: in throwing away details such as document titles, destination addresses for emails, and specifics of visited web pages, researchers lose information that might have been used to develop a more nuanced understanding of the underlying activity. For example, was the user sending email to colleagues at work, or at home? Can you think of ways to configure logging software to collect certain attributes of document titles, email headers, and related information in a manner that might prove useful for research purposes while still being respectful of user privacy? What effect might greater notification—perhaps telling participants that you might record sensitive information—have on your experiments?

4. The Fitts' Law, Children, and Mouse Control sidebar provides an example of experiments that used mouse motion data to study how children differed from adults in their use of mice. Although 3- to 5-year-old children do not make much of use keyboards, slightly older children might begin to type. How would you study differences between children and adults in terms of their use of keyboards? What sort of data would you collect and how would you interpret it? How would this differ if you were considering smaller keyboards such as those used on some cell phones?

RESEARCH DESIGN EXERCISES

1. Experiment with web server logs and log analysis on your desktop.

(a) Start by getting or generating a web server log file. You might ask computing support people in your school or company for some web log data. Log files can be *very* large: you probably only want a small snapshot. If your school or department gets a good deal of web traffic, you should be able to get a few megabytes of log data. Be forewarned, some network administrators may not like the idea of handing out this information. You may have to convince them that you will use it responsibly. Alternatively, you can install a web server and run it. If your computer does not have a

web server installed, the Apache web server (http://httpd.apache.org) is available for most major platforms. Download the server, install it, and configure it. The server configuration file (httpd.conf) will have entries that indicate where log files can be found. Once you get the server running, build a few web pages with links between them and access them.

(b) Examine the log files to determine what they can tell you about pages that were accessed, when they were accessed, and other related details.

(c) Find an open source web log analysis tool and use it to analyze the log files.

(d) For a further challenge, try to configure and use a web proxy server, such as Squid (www.squid-cache.org).

2. Use implicitly collected information data from your computer to conduct an investigation of information management patterns. Start with folders and subfolders for documents: Do you have all of your documents in one folder or do you have many subfolders? How many documents in each folder? How many subfolders in each subfolder? What is the maximum "depth" of your subfolders? How many documents do you have on your desktop? Collect similar information for your email: How many items are in the inbox? How many folders? Repeat this analysis with a friend's data. Can you draw any conclusions about data management habits and practices?

3. Try to find and use a keyboard logger or general activity tracker. Install the program on your computer and use it to accomplish some tasks. Find and examine the log files: what do they tell you about how you used the program? Can you relate the contents of the log files to the tasks that you performed with the program?

4. Some simple excursions into collecting data on keyboard and mouse usage can be conducted without writing custom software.

(a) For keyboard usage, carefully remove the backspace and arrow keys from your keyboard. Disconnect your mouse as well. Ask someone to type a paragraph of text into a word processor and time their response. As your participant will be unable to delete any mistakes or use the arrow keys to move to a different part of the text, you will get a record of exactly which keys were pressed. You can use this data to collect error rates.

(b) Mouse usage can be measured with a drawing program. Draw two circles on opposite sides of the screen. Select the "pencil" tool and ask the user to hold down the mouse while moving back and forth several times between the two targets. As long as the mouse is held down, this will lead to a set of trails similar to those found in Figures 12.10 and 12.11. Time the results. If you vary the distances between the targets and the size of the targets, you can run a Fitts' law study.

REFERENCES

Agrawal, R., Imieliński, T., Swami, A., 1993. Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. ACM, Washington, D.C., pp. 207–216.

Arroyo, E., Selker, T., Wei, W., 2006. Usability tool for analysis of web designs using mouse tracks. In: Extended Abstracts, ACM Conference on Human Factors in Computing Systems. ACM Press, New York, pp. 484–489.

Atterer, R., Wnuk, M., Schmidt, A., 2006. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In: Proceedings of the 15th International Conference on World Wide Web, Edinburgh, Scotland. ACM Press, New York.

Ben-Naim, D., Marcus, N., Bain, M., 2008. Visualization and analysis of student interactions in an exploratory learning environment. In: Proceedings of the 1st International Workshop on Intelligent Support for Exploratory Environments (Part of ECTEL 2008).

Burstyn, J., Carrascal, J.P., Vertegaal, R., 2016. Fitts' law and the effects of input mapping and stiffness on flexible display interactions. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, Santa Clara, CA, USA. ACM, New York, pp. 3649–3658.

Buschek, D., Auch, A., Alt, F., 2015. A toolkit for analysis and prediction of touch targeting behaviour on mobile websites. In: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Duisburg, Germany. ACM, New York, pp. 54–63.

Cangiano, G.R., Hollan, J.D., 2009. Capturing and restoring the context of everyday work: a case study at a law office. In: Kurosu, M. (Ed.), Human Centered Design: First International Conference, HCD 2009, Held as Part of HCI International 2009, San Diego, CA, USA, July 19–24, 2009 Proceedings. Springer, Berlin, pp. 945–954.

Canzian, L., Musolesi, M., 2015. Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan. ACM, New York, pp. 1293–1304.

Carta, T., Paternò, F., de Santana, V.F., 2011a. Web usability probe: a tool for supporting remote usability evaluation of web sites. In: Campos, P., Graham, N., Jorge, J., et al. (Eds.), Human-Computer Interaction—INTERACT 2011: 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5–9, 2011, Proceedings, Part IV. Heidelberg, Berlin, pp. 349–357.

Carta, T., Paternò, F., Santana, V., 2011b. Support for remote usability evaluation of web mobile applications. In: Proceedings of the 29th ACM International Conference on Design of Communication, Pisa, Italy. ACM, New York, pp. 129–136.

Clarkson, E., Lyons, K., Clawson, J., Starner, T., 2007. Revisiting and validating a model of two-thumb text entry. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA. ACM, New York, pp. 163–166.

Crabtree, A., Benford, S., Greenhalgh, C., Tennent, P., Chalmers, M., Brown, B., 2006. Supporting ethnographic studies of ubiquitous computing in the wild. In: ACM Conference on Designing Interactive Systems, University Park, PA, USA. ACM Press, New York.

Eirinaki, M., Vazirgiannis, M., 2003. Web mining for web personalization. ACM Transactions on Internet Technology 3 (1), 1–27.

Fan, Z., Song, X., Shibasaki, R., Adachi, R., 2015. CityMomentum: an online approach for crowd behavior prediction at a citywide level. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Osaka, Japan. ACM, New York, pp. 559–569.

Fielding, R., Reschke, J. (Eds.), 2014. Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. Retrieved from https://tools.ietf.org/html/rfc7231 (3.17.2017).

Fielding, R.T., Taylor, R.N., 2002. Principled design of the modern Web architecture. ACM Transactions on Internet Technology 2 (2), 115–150.

Fitts, P.M., 1954. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 47 (6), 381–391.

Fouse, A., Weibel, N., Hutchins, E., Hollan, J.D., 2011. ChronoViz: a system for supporting navigation of time-coded data. In: CHI'11 Extended Abstracts on Human Factors in Computing Systems, Vancouver, BC, Canada. ACM, New York, pp. 299–304.

Grossman, T., Kong, N., Balakrishnan, R., 2007. Modeling pointing at targets of arbitrary shapes. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA. ACM, New York, pp. 463–472.

Guo, Q., Agichtein, E., 2009. Beyond session segmentation: predicting changes in search intent with client-side user interactions. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, MA, USA. ACM, New York, pp. 636–637.

Hammontree, M.L., Hendrickson, J.J., Hensley, B.W., 1992. Integrated data capture and analysis tools for research and testing on graphical user interfaces. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Monterey, CA, United States. ACM Press, New York, pp. 431–432.

Harris, J., 2005. Inside Deep Thought (Why the UI, Part 6). Jensen Harris: An Office User Interface Blog. Retrieved from http://blogs.msdn.com/jensenh/archive/2005/10/31/487247.aspx (07.01.17).

Harris, J., 2006. No Distaste for Paste (Why the UI, Part 7). Jensen Harris: An Office User Interface Blog. Retrieved from http://blogs.msdn.com/jensenh/archive/2006/04/07/570798.aspx (07.01.17).

Heer, J., Chi, E.H., 2002. Separating the swarm: categorization methods for user sessions on the web. In: ACM Conference on Human Factors in Computing Systems, Minneapolis, MN, USA. ACM Press, New York, pp. 243–250.

Hilbert, D.M., Redmiles, D.F., 2000. Extracting usability information from user interface events. ACM Computing Surveys 32, 384–421.

Hochheiser, H., Shneiderman, B., 2000. Performance benefits of simultaneous over sequential menus as task complexity increases. International Journal of Human–Computer Interaction 12 (2), 173–192.

Hochheiser, H., Shneiderman, B., 2001. Using interactive visualizations of www log data to characterize access patterns and inform site design. Journal of the American Society for Information Science and Technology 52 (4), 331–343.

Hong, J.I., Heer, J., Waterson, S., Landay, J.A., 2001. WebQuilt: a proxy-based approach to remote web usability testing. ACM Transactions on Information Systems 19 (3), 263–285.

Hourcade, J.P., Bederson, B.B., Druin, A., Guimbretière, F., 2004. Differences in pointing task performance between preschool children and adults using mice. ACM Transactions on Computer-Human Interaction 11 (4), 357–386.

Huang, J., White, R.W., Buscher, G., Wang, K., 2012. Improving searcher models using mouse cursor activity. In: Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, Portland, OR, USA. ACM, New York, pp. 195–204.

Huang, J., White, R.W., Dumais, S., 2011. No clicks, no problem: using cursor movements to understand and improve search. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC, Canada. ACM, New York, pp. 1225–1234.

Humayoun, S.R., Dubinsky, Y., Catarci, T., Nazarov, E., Israel, A., 2012. A model-based approach to ongoing product evaluation. In: Proceedings of the International Working Conference on Advanced Visual Interfaces, Capri Island, Italy. ACM, New York, pp. 596–603.

Hurst, A., Hudson, S.E., Mankoff, J., Trewin, S., 2013. Distinguishing users by pointing performance in laboratory and real-world tasks. ACM Transactions on Accessible Computing 5 (2), 1–27.

Iqbal, S.T., Horvitz, E., 2007. Disruption and recovery of computing tasks: field study, analysis, and directions. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA. ACM Press, New York, pp. 677–686.

Ivory, M.Y., Hearst, M.A., 2001. The state of the art in automating usability evaluation of user interfaces. ACM Computing Surveys 33 (4), 470–516.

Janzen, I., Rajendran, V.K., Booth, K.S., 2016. Modeling the impact of depth on pointing performance. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, Santa Clara, CA, USA. ACM, New York, pp. 188–199.

Kairam, S., Brzozowski, M., Huffaker, D., Chi, E., 2012. Talking in circles: selective sharing in Google+. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, TX, USA. ACM, New York, pp. 1065–1074.

Kiciman, E., Livshits, B., 2010. AjaxScope: a platform for remotely monitoring the client-side behavior of web 2.0 applications. ACM Transactions on the Web 4 (4), 1–52.

Koster, M., 2007. Robots Exclusion. Retrieved from http://www.robotstxt.org/ (07.01.17).

Lin, M., Goldman, R., Price, K.J., Sears, A., Jacko, J., 2007. How do people tap when walking? An empirical investigation of nomadic data entry. International Journal of Human–Computer Studies 65 (9), 759–769.

Lubos, P., Bruder, G., Steinicke, F., 2014. Are 4 hands better than 2?: bimanual interaction for quadmanual user interfaces. In: Proceedings of the 2nd ACM Symposium on Spatial User Interaction Honolulu, HI, USA. ACM, New York, pp. 123–126.

MacKenzie, I.S., 1992. Fitts' law as a research and design tool in human-computer interaction. Human–Computer Interaction 7 (1), 91–139.

MacKenzie, I.S., Buxton, W., 1992. Extending Fitts' law to two-dimensional tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Monterey, CA, United States. ACM, New York, pp. 219–226.

Malik, S., Koh, E., 2016. High-volume hypothesis testing for large-scale web log analysis. In: Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, Santa Clara, CA, USA. ACM, New York, pp. 1583–1590.

Mark, G., Iqbal, T.S., Czerwinski, M., Johns, P., Sano, A., Lutchyn, Y., 2016. Email duration, batching and self-interruption: patterns of email use on productivity and stress. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, Santa Clara, CA, USA. ACM, New York, pp. 1717–1728.

Mavrikis, M., Zhu, Z., Gutierrez-Santos, S., Poulovassilis, A., 2015. Visualisation and analysis of students' interaction data in exploratory earning environments. In: Proceedings of the 24th International Conference on World Wide Web, Florence, Italy. ACM, New York, pp. 1419–1424.

McGrenere, J., Moore, G., 2000. Are we all in the same “Bloat”? In: Proceedings of Graphics Interface, pp. 187–196.

McGrenere, J., Baecker, R.M., Booth, K.S., 2002. An evaluation of a multiple interface design solution for bloated software. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Minneapolis, MN, USA. ACM Press, New York.

Nguyen, E., Modak, T., Dias, E., Yu, Y., Huang, L., 2014. Fitnamo: using bodydata to encourage exercise through Google glass™. In: CHI'14 Extended Abstracts on Human Factors in Computing Systems, Toronto, Ontario, Canada. ACM, New York, pp. 239–244.

Obendorf, H., Weinreich, H., Herder, E., Mayer, M., 2007. Web page revisitation revisited: implications of a long-term click-stream study of browser usage. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, CA, USA. ACM Press, New York, pp. 597–606.

Pirolli, P., Pitkow, J., 1999. Distributions of surfers' paths through the World Wide Web: empirical characterizations. World Wide Web 2 (1), 29–45.

Srivastava, J., Cooley, R., Deshpande, M., Tanw, P.-N., 2000. Web usage mining: discovery and applications of usage patterns from Web data. SIGKDD Explorations Newsletter 1 (2), 12–23.

Teather, R.J., Stuerzlinger, W., 2014. Visual aids in 3D point selection experiments. In: Proceedings of the 2nd ACM Symposium on Spatial User Interaction, Honolulu, HI, USA. ACM, New York, pp. 127–136.

Teevan, J., Dumais, S.T., Horvitz, E., 2005. Personalizing search via automated analysis of interests and activities. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil. ACM, New York, pp. 449–456.

Terry, M., Kay, M., Van Vugt, B., Slack, B., Park, T., 2008. Ingimp: introducing instrumentation to an end-user open source application. In: Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy. ACM, New York, pp. 607–616.

Velloso, E., Alexander, J., Bulling, A., Gellersen, H., 2015. Interactions under the desk: a characterisation of foot movements for input in a seated position. In: Abascal, J., Barbosa, S., Fetter, M., et al. (Eds.), Human-Computer Interaction—INTERACT 2015: 15th IFIP TC 13 International Conference, Bamberg, Germany, September 14–18, 2015, Proceedings, Part I, Cham. Springer, New York, pp. 384–401.

White, R., 2013. Beliefs and biases in web search. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland. ACM, New York, pp. 3–12.

White, R.W., Hassan, A., 2014. Content Bias in Online Health Search. ACM Transactions on the Web 8 (4), 1–33.

White, R.W., Horvitz, E., 2009. Cyberchondria: studies of the escalation of medical concerns in Web search. ACM Transactions on Information Systems 27 (4), 1–37.

Whittaker, S., Matthews, T., Cerruti, J., Badenes, H., Tang, J., 2011. Am I wasting my time organizing email? A study of email refinding. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Vancouver, BC, Canada. ACM, New York, pp. 3449–3458.

Wollner, P.K.A., Langdon, P.M., Clarkson, P.J., 2015. Integrating a cognitive modelling framework into the design process of touchscreen user interfaces. In: Marcus, A. (Ed.), Design, User Experience, and Usability: Users and Interactions: 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2–7, 2015, Proceedings, Part II. Springer, Cham, pp. 473–484.

World Wide Web Consortium, 1995. Logging Control in W3C httpd. Retrieved from http://www.w3.org/Daemon/User/Config/Logging.html (29.11.15).