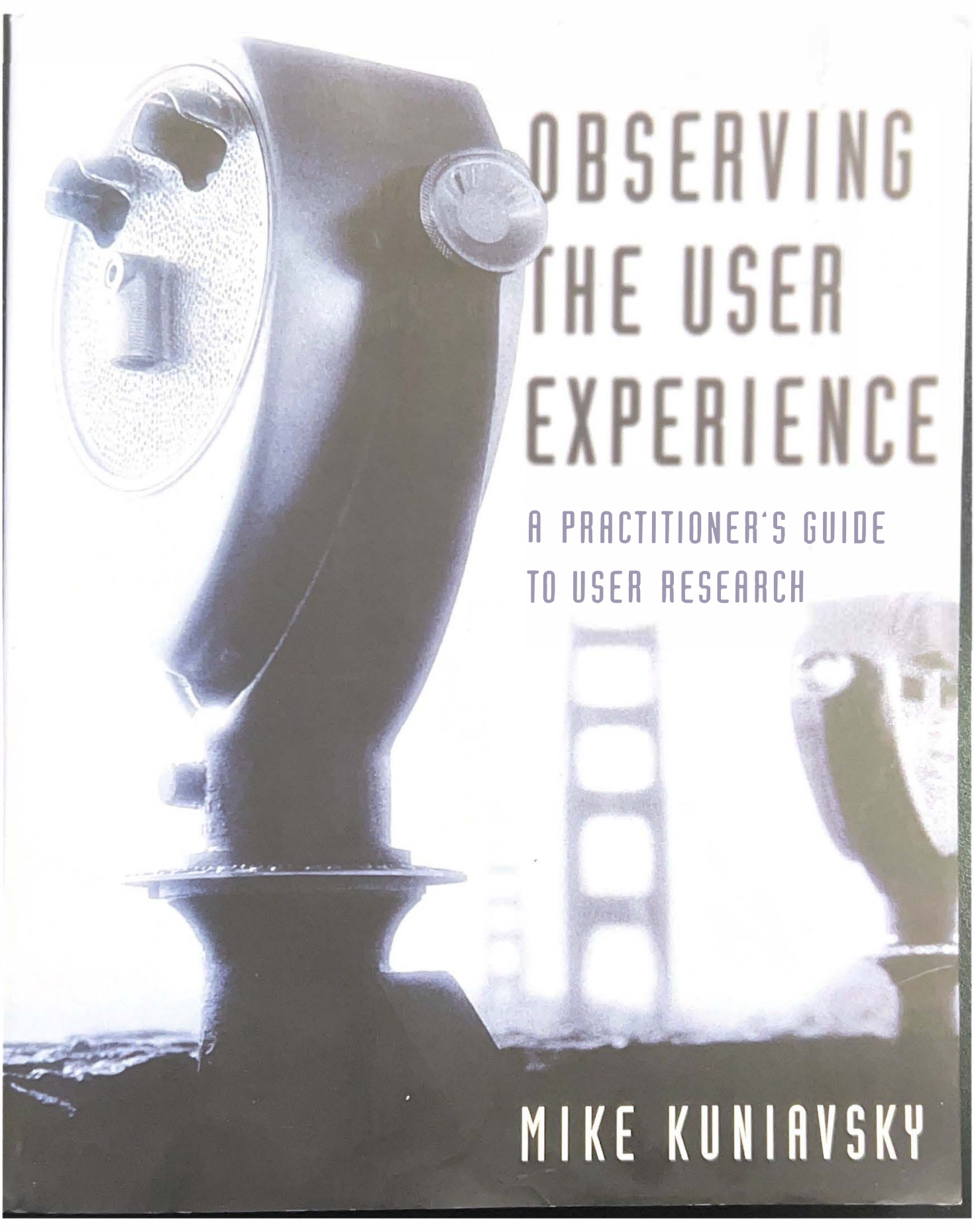




OBSERVING THE USER EXPERIENCE

A PRACTITIONER'S GUIDE
TO USER RESEARCH

MIKE KUNIAVSKY

The Morgan Kaufmann Series in Interactive Technologies

Series Editors:

- Stuart Card, PARC
- Jonathan Grudin, Microsoft
- Jakob Nielsen, Nielsen Norman Group

Observing the User Experience: A Practitioner's Guide to User Research

Mike Kuniavsky

Getting the Work Right: Interaction Design for Complex Problem Solving

Barbara Mirel

The Craft of Information Visualization: Readings and Reflections

Edited by Ben Bederson and Ben Shneiderman

HCI Models, Theories, and Frameworks: Towards a Multidisciplinary Science

Edited by John M. Carroll

Web Bloopers: 60 Common Web Design Mistakes, and How to Avoid Them

Jeff Johnson

Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces

Carolyn Snyder

Persuasive Technology: Using Computers to Change What We Think and Do

B.J. Fogg

Coordinating User Interfaces for Consistency

Edited by Jakob Nielsen

Usability for the Web: Designing Web Sites that Work

Tom Brinck, Darren Gergle, and Scott D. Wood

Usability Engineering: Scenario-Based Development of Human-Computer Interaction

Mary Beth Rosson and John M. Carroll

Your Wish is My Command: Programming by Example

Edited by Henry Lieberman

GUI Bloopers: Don'ts and Dos for Software Developers and Web Designers

Jeff Johnson

Information Visualization: Perception for Design

Colin Ware

Robots for Kids: Exploring New Technologies for Learning

Edited by Allison Druin and James Hendler

Information Appliances and Beyond: Interaction Design for Consumer Products

Edited by Eric Bergman

Readings in Information Visualization: Using Vision to Think

Written and edited by Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman

The Design of Children's Technology

Edited by Allison Druin

Web Site Usability: A Designer's Guide

Jared M. Spool, Tara Scanlon, Will Schroeder, Carolyn Snyder, and Terri DeAngelo

The Usability Engineering Lifecycle: A Practitioner's Handbook for User Interface Design

Deborah J. Mayhew

Contextual Design: Defining Customer-Centered Systems

Hugh Beyer and Karen Holtzblatt

Human-Computer Interface Design: Success Stories, Emerging Methods, and Real World Context

Edited by Marianne Rudisill, Clayton Lewis, Peter P. Polson, and Timothy D. McKay

Observing the User Experience

A Practitioner's Guide to User Research

Mike Kuniavsky



MORGAN KAUFMANN PUBLISHERS

AN IMPRINT OF ELSEVIER SCIENCE

SAN FRANCISCO SAN DIEGO NEW YORK BOSTON
LONDON SYDNEY TOKYO

Publishing Director
Assistant Publishing Services Manager
Senior Project Manager
Editorial Assistant
Design
Illustrations
Project Management and Composition
Cover Design
Copyeditor
Proofreader
Indexer
Printer
Cover Printer
Cover Image

Diane D. Cerra
Simon Crump
Julio Esperas
Mona Buchler
Frances Baca Design
Dartmouth Publishers
Top Graphics
Yvo Riezbos Design
Robert Fiske
Jennifer McClain
Ty Koontz
The Maple-Vail Book Manufacturing Group
Phoenix Color
Viewing binoculars, Golden Gate Bridge in background (B&W),
The Image Bank, Bill Helsel

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances in which Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Morgan Kaufmann Publishers
An Imprint of Elsevier Science
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205, USA
www.mkp.com

© 2003 by Elsevier Science (USA)
All rights reserved.
Printed in the United States of America

07 06 05 04 03 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, or otherwise—without the prior written permission of the publisher.

Library of Congress Control Number: 2002115471

ISBN: 1-55860-923-7

This book is printed on acid-free paper.

CHAPTER 7

User Profiles

“Who’s going to use your product?”

“Everyone!”

“And what will they do with it?”

“Everything!”

As technology and bandwidth allowed for real-time voice communication over the Internet, a bunch of companies entered the market to promote and develop it. Many decided to aim big; otherwise, the reasoning went, why aim at all? When asked about who would use their product, an executive at one of these companies replied, “Everyone who uses a telephone!”

Although at the time that was a great ambition to have, and a good story to tell investors, the group of people who would have been likely to use an Internet telephone in 1998 was a lot smaller than “everyone who uses a telephone.” By defining the audience so broadly, they were, in effect, not defining it at all. Of course, they could have claimed to be happy if they captured a small fraction of the large market, but that reasoning would have been equally flawed. Aiming for a small share of a large market would have created an impossible task for the company. To capture even a small piece of a big market, they would have needed to understand the dynamics of the *whole* market. There are a lot of telephone users in the world. To understand the market they had so flippantly defined, the Internet telephone companies would need to know the needs, abilities, and desires of a large chunk of the world’s population—in a 1.0 product.

The company in question eventually realized that they would have to focus their efforts. After trying to make a completely generic product and failing, they settled on a more focused audience definition, but only after spending a long time working on features that most of their (few) users didn't care for and that didn't make sense when taken together with the rest of the product.

This is where user profiles would have been valuable. When you create *user profiles* (or *personas*, as Alan Cooper, founder of Cooper Interaction Design, a major innovator and proponent of the technique, calls it), you create one or more fictitious users and try to model them and their lives. From general characteristics, you model specific individuals who would buy and use your stuff. Each final profile is a tool that helps you define and compare your ideal with reality. Role playing with user profiles can reveal functionality and needs that would be difficult to uncover otherwise, while adding a dose of reality to product feature brainstorming sessions ("Keeping the 'nifty factor' down," as Christina Wodtke, senior interaction designer at Yahoo!, puts it). User profiles allow you to focus on specific images of users that everyone on the team can reference and use as sounding boards for development ideas. They're also much simpler to work with than "everyone."

The bulk of this book is about understanding who your users are, what they can do, and what they want. That information is going to guide your product in the long run, but even before you start collecting hard data, you should begin focusing on the user and his or her experience. In *user profiling*, you model your users based on your intuition, your judgment, and the information you have at hand. This quickly gives you insight into what makes a good user experience and can keep your product out of the "everything for everyone" trap. Profiles provide a conceptual envelope to work in and benchmarks against which to focus the rest of your research. When you follow up with surveys, contextual inquiry, or user tests, you have a model with which to compare your results.

Except for the fact that they're fictional, your user profiles are people whom you will get to know, to spend time with, to like. They are more than just a face put on a list of demographic data; they are entities whom you work with to make a better product.

distinction should be drawn when creating a profile. Desires, when fulfilled, may not make the product any more effective or efficient, but they may make the user experience more satisfying, so they need to be taken into account.

- *Stated desires.* What do your users say they want?
- *Unstated desires.* What do they really want?

Knowledge

As they say in *noir* detective stories, “What did she know and when did she know it?” The back story tells you how people got to your Web site and why they ended up there. There are plenty of things she could have been doing on that sultry August afternoon when she walked into your office, er, clicked to your site. So what is the context in which you find “her” on your doorstep?

- *Domain knowledge.* How much does your audience know about what they’re trying to do? Are they experts trying to find a more efficient tool or newbies who need to be taught about what’s going on?
- *Product knowledge.* How much do your customers know about your product or service? How much do they know about their own needs? About the options available to them?
- *Competitive awareness.* How aware are they about the various brands competing with yours? How much do they care?

Usage Trends

The patterns in which people use a product determine much about what they want and expect from it.

- *Frequency.* Will they use your service just once, or will it be a regular part of their job cycle?
- *Considerations.* Will they do a lot of research before settling on a service the first time they use it? Will they do this research every time, or will they just do it once?
- *Loyalty.* How loyal are they to their chosen product? Do they just choose the cheapest or most convenient one, or do they stick with one for a while?

Tasks

Ultimately, people are using your product to do *something*. What? Tasks are actions that users do to move themselves closer to their goals. For the purposes of the user profile, tasks should be seen from the user's perspective. They should be considered at the granularity at which the user thinks. Users rarely think of database row locking (as a software engineer would); they think of transferring money to their checking account or making a hotel reservation. The profile should contain a list of the most important tasks that the user is going to want to do with the system (whether or not the current technical implementation can do these tasks).

- *Reason*. Why do they want to do this?
- *Duration*. How long do they do it for?
- *Order*. Which tasks come after which other ones? Where does the user think the sequence matters? Where does it not matter? How are tasks linked?
- *Criticality*. How important are the tasks? Are they rare, but extremely important, tasks? For example, data recovery should not happen frequently, but it's very important when it does.
- *Method*. How do they want to accomplish the task?
- *Model*. Is there some way they're doing this already? What tools are they using, and how are they using them?

As you're creating tasks, you may notice clusters that are completely independent for otherwise identical profiles. For example, almost all tasks done by people selling homes and people buying homes are completely different although they're related topically, and the people doing the buying and selling are pretty similar. This is an important difference to recognize. The two clusters do more than just define two different profiles; they define two different *products*.

Cluster the Attributes

Once you've gathered the giant list of attributes, it's time to cluster them into profiles. This is done in a similar way to the Beyer and Holzblatt *affinity diagrams* used in contextual inquiry, but somewhat simpler.

1. Start by picking someone and asking him to put his attribute Post-its into several clusters he feels may make useful user profiles. For an online banking site, he may want one cluster with "40-something," "male," "vice president," "Internet newbie," and "responsible for corporate budget," as well as another cluster with "40-something," "female," "store manager," "does books every night." Although everyone started out with an image of his or her audience, this will likely produce multiple profiles.
2. Ask another participant to put her notes that relate to the clusters around them one at a time. It doesn't matter how the notes relate, just as long as the participant feels that they do relate. If necessary, have people copy their notes so that attributes can exist in more than one profile.
3. If a participant has notes that don't fit one of the existing clusters, put those up as new clusters.
4. After each person has put his or her attributes up, discuss them with the group. Do they make sense where they are? Why are they there? Move notes around until you have clusters that everyone agrees on.
5. Repeat with everyone in the group.

Warning Be wary of bias creeping into the profile when participants with an agenda or disproportionate status (say, a senior vice president in a group that comprises primarily junior staff) emphasize a particular view of users. If there are clear contradictions in the profiles, attempt to talk through them, stress that user markets are not the same as target markets, and that unless there's hard data, it's mostly conjecture.

This may take several hours since there may be a lot of attributes and because the discussion can uncover assumptions about what users want and what they're going to be doing.

There may be clusters that define prominent user markets and smaller clusters that define potential markets or secondary user profiles. The clusters may be similar to the target market segments that business development or marketing has created but with greater precision in terms of the audiences' relationship to your product.

Once the attributes are put up on the board, stop and discuss the trends that you see in the clusters. Are there obvious trends? Do they fit the user types that the participants had expected? Where are the surprises?

If there are many different clusters, winnow the number of profile clusters down. Keep the number of major clusters between three and eight. More than eight clusters will probably be unwieldy; fewer than three will not provide enough diversity for the profiles to be useful.

in the front line of collecting user feedback. If customer service has an existing classification for their questions, these content-based categories may help organize future comments, while insights into user profiles and mental models can help them formulate answers. Once the analysis has been done one time, regularly revisiting support comments makes it easier to keep tabs on how the audience changes and how they react to site modifications.

But people's perceptions of the problems they face may not reflect the actual core problems in their experience. Knowing how they actually behave, and comparing that to how they're expected to behave, can expose the places where things are *really* going wrong. For this, you can turn to *log files*.

Log Files

Whenever someone gets a page from a Web site, the server makes note of it. In a traditional store or office, the management can only guess where people go and what they look at, but Web servers know exactly who is looking at what when (though, of course, looking doesn't guarantee understanding or interest). A Web site visitor has to ask to see every piece they're interested in, like in a jewelry store. This is different from a supermarket, where he or she can spend all day squeezing every tomato, and no one would ever know. This is why a clerk at a jewelry store is likely to have a much better understanding of customers' behavior than a supermarket cashier.

Web servers have the ability to collect a surprisingly large amount of information. For every request (including every page, every image, every search), they store the address of where the request came from, what was requested, when it was requested, what browser was doing the requesting, what operating system the browser was running on, and a number of other facts about the connection. This produces a lot of data, and it's not unusual for busy sites to produce gigabyte-sized log files on a daily basis.

So how is this data mountain mined? Carefully. Up-front planning can reduce the vertigo that comes from confronting the task of data mining. As you answer initial questions, you discover what other information should be examined, collected, and processed.

What's in a Log File, and What's Not

A log file is a raw record of everything that the programmers felt warranted being recorded. It's like the program's stream of consciousness. Web servers are capable of logging all kinds of information and interacting with all kinds of software, so it's possible to have a lot of different log files. Some may be lists of purchases, some may be records of authenticated logins, and others are litanies of internal software errors. The type of log file that generally provides the most user experience information is the *access log*. This file contains a record for each item requested from the server with as much information about the user's browser as the Web server is able to gather.

Here are some elements that are found in many Web log files.

TYPICAL ACCESS LOG ELEMENTS

- The *IP address* of the computer that made the request (most servers can be set to resolve these numeric addresses such as 192.0.34.72 into human-readable DNS names such as *www.example.com*).
- The *date and time* the request was made.
- The *kind of request* made (whether to get an HTML page, post a form response, etc.).
- The *name of the file* that was requested. This takes the form of everything in the URL that was requested that comes after the domain name and often includes query strings to dynamic pages generated by content management systems.
- The *server status* code. This is a numeric code that represents whether the server was able to deliver the file or whether it was redirected or produced an error. The infamous "404 File not found" is one such error.
- The *user agent*, information identifying the browser and operating system used by the client.
- *Cookies* sent by the client. Cookies are ways that a client program (such as a browser) sends the server its identity.
- The *referrer*. The page and site that the client viewed immediately before making the current request.

For example, here is a typical access log entry requesting a page called `example.html`.

```
192.0.34.72 - - [05/Apr/2002:04:50:22 -0800] "GET /example.html
HTTP/1.1" 200 18793
```

(Continued)

And here is a referrer log entry that says that the last page visited (likely the page that linked to the current page) was a Google search result page for the word *adaptive* and the page that was fetched was */adapt.html*.

<http://www.google.com/search?hl=en&q=adaptive->/adapt.html>

There are several excellent articles on understanding and interpreting log files on Webmonkey.

<http://hotwired.lycos.com/webmonkey/e-business/tracking/index.html>

With all this information, it's theoretically possible to get a precise idea of who asked for what when. However, log files are not without their problems.

First of all, when a browser requests a Web page with graphics, it doesn't make a single request. It requests every element on the page separately. Each graphic and HTML segment is asked for and logged separately. Thus, what looks like a single page can produce a dozen or more requests and log entries. Similarly, when a page contains frames, the browser typically asks for the frameset (the document that defines how the frames are visually displayed) and the HTML files (and all of their contents) individually. This can potentially create a score of log entries and a confusing interlacing of framesets, navigation, and content pages, all in response to a single click by the user.

Server logs also don't record whether a browser stopped the transfer of a file before it was completely delivered. When looking at a standard log file, it's impossible to tell whether people are abandoning a page because there's too much content or if they're bailing because the server is too slow. Without a packet sniffer (a piece of equipment that monitors all information coming in and out of the server), you know a page was requested, but you don't know whether it ever arrived.

Dynamically generated sites (typically database-driven or content management system-driven sites) can also produce confusing log files since the logs may only contain the fact that a certain piece of software was invoked, not what the software returned. Unless the software is set up appropriately, all access to the dynamic content of a site can look as if it's going to a single page.

The most severe problems come from the *caching* of Web pages. Caching is the process of storing files in an intermediate, quickly ac-

cessible location so that they don't have to be downloaded from the server again. This improves the user's browsing experience immensely since clicking a back button does not cause the previous page to download again because the files get fetched from the local hard disk instead. Though great from a user perspective, this introduces a level of complexity into analyzing site logs. When someone clicks the back button to get to a previous page or uses his or her browser's history mechanism, the browser may get the page from a cache and never contact the server when showing the page. From the perspective of the server, it may look like someone is spending a long time looking at one page, when in fact he or she is in a completely different section of the site.

There are two kinds of caching: *personal* and *institutional* (see Figure 13.1). All common Web browsers have some sort of personal caching mechanism built in: they save the last several days' or weeks' worth of browsing into a cache on the local disk. Modern browsers have a setting that allows them to check for updated versions of pages, and most have it set on by default, so they'll often do a quick check with the server to make sure that the HTML page they're about to load hasn't been updated since they put it into their cache. This solves the most severe of the problems though there are still situations where the browser never gives the server any indication that the user is looking at a different page.

Institutional caches are more problematic. These are caches maintained by computers that sit between a user's machine and the Web server and intercept browser requests before they get to the server. They make local copies of Web site data that the browser requested. When another request comes along for data they have stored, they serve it directly rather than passing the request along to the server where the information originally came from.

From a user's perspective, this saves bandwidth and can improve the apparent speed of a site, but it circumvents any way for the site's server to track usage, and adds an element of uncertainty to all log analysis. For example, AOL has a large institutional cache for all its users. When an AOL user requests a page that was requested by someone else in the recent past, the AOL cache will serve the page to the user directly, bypassing the "home" server. The site's original server may *never* know that someone is looking at their content since it will never have been contacted. Hundreds of AOL users may be

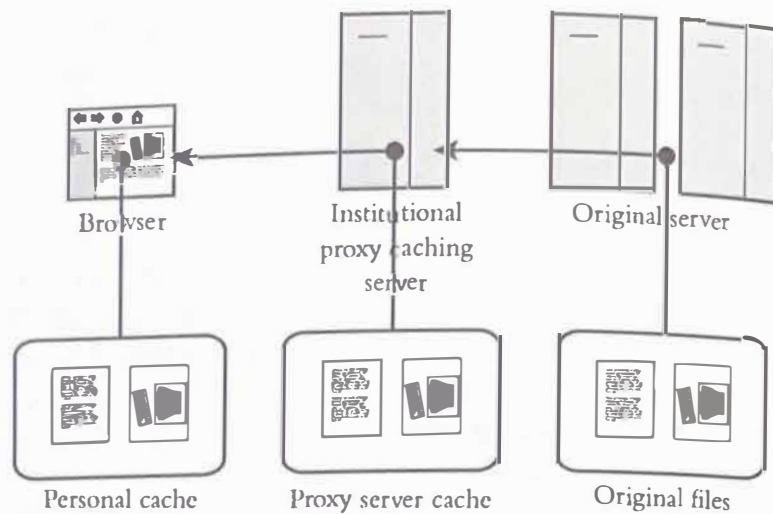


Figure 13.1 How Web files are cached.

looking at your site, but your Web server will have registered only a single visit, the original one that got its content into AOL's cache.

This is not to vilify caching. It provides many benefits in terms of the user experience, and it creates a more efficient Internet, but it does make getting perfect statistics about usage impossible since some amount of traffic will always be obscured by caches. This quality makes log analysis more of a qualitative analysis technique than a quantitative one.

For a sales rep who needs to know how many AOL users clicked on an ad banner, exact numbers are important, but fortunately when it comes to understanding the user experience, exact counts are not strictly necessary. Treating log files as a random sample of your users' behavior is generally sufficient: you need to know how people behave and how their behavior is affected by changes to the site, and a random sample can tell you that. Knowing the *exact* number of people who behave in a certain way is rarely useful for drawing general user experience conclusions.

Logs and Cookies

If only anonymous use logs are collected, it's impossible to say anything particularly subtle about people's behavior. Because the logs

just collect information about which files were requested, it's only possible to tabulate conglomerate statistics about the overall popularity of pages. To allow deeper insight into information such as the order people visit pages, how long they spend on each page, and how often they visit the site, it's necessary to isolate individuals in the log files and to track those individuals across visits.

Originally, individuals were tracked by keeping a note of unique IP addresses, but by using this method, it's difficult to track different individuals. Even if people are identified, it's almost impossible to track the same person over different sessions. So Web cookies were invented. A Web cookie is a token—generally, a tiny file—that's exchanged by a browser and a Web server, and every time the browser connects to the server, it sends the file. For most cookies, a unique identification code embedded in the cookie identifies it as belonging to a specific visitor's browser. Cookie identification is commonly used for personalization (Amazon, for example, uses them to identify you so that it can make recommendations) and order tracking, but they're also useful for tracking user behavior. The same cookie that lets Amazon give you a personalized experience can be used to see how often you visit their site, what you've looked at, and what you've bought.

Like their edible namesakes, cookies have expiration dates. These dates can range from the short term ("10 seconds from now") to several years. Although primarily a security feature, expiration dates can be used to maximize their utility as tools for understanding how users use the site. One trick that makes user behavior analysis easier is to use two different kinds of cookies with two different expiration times.

- *Session cookies*, with expiration times between a couple of minutes and several hours, identify an individual session. If no page has been fetched at a search engine site in 10 minutes, for example, it's likely that the user has finished using it. Likewise, a newspaper site may want to give people 30-minute session cookies, whereas an online game might give ones that last several hours.
- *Identity cookies* have much longer expiration times, on the order of months or years. These cookies identify a single user and can be used to track that person's behavior over multiple sessions.

Note Cookie-based session tracking is built into both Apache and the Microsoft IIS server, and just needs to be turned on for cookie information to start appearing in your log files. Please see your server software documentation to see how to do this.

Information about the Apache user-tracking module (which uses cookies) is available at http://apache.org/docs/mod/mod_usertrack.html.

An article describing Microsoft IIS 4.0 user tracking is available at msdn.microsoft.com/library/default.asp?url=/library/en-us/dnw2kmag01/html/EventLogging.asp.