Mining Gold from Server Logs

Using existing site data to improve Web testing

by Karen Johnson

QUICK LOOK

- Organizing testing based on user statistics
- Determining realistic load tests

How often have you wished you knew how your customers *really* use your Web site?

That information actually exists—it's just lying untapped in your Web server's logs. Those records reveal the reality of how your site is used, and can provide a great deal of data if you know how to mine it. Your quality assurance team can use this information to organize browser testing based on user statistics, improve testing coverage of your Web site, and plan more realistic load testing.

Let's look at just what information can be pulled from Web server logs, and how that information is used at one online company: Peapod, an e-grocer serving multiple metropolitan cities in the United States. In addition to local delivery, Peapod ships nonperishable food from its "Packages" store to almost anywhere in the United States. Peapod has been in business eleven years—long before the "dot-com" craze began. Customers used to receive diskettes with Peapod's software and connect via modem to place their orders. But now with the Internet open for business, Peapod is dependent on its Web site to keep the business alive and thriving. Peapod management knows their Web site has to be easy to navigate, and that performance time has to be at least up to standard for the industry. One tactic to keep quality high is to use the Web server logs to help the development staff and other internal departments stay in touch with the customer experience. Some of the statistics contained in the Web server logs and how that information is used at Peapod are presented in this article.

Narrow Your Focus

There is always too much testing to get through in the time available for testing. Add up all the combinations of operating systems and browser versions your customers may be using, and it's not possible to test *all* the features of your Web site in *every* configuration. Fortunately, you can apply two high-level strategies to cut to the core of what needs to be tested.

One strategy is to identify which browser and operating system combination is the most popular among

your customers (you can determine the most typical configurations from your Web server logs).

A second strategy is to get organized about what aspects of your Web site are vulnerable to variances in either the operating system or the browser version (or the combination of the two) and plan testing accordingly. This second strategy is often referred to as risk-based testing. By determining the most common configurations and focusing your testing efforts based on risk, your quality assurance team can optimize their testing efforts.

For example, at Peapod our customers use more than 700 combinations of operating systems and browsers. A majority of them, however, are using one of two setups: 48% of them run Microsoft's Internet Explorer 5x on some version of Windows, and another 26% use Netscape 4x on Windows NT. All the other combinations are in the low single digits. So when testing time is limited and testing effort has to be cut, QA focuses on those two configurations. We often eliminate either Win95 or Win98 from the testing configurations because we haven't found IE bugs that exist on Win95 that we didn't also encounter on Win98. The time spent on the remaining combinations is based on customer usage statistics. This efficient use of testing coverage would not be possible without using the knowledge gained from the Web server logs.

At Peapod, we regularly test against the browser and operating system combinations found in Table 1. We decided on these specific versions based on our server logs. We also keep an eye on the download counts at **www.browsers.com**—a CNET site that offers free downloads of MSIE and Netscape browsers and records how many downloads of each its site has provided. We use their data as a gauge of what the most common browser versions are in the general population, and we also use their site as a place to download the installs of new browser versions.

Even if we restricted testing to the top environments, we still wouldn't have time to run *all* our tests against each of the configurations. Fortunately, we don't need to. Some testing is not Web specific. For instance, suppose we added a new data entry field to an existing online registration page, and the data in that field was verified on the server (rather than, for example, by using JavaScript in the browser). The verification and new "write" operation to the database happen entirely on the server, so we know the new field will not be impacted by the user's operating system or browser version. Verifying the new field in one test environment is sufficient. Identifying what aspects of your site only need to be tested in one environment is a tremendous timesaving strategy.

But some Web testing needs to be performed on multiple environments because the operating system and browser combination can have an impact on whether the new feature works correctly. These are the testing activities that have to be assessed more carefully, since this testing will impact the QA time significantly.

So what types of features require testing in multiple browser and operating system combinations? I've never found a standard list that answered that question, so I invented one of my own, based on my testing experiences at Peapod. I use this seven-item list as a point of conversation between Peapod's Development and Quality Assurance departments when discussing the testing strategy for each release. Use this list as a starting point; I'd recommend adding to the list based on your own testing experiences.

1. **JavaScript** executes code in the browser. This code is browser specific, so you'll need to identify all the places JavaScript is used within your site (e.g., button clicks, image clicks, field validation, and form validation). If your developers have not specified where JavaScript is used, you can check the source to search for it: access a page, use the browser menu option to **View Source,** then use **Find** and search for *javascript.* You'll especially want to check pages that have forms, buttons, and images. Your team needs to test each use of JavaScript on each of the browser and OS combinations listed in Table 1 and should also test with JavaScript disabled in the browser.

2. Consider the **forms** you use carefully. If your forms use JavaScript for form validation, those forms need to be tested on each of the Table 1 combinations. (The general warning on JavaScript isn't enough; you have to pay special attention to forms, because every form adds a twist. Focusing only on JavaScript might tempt you to think only of clickable images and field-level validation.)

3. **Pages heavy with HTML formatting** may appear differently in IE than in Netscape. We check formatting against one version each of IE 4, IE 5, and Netscape 4. It's a quick look/see, not a major time-consuming test.

4. **User Interface:** If your site uses Cascading Style Sheets, Active X Components, other newer technologies, or browser plug-ins, your testing coverage needs to watch these areas of your site. You need to test in browsers that support the newer technologies, as well as in browsers that don't.

5. **Security:** If your site has a secure checkout or a credit card page, you'll want to verify that users can access those pages in secure mode. Since security is a critical component of any e-commerce site (and because secure pages can behave differently based on operating system and browser combinations), secure pages need to be checked carefully on each environment. Our goal is to check security against one version each of IE 4, IE 5, and Netscape 4—on both Win98 and WinNT. We don't insist on testing multiple versions of Netscape 4 for security issues; our experience has been that JavaScript bugs are more of a problem than security bugs in the minor versions of the Netscape browsers. [Note that there are two security protocols: SSL (secure sockets layer) and HTTPS (secure HTTP), both of which provide data encryption and user authentication. Secure pages are signified by the inclusion of *https* in the URL of the page and by a small closed padlock symbol at

the bottom of a Netscape browser (or unbroken key on an MS IE browser).]

6. If your site uses **pop-up windows,** you will need to test these carefully on each browser. You should be aware that WebTV does not handle pop-up windows (or support a mouse).

7. If your site uses **cookies,** testing coverage needs to cover one version each of IE 4 and IE 5 and one version of Netscape 4. Testing with cookies disabled should also be covered.

In addition to new features for a release, we also plan regression testing for each release. At Peapod we make sure that Web site features such as *Find Item* or *Checkout* are re-tested in every release. We create a coverage matrix that shows which features must be tested against which environments.

What do we do if we don't have time to complete the entire matrix—or even the core environments? In those situations our team focuses on a combination of *popular* and *fragile* environments. For example, we would test against IE 5 because it's the most common environment. We'd also test against WebTV—even though fewer than 1% of our users use it—because of its fragility (since our experience has shown that a feature that works on WebTV is likely to work elsewhere).

Recognizing Site Patterns

Log statistics can help you determine realistic site patterns. The logs' data reveals which URLs are visited the most frequently, and which URLs' users are least "sticky"—those pages most likely to be the ones from which your users leave to surf other pages. Your team can review those "high exit" pages and do an assessment. Do you need to improve graphics or content on that page? Can you verify that performance is acceptable for those URLs? Are there any other usability issues?

At Peapod, we have a logging mechanism that enables us to track which features are used most often. We use this information to determine which features and functions users prefer. For example, our users can perform one task (finding items) by either searching (*Find Item*) or browsing (*Browse Aisles*). Whichever is the most commonly used feature will receive the most testing attention.

OPERATING SYSTEM	BROWSER
Windows NT	Any minor version of IE 5
Windows NT	Any minor version of IE 4
Windows NT	Netscape 4.75
Windows NT	Netscape 4.52
Windows 98	Any minor version of IE 5
Windows 98	Any minor version of IE 4
Windows 98	Netscape 4.75
Windows 98	Netscape 4.52
Windows 95	Any minor version of IE 5
Windows 95	Any minor version of IE 4
Windows 95	Netscape 4.75
Windows 95	Netscape 4.52
Macintosh	Any minor version of IE 5
Macintosh	Netscape 4.75
WebTV	WebTV browser

TABLE 1 Common browser and operating system combinations to test

Realistic Load Tests

You can also use the server logs to determine realistic load tests, by quantifying your site's daily statistics—such as the number of visitors per day, or the traffic patterns by time of day and day of week. When planning load tests, this information will help your team know what peak loads your site needs to handle today. You can use the statistics to know which URLs are visited most often, and include those URLs in your load test user scenarios. The statistics may also reveal the length of a typical user session, which can help to determine how long to sustain the load. (At Peapod, for example, our statistics show that most users spend twenty-two minutes per shopping session.)

Improving Searches

If your site uses a search facility, data from your server logs can go a long way toward improving your customers' experience—by helping you identify common search criteria. Your quality assurance team can then use this information as they try to mimic your user audience. Do most user sessions include one or more searches? Then include searching in a typical user scenario. What is the typical user behavior after a search? Do your customers typically purchase the product they find? Do they review the product details? If your site is a content-only site, do your users typically drill down for more information after a search? Take a look at your server logs' data to learn your users' behavior—and build your testing scenarios to mirror the user experience.

At Peapod, for example, we've found that hungry Peapod customers often mistype a search entry, correct the search criteria, and rerun the search. To mimic the typical user be-

havior, we test failed searches on regression tests of our site.

Even unsuccessful searches can provide quality assurance teams valuable information. Use the log information to determine which searches fail, or which searches return too many matches. When testing your search facility, include a search that has a match, a search with too many matches, and a search with no matches. Take a proactive approach to the failed searches; you may want to have your content department add words to the dictionary of available search words.

Your logs can also reveal the performance levels that your users experience as they perform their searches. Your development group can add a time stamp to each search query and a time stamp to each result. This information enables Quality Assurance and Development to be aware of current and realistic search performance—not just what's projected or estimated. This information can be used to ensure that appropriate hardware is in place for your production site. Your quality assurance team can use this performance information to test any changes made to the search facility, to ensure there is no decrease in search performance.

You may wonder why we concentrate so much on the search function (which our site calls *Find Item*). Shouldn't we care about performance everywhere?

This is one example of how we use our server logs as a guide for focusing our resources. While we certainly care about performance throughout our site, we know—because of the logs—that virtually *every* user session includes at least one search. That makes testing search performance especially important. Because we learned how to mine untapped data from our server logs, we don't have to make guesses about our customers' online experience and site needs; the trail of virtual breadcrumbs down our virtual grocery aisles tells the story. STQE

Karen Johnson (kjohns@peapod.com) *is the QA Manager of E-Commerce at Peapod, Inc., an online food grocer. Karen has eight years of QA experience with client-server and Web testing.*

How to Get Started on Pulling Data from the Logs

Web server logs grow at an amazing rate and left untreated in their natural state, the logs provide a daunting stack of files; but with some effort, the log data can be parsed and the information summed to provide valuable information. What if your company does not extract information from the logs? Here are three steps you can start with:

1. **Work with your network or operations staff to gain access to the Web server logs.**
2. **Import the logs into Excel. The Web server logs are typically maintained in a standard comma-separated value format.**
3. **Use Excel's math functions to determine sums and percentages.**

Weekly Statistics

These are some of the weekly statistics that are generated by the logs. This information is captured and then placed in an email that is distributed among several people and departments at Peapod (see this article's StickyNotes link for more details).

- **Number of hits for home page**
- **Number of successful hits for entire site**
- **Number of user sessions**
- **Average number of hits per day**
- **Average number of page views per day**
- **Average number of user sessions per day**
- **Average user session length**
- **Top entry and exit pages**
- **Activity by city**
- **Most active day of the week**
- **Top browsers**
- **Version breakdown by browsers**
- **Most used operating systems**
- **Activity by hour of the day**

Daily Statistics

- **Total simultaneous sessions**
- **HTML simultaneous sessions**
- **Daily summary**
- **Online traffic**
- **Order summary**
- **Thirty-day max sessions summary**
- **Thirty-day total sessions summary**
- **Number of users currently online**
- **Server status**
- **CPU load**
- **Database stats**

How Statistics Are Used Throughout Peapod, Inc.

Many statistics are captured by the Web server logs. Testers aren't the only people who use them. The following are just a few examples of how Peapod uses Web server log information.

Marketing

Banner ads: Ads will be placed with different discount amounts offered in different cities to assess the impact of the discount that is offered.

Web Design

Feature Logging: The log information reveals which feature consumers use more often. Designers pay attention to popular features and try to optimize the user experience. They may also try to make less popular features more popular.

Merchandising

At Peapod, a department referred to as Merchandising is responsible for the products that are available in the store and the products that customers can find via searching or browsing the aisles. Each search that is run by each customer is assigned one of several codes. These codes indicate whether a search had a successful match or not. If the search was not successful, one of several codes is assigned to categorize why the search failed. Totals and statistics are tracked by each code to determine the number of successful searches and failed searches. Failed search information is reviewed to determine if the dictionary of search words should be modified or expanded. Merchandising uses the failed search information to make common search terms produce more exact matches.

E-Commerce Development

User interface and design considerations are made based on which browsers are more popular.

Network/Operations

Network and operations staff use the time-of-day and day-of-week statistics to determine the best time to perform system updates. We know, from determining the slowest day of the week and the quietest time of the day, when the best time is to take our site down for maintenance.