NIST Special Publication 500-248
April 2001

Design of a File Format for Logging Website Interaction

John Cugini / cuz@nist.gov
Sharon Laskowski / sharon.laskowski@nist.gov
Information Technology Laboratory
National Institute of Standards and Technology (NIST)

Abstract

The logging of user behavior in support of web usability testing is constrained by the difficulty of capturing and analyzing large amounts of logged data. However, there is great potential for the development of tools to support automated recording and analysis, especially for remote or large scale testing. In this paper, we propose a format for the representation of user interaction with a website. A widely accepted format enables the development of a set of software tools to process the data, the sharing of data sets for longer term analysis and research, and provides a common language for expressing user interaction with a website.

Keywords

log format; user logging; usability testing; web-based applications

1. Introduction

This paper describes a format called FLUD, Framework for Logging Usability Data, for representing user interaction with a website. A common file format enables sharing of logged data and the development of interoperable tools. The logging of user behavior in support of web usability testing is constrained by the difficulty of capturing and analyzing large amounts of logged data. However, there is great potential for the development of tools to support automated recording and analysis, especially for remote or large scale testing. The few tools that currently exist do not interoperate and so cannot exchange data easily. Researchers cannot easily share large data sets of user logs for further post-mortem analysis and exploration. The file format that we propose here is designed to address these problems. It can be viewed as a browser-independent language for expressing user interaction with a website. Given such a language, one can then formally describe what is being collected during a usability test.

In Section 2 we provide the background that led to the development of of FLUD. Section 3 summarizes related work. The FLUD design and accompanying tools we developed for its use are presented in Sections 4 and 5. In Section 6 we list a number of issues that are factors in the adoption of FLUD by the usability community.

2. Background and Motivation

2.1 Why FLUD?

The Visualization and Usability Group, which is part of the Information Access Division of NIST's Information Technology Laboratory has been working on measurement, testing, and standards relating to usability engineering since 1997. In particular, the NIST Web Metrics Testbed has focused on automation to support the usability engineering process which has resulted in the development of prototype tools that are publically available to the usability community.

One focus of this effort is the recording of the users' interaction with a web-based application as they attempt to perform given tasks. This captured log data can be valuable for analyzing and improving usability [Etgen00]. As we developed our recording tool WebVIP, [WebVIP] we found that we did not have a formal specification for the user logs we wanted to collect. Initially, our output was closely tied to the specific event model of the browser we were working with. It soon became evident that log data is quite complex and that a common file format was needed to allow various software components (such as recorders, parsers, analyzers, and visualizers) to exchange information. At that point, we focused part of our effort on FLUD.

2.2 Approaches to Usability Testing

While there are some purely automated approaches to usability evaluation (but see [Niel99] for a skeptical view), the automation provided by most of the NIST Web Metrics prototypes supports the usability engineer during the course of *usability testing,* in which subjects are asked to perform some tasks.

Usability testing encompasses a range of approaches:

- Direct human observation of the subject by a usability engineer, who records and interprets the subject's behavior. This approach has great semantic depth, since all the subject's external behavior is available for analysis by an intelligent observer; indeed, the subject's thought processes may be queried as well. The process is, however, time-consuming and there seems to be only slight opportunity for automation.
- High-level automated monitoring. The software under test can be instrumented by hand so as to report application-specific performance metrics, such as a score indicating degree of success in achieving a task and time taken. This approach allows a larger number of subjects to participate in the test, but is less helpful in analyzing *why* subjects succeed or fail.
- Automated monitoring of low-level user behavior. This can often be done with the help of existing software, e.g. by capturing events as reported by a browser. The problem here is that the higher-level, more meaningful description of the subjects' behavior is lost.

Our efforts are aimed at combining the best aspects of the last two approaches: we want to automatically generate a mid-level description of behavior such that we can achieve breadth (a large number of subjects), but also enable some computer-assisted analysis of *why* individual subjects performed as they did.

3. Other Related Work on Log Files

W3C has promulgated a Common Log Format [W3Log] for server logs, which record requests for web pages from external clients. These logs are compiled automatically as a byproduct of running a web server, but at best contain information only about page jumps made by a user. No intra-page activity is captured. Moreover, it can be difficult to distinguish the activity of a particular user among requests from several sources and the use of cache storage by a browser may hide repeat requests. In short, a server log tracks the activity of a server, not a subject. It is noteworthy, however, that several analysis and visualization tools for server logs [Analog, Flash, Hoch99, Hoch00, LogAn, PWeb, Webal] have emerged, encouraged no doubt by the existence of this *common* log format.

There are several event models, such as those supported by popular browsers, e.g. Netscape [Netsc] and Internet Explorer [MSIE], the widely used Xwindow system [Nye93], and the Document Object Model proposed by W3C [DOM]. Because they represent user activity at a low level (e.g. mouse clicks and keystrokes), they are application-independent. They do not, however, attempt to represent user behavior at a higher level of abstraction, such as task performance.

Hilbert and Redmiles [Hilb98, Hilb99] have developed a prototype system for gathering information on user activity, but have not proposed a format for that information. Finally, there has been some work by Fu [Fu01] whose goal is to compile low-level information on user activity into higher-level abstractions.

4. FLUD Design

The FLUD (Framework for Logging Usability Data) format is intended to provide a representation of user interaction that is general enough to support a wide range of usability testing. The complete specification [Cugi01] of FLUD's syntax and semantics is available on-line.

4.1 Requirements and Scope

In light of the general goals outlined above, the FLUD format is designed to satisfy the following set of specific requirements and constraints:

- The format should be machine readable and writable, but also somewhat human-readable as well.
- It should support a database approach to user tests, by identifying various dimensions typical of usability log data: subject, website, dataset, task, and date, among others.
- It should define the operations *typically* encountered by a user of web-based applications. Concepts such as mouse clicks, keystrokes, radio buttons, scrolling, and jumping to a new page must all be included.
- It should also encompass the *context* of interaction, including window and browser operations as well as activity internal to the webpage.
- However, it will exclude *non-interactive* user behavior such as eye gaze, oral reports, and forehead-slapping for the obvious reason that in order to capture such behavior, one needs special equipment beyond the usual keyboard, mouse, and display. Also, the formal representation of such behavior is more problematic.
- Also, it will not define (at least for now) more exotic interaction modes, such as voice-activated commands or gaze-controlled cursor location.
- However, FLUD should be extensible; it should allow for the representation of unforeseen types of activity (although such extensions cannot be meaningfully processed by *generic* tools). For instance, if an application makes use of an input device or a widget of a type not pre-defined by the FLUD specification, grammatical hooks are provided so that the subject's interaction can nonetheless be represented and parsed, as so-called *ad hoc* fields. Of course, any analysis that depended on the semantics of such fields could be performed only by suitably customized software.

4.2 Top-level FLUD File Concepts and Definitions

4.2.1 Session - A FLUD file records exactly one session. A session is defined as the interaction of a single subject with a single fully configured hardware system during a continuous time interval. A switch of platform or subject is therefore considered to be a new session, by definition. Within a session, a subject may visit several websites and webpages and attempt to perform several tasks.

4.2.2 Task - The FLUD file format is designed for task-oriented usability testing: the subject is given a task to perform (e.g. find at least three documents about Iowa, find out how much a Boeing 747 weighs) and then his/her performance is monitored. Undirected browsing can also be recorded within a single "dummy" task.

4.2.3 Events - An event is defined to be a nearly instantaneous occurrence involving the subject, the system under test, or both. The event model is described in detail below. FLUD events are not quite the same as those within current, well-known, low-level event models. Therefore, a generator may need to map from their output to the FLUD level of abstraction. For instance, FLUD includes higher-level information such as webpage navigation, as well as typical low-level events (mouse, keyboard, widget).

4.2.4 Questionnaire - A FLUD file can represent the results of a portion of the session wherein the subject responds to a questionnaire set up by the tester. The difference between a questionnaire and a task is that a questionnaire requests information directly from the subject (e.g. "How old are you?", "Do you think the graphics are helpful or annoying?"), whereas a task is usually meant to simulate the intended usage of the website and the subject is monitored to find out such things as whether most people use the website effectively. Also, only the results of the questionnaire are reported, not the process by which they were answered (e.g. timing of the responses is not reported).

4.2.5 Notes - A note record captures information typed in during the session by the subject or tester or some other author. The idea is that test session manager software might provide a facility for observations, comments, complaints, or recommendations by interested parties if they encounter some unusual situation. This could be invoked at the initiative of the note's author or prompted by the system.

4.2.6 Conformance - The FLUD specification [Cugi01] defines conformance, that is, what constitutes a valid FLUD file. Conformance breaks down into syntactic and semantic requirements.

Syntactically, the file is described in a context-free grammar with a few context-sensitive constraints. The file is a sequence of records, each of which is a sequence of fields. Each field has a name (explicit or implicit), a value, and a type which defines the range of those values. The specification attempts to define fields so as to cover most common types of user interaction. Some of these defined fields are required to be present, some are optional. In additional, the file may contain so-called *ad hoc* fields, which are not defined -- this is the mechanism for extensibilty.

Semantically, the basic requirement is that the file truly reflect a subject's behavior: if the file says the subject pressed mouse button #2 at a given time, then that must really have happened. It is *not* required that the file capture all of the subject's interactive behavior, even if that behavior is representable. Producing a complete record of behavior is probably beyond the capability of most generators. Furthermore, the generator may deliberately omit reporting some kinds of activity, e.g. mouse motion. In short, FLUD requires the truth, and nothing but the truth, but not the whole truth.

4.3 Basic Event Model

Events are occurrences of short duration that are apparent to the user and involve the system under test. By this definition, we exclude operations of long duration (which may, however, be represented as a sequence of several events), purely internal changes to the system, and non-interactive user behavior. After some analysis, we decided that an event could have up to three distinct aspects, as listed below. The FLUD syntax marker is shown next to each component.

User_action (#U)
: An action performed directly by the user and associated with a particular input device, typically a mouse or a keyboard.

This_widget (#W)
: Describes state changes in the widget, if any, to which the user_action was targeted. Screen objects, such as buttons, textboxes, menus, checkboxes, and sliders, are typical widgets.

System_effect
: System_effect is used to describe "everything else"; in particular, how the state of the system as seen by the user changes (either as a result of the user_action or autonomously). System_effects are further sub-divided into three categories:

  Other_widget (#OW)
  : State changes in any widget, other than this_widget.

  Window_state (#WN)
  : Includes typical window operations, such as open, close, move, re-size, and iconify.

  Webpage_operation (#OP)
  - print: usually available as a browser operation
  - newpage: operations involving a new webpage
    - request, loading, complete
  - page_locate: indicates which part of the webpage is visible within a window

An event record can contain any combination of these components. While there can be at most one user_action and this_widget component, there may be several system_effects. An example would be a user clicking on a "Clear" button: the mouse-click is the user_action, the triggering of Clear button is the this_widget aspect, but there may be several other system_effects such as textboxes cleared to the null string, checkboxes set to "off", and windows closed. When several components share an event record, it means that they are *causally* related, not merely contemporaneous.

4.4 Example

Click to see an example of a complete FLUD file.

5. Tools

If the FLUD format is to be more than an academic exercise, it must be supported by a set of software tools. The potential advantage of course is that widespread adoption of a common format enables these tools to be generic and sharable.

5.1 Generators

A FLUD generator is any software that monitors user and system behavior during a test session, and produces a FLUD file that accurately represents (some of) that behavior. Thus, the more information about the session available to the generator, the better. An ideal generator would know about not only low-level events, but also the broader computing context (browser and window operations), and the application (task metrics). Two implementation strategies suggest themselves:

5.1.1 Instrumentation of the website - There are at least two extant systems, WebVIP [WebVIP] and WET [Etgen99] that semi-automatically instrument the pages of a website so as to report user activity thereon. This approach makes sense when the focus is on design and review of a particular website, as opposed to study of user behavior on the web in general. Obviously, user activity outside of the instrumented site is not captured. WebVIP has recently been modified to generate its output in the FLUD format.

Website instrumentation supports remote testing and does not require a special browser. Webpages can be further customized by hand to incorporate task knowledge. However, there are some implementation difficulties with delivering the log data back to the website owner. In particular, the generator must either transmit data back to the server for every event or find storage on the client side where data can be buffered and managed. Another problem is the issue of privacy: instrumentation opens the door to the possibility of extensive covert tracking of user activity.

5.1.2 Instrumentation of the browser - At least one study [Choo00] has been done to track the way users navigate the web, using special software called WebTracker that traces *browser* activity. This technique enables the researcher to follow a subject's travel throughout the entire web, not just a chosen website. This approach implies installation overhead for each subject, rather than per website. In theory, since *all* of a subject's interaction with the web is through a browser, an instrumented browser potentially offers a more complete trace of user activity than instrumentation of a webpage. This approach, however, seems less oriented towards the incorporation of task or application knowledge. Customizing a website, by contrast, may more easily allow automatic reporting of task metrics.

5.2 Parser

We have developed a FLUD parser to support the format. It checks the syntax of a logfile and can generate three kinds of output as a result:

Parse file
: This is a highly stylized rendition of the original logfile. Its intended purpose is to serve as input to other automated processes, such as analyzers and visualizers (see below). Instead of processing the logfile directly, they can invoke the parser to perform low-level syntax checking. The resulting parse file can then be easily analyzed for higher-level purposes, such as statistical summaries and the like.

HTML file
: This file is essentially a pretty-print version of the logfile. Indentation and color are used to clarify the file structure. Thus, this file is primarily oriented towards human review. If syntax errors are found, an error message will be inserted in the HTML file.

Userpath files
: These files are designed as input to the NIST VisVIP software, [VisVIP] which presents a 3D visualization of a user's navigation of a website. Each task within the session generates a separate file. Each file represents the web pages visited by the subject during a single task, and the length of time spent on each page.

5.3 Post-processors

The real payoff for the FLUD format would be a large varied suite of software tools available to the usability engineer for analysis (statistical and otherwise) and visualization of user behavior on the web. As mentioned above, VisVIP [VisVIP] is one such tool, but we at NIST are planning to develop others and encourage the community of usability experts to contribute as well.

6. Open Issues

6.1 Scope

FLUD is targeted at a level of representation appropriate for usability analysis and evaluation. We hope to get feedback from usability preofessionals as to how well this goal has been achieved. In particular, are some of the entities defined in FLUD of little interest? Conversely, does FLUD fail to provide a representation for certain valuable kinds of entities and user behavior?

6.2 Feasibility

How feasible is it to build sophisticated generators of FLUD files? Are there special difficulties representing user interaction with dynamically generated pages? What problems are presented by the need to map between FLUD and other event models? Finally, our experience with WebVIP has revealed some low-level issues with the website instrumentation approach, such as events being reported back to the Javascript code out of chronological sequence, and inadequate mechanisms for transmitting information back to the webserver.

6.3 Acceptance

Even given a specification of good technical merit, it will have little value unless it is adopted among a wide enough circle of users. The factors governing acceptance of a standard for exchanging information include ease of use, perceived technical benefit, and, of course, recursively, acceptance by others with whom one wishes to communicate. The first two factors, at least, can be addressed by a suite of software tools that are readily available and confer some advantage on their users.

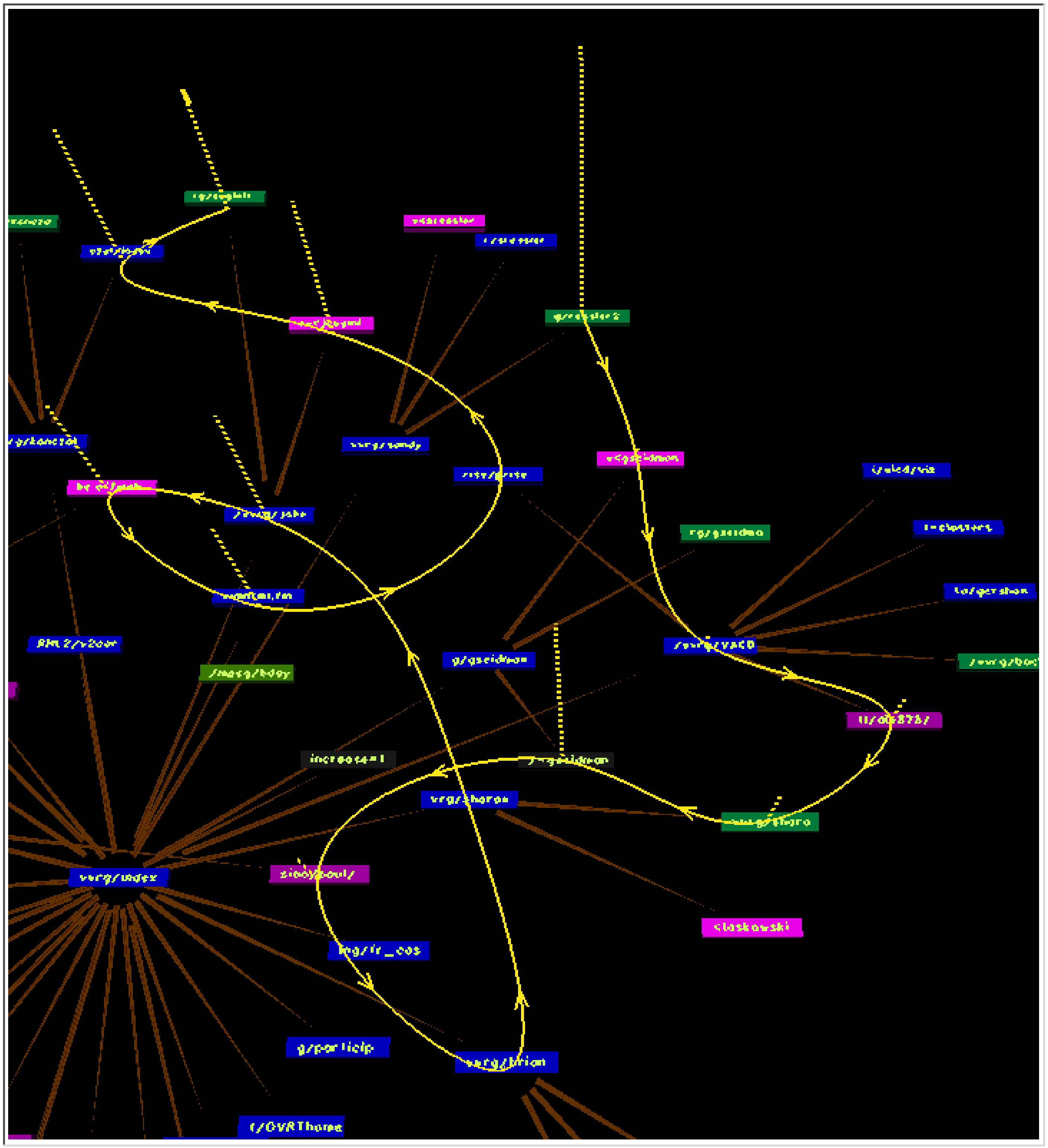

Figure 2: VisVIP displays a userpath through a website

References

[Analog]
Analog (server log analyzer): http://www.analog.cx/

[Choo00]

Chun Wei Choo, Brian Detlor, Don Turnbull, "Information Seeking on the Web - An Integrated Model of Browsing and Searching", *First Monday,* Volume 5 no. 2, Feb 7, 2000: http://choo.fis.utoronto.ca/FIS/SSHRC/ and http://firstmonday.org/issues/issue5_2/choo/index.html

[Cugi01]
J. Cugini, "The FLUD format: Logging Usability Data from Web-based Applications", NIST Special Publication 500-247, January 2001: http://www.itl.nist.gov/iad/vug/cugini/webmet/flud/specification.html

[DOM]
The DOM (Document Object Model) Level 2 Event Model: http://www.w3.org/TR/DOM-Level-2-Events/events.html

[Etgen99]
M.P. Etgen, J. Cantor, "What does getting WET (Web Event-logging Tool) Mean for Web Usability?", *Proceedings of the 5th Conference on Human Factors and the Web,* Gaithersburg, MD, June 1999: http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html

[Etgen00]
M.P. Etgen, J. Cantor, "A Comparison of Two Usability Testing Methods: Formal Usability Testing and Automated Usability Logging", Proceedings of UPA 2000, Asheville, North Carolina, August 14-18, 2000.

[Flash]
FlashStats (server log analyzer): http://www.maximized.com/products/flashstats/

[Fu01]
W.-T. Fu (in press), "ACT-PRO: Action protocol tracer -- a tool for analyzing simple, rule-based tasks", Behavior Research Methods, Instruments, & Computers.

[Hilb98]
D.M. Hilbert and D.F. Redmiles, "Agents for Collecting Application Usage data Over the Internet", *Proceedings of the Second International Conference on Autonomous Agents,* Minneapolis/St. Paul, MN, ACM, May 10-13, 1998.

[Hilb99]
D.M. Hilbert and D.F. Redmiles, "Extracting Usability Information from User Interface Events", Technical Report UCI-ICS-99-40, Department of Information and Computer Science, University of California, Irvine.

[Hoch99]
H. Hochheiser and B. Shneiderman, "Using Interactive Visualizations of WWW Log Data to Characterize Access Patterns and Inform Site Design", *ASIS'99 Proceedings of the 62nd Annual Meeting of the American Society for Information Science,* October 31-November 4, 1999, Vol. 36, 331-344.

[Hoch00]
H. Hochheiser and B. Shneiderman, "Coordinating Overviews and Detail Views of WWW Log Data", Tech report 200-25, Human-Computer Interaction Lab (HCIL) at the University of Maryland, October 2000.

[MSIE]
The Internet Explorer Event Model: http://msdn.microsoft.com/workshop/author/om/event_model.asp or http://www.webreference.com/js/column10/

[LogAn]
HTTPD Log Analyzers (list of server log analyzers): http://www.hypernews.org/HyperNews/get/www/log-analyzers.html

[Netsc]
The Netscape Navigator Event Model: http://www.webreference.com/js/column9/

[Niel99]
J. Nielsen, "Voodoo Usability", Jakob Nielsen's Alertbox, December 12, 1999: http://www.useit.com/alertbox/991212.html

[Nye93]
A. Nye, Xlib Reference Manual, O'Reilly & Associates, 1993.

[PWeb]
pwebstats (server log analyzer): http://martin.gleeson.com/pwebstats/

[VisVIP]
VisVIP (visualization of user paths through websites): http://www.itl.nist.gov/iad/vug/cugini/webmet/visvip/vv-home.html

[W3Log]
"Logging Control In W3C httpd": http://www.w3.org/Daemon/User/Config/Logging.html

[Webal]
Webalizer (server log analyzer): http://webalizer.dexa.org/

[WebVIP]
WebVIP (website instrumenter): http://www.nist.gov/webmetrics/