Usability Testing of World Wide Web Sites

Michael D. Levi and Frederick G. Conrad

Background

Building a medium or large World Wide Web site, whether for distribution over the Internet or over an intranet, can and should be viewed as a major software development effort. As in any other software development project, a central role must be played by task experts -- in this case staff familiar with the content being presented. Staff versed in traditional publishing can also contribute a great deal. But designing a structure sufficient to hold hundreds or thousands of static documents and possibly scores of embedded applications, not to speak of configuring and maintaining a Web server, building and maintaining a document repository, keeping hypertext links up to date, or writing and maintaining Common Gateway Interface (CGI) scripts and Java or ActiveX applets, is what systems analysts are trained to do.

Once Web site creation is seen as software development, it becomes natural to apply the tools and methods we have learned in past projects. The life cycle of Web creation is identical to that of traditional software: requirements gathering, analysis, design, implementation, testing, and deployment. And, just as traditional software development should have a functionality and a usability component, so should Web development efforts.

Usability can be defined as the degree to which a given piece of software assists the person sitting at the keyboard to accomplish a task, as opposed to becoming an additional impediment to such accomplishment. The broad goal of usable systems is often assessed using several criteria:

- Ease of learning
- Retention of learning over time
- Speed of task completion
- Error rate
- Subjective user satisfaction

Methodologies for building usable systems have been introduced and refined over the past fifteen or so years under the discipline of Human-Computer Interaction (HCI). HCI principles include an early and consistent focus on end users and their tasks, empirical measurements of system usage, and iterative development. Much effort has been put into exploring cognitive models of human behavior as it relates to computer usage, and developing guidelines for screen layout and system dialogues. These are predictive endeavors whose purpose is to assist the software developer in the initial task analysis and system design.

But, just as comprehensive functional requirements and a detailed design document do not by themselves guarantee that a programmer's final code will be correct, so up-front usability guidelines do not by themselves guarantee a usable end product. In both cases a distinct validation process is required.

Usability testing is the process by which the human-computer interaction characteristics of a system are measured, and weaknesses are identified for correction. Such testing can range from rigorously structured to highly informal, from quite expensive to virtually free, and from time-consuming to quick. While the amount of improvement is related to the effort invested in usability testing, all of these approaches lead to better systems.

Many large organizations have invested heavily in fully equipped usability labs staffed by experienced professionals. Companies such as Apple and Microsoft routinely subject new software to a battery of sophisticated tests. Usability testing need not involve a laboratory, however, nor need it be expensive nor require an army of human factors experts. Our experience at the Bureau of Labor Statistics (BLS) indicates that extremely useful usability testing can be performed reasonably easily, reasonably quickly, and for almost no cost other than staff time. The bottom line is that virtually any kind of usability test -- as long as its results are fed back to the development group and acted on -- will improve the product. Usability testing, like most methodological process improvements, is likely to gain attention and devotees as its benefits emerge through use.

Of particular interest should be the potential cost savings that can be realized through usability engineering. The purpose of most Web sites is to attract users and distribute information or products. Losing users because of a poor design could be catastrophic for a commercial venture. Even in the absence of direct monetary considerations, an organization may find the cost of user support -- such as calls or e-mail to a help desk -- is directly related to a site's ease of use.

There are three main styles of testing. Exploratory testing examines a system and looks for areas of user confusion, slow-down, or mistakes. Such testing is performed with no particular preconceived notions about where the problems lie or what form they may take. The deliverable for an exploratory test is a list of problem areas for further examination: "users were visibly confused when faced with page *x*; only half the users were able to complete task *y*; task *z* takes longer than it should." Exploratory testing can be used at any point in the development life cycle, but is most effective when implemented early and often.

Threshold testing measures the performance characteristics of a system against predetermined goals. This is a pass/fail effort: "with this system users were able to complete task *x* in *y* seconds, making an average of *z* mistakes. This does (does not) meet the release criteria." Threshold testing typically accompanies a beta release.

Finally, comparison testing measures the usability characteristics of two approaches or designs to determine which better suits users' needs. This is usually done at the early prototyping stage.

Usability testing can be performed with developers, HCI experts, or representative end users. Some authors distinguish between "testing", which they limit to empirical end-user oriented methods, and "evaluation", which utilizes HCI professionals' expertise. In what follows we shall use "testing" broadly to describe all methods of assessing or measuring system usability, regardless of participant population.

This article describes a set of usability testing techniques the authors have employed over the past 18 months at the Bureau of Labor Statistics to evaluate the BLS public access Web site (http://stats.bls.gov) and a joint BLS-Bureau of the Census site for the Current Population Survey. In the course of this work we have mined the HCI literature for information and modified many of the methods to apply to Web systems. Our work has been conducted largely for release over the Internet, but the techniques we describe are equally relevant to intranet testing. Organizing distributed documents and applications, regardless of the network used to transport them, requires the same design considerations.

Card Sorting

While many usability testing techniques provide feedback on the details of individual pages or sequences, few analyze more global questions of organization and structure. The card sorting technique gives such a broad overview.

A group of end users is given a set of randomly ordered index cards, each of which is labeled with a concept from the task domain such as "Consumer Price Index news release" or "Top 20 requested time series for Employment and Earnings". Users are instructed to

1) Scatter all the index cards on your desk.

2) Sort the index cards into small piles according to similarity, and place a rubber band around each small pile.

3) Arrange the small piles into larger groups that appear to belong to an overall category, and then place a large rubber band around each group.

4) Invent a name for each of the larger groupings, write it on a slip of paper, and attach each slip to the corresponding group.

We have performed a card sorting study with remote users, handling all interactions through the mail, and this has worked quite smoothly.

Several statistical packages have a cluster analysis procedure which can take the data the card sort generated and break it into cross-subject clusters (we used the Statistical Package for the Social Sciences, SPSS). If a relatively small sample of users (10 -15) is utilized, then simply eyeballing the returns gives much the same insight as the cluster analysis will.

[Sidebar #1: Hierarchical Cluster Analysis for 26 Domain Concepts]

The card sort technique is more commonly used as a design tool for building menu trees. In our case, however, we compared the card sort results to our draft Web structural design, and identified several areas where we could improve the underlying hierarchy so that users could more easily find the information they were looking for.

Many variants of this technique exist. During a session at Software Development '91 Larry Constantine described a group exercise using sticky notes. Other practitioners use slightly different approaches. What all have in common is the goal of partitioning a large information space into manageable subsections that reflect the intuitive expectations and mental models of the user base.

One advantage a Web-based system has over a more traditional software package is that the Web is hypertext based. More than one hierarchy can be imposed upon a site, and links between hierarchies can be constructed. If several distinct organizational models emerge from the card sort, many or all can be accommodated. The risk, of course, is that too large a proliferation of cross-links can be just as confusing and frustrating as too few links. Discipline is still required, and the card sort technique is a good way to narrow in on a fruitful subset of all possibilities.

We have called upon internal help desk staff and information officers to identify and contact potential end users for us. In another type of organization the marketing or sales departments might be able to solicit participation. We have been gratified to discover how many people are willing to assist in such effort -- perhaps because they stand to gain if better systems are developed for them.

Heuristic Evaluation

Heuristic evaluation consists of HCI experts exploring a system, identifying usability problems, and classifying each problem found as a violation of one or more usability principles, or heuristics. In order to prepare for such an evaluation session, the testers need to assemble two documents. The first is a project overview, describing the objectives, target audiences, and expected usage patterns of the system being tested. The second is a list of heuristics.

[Sidebar #2: Heuristics]

The testers meet with the evaluators as a group for about 45 minutes to explain the purpose of the sessions, preview the process, present the project overview and heuristics, and answer any questions. Any special training that might be required is conducted at this session.

The individual evaluation sessions tend to last sixty to ninety minutes each. The evaluators can either be instructed to browse through the system concentrating on the sample usage patterns provided or can be given concrete tasks to accomplish. In either case the evaluators identify potential usability problems, and tie each problem found to the specific heuristic it violated. Multiple heuristics can be linked to any given problem. The testers record data from all sessions.

After all the individual sessions have been completed, the group meets as a whole for about ninety minutes. During this meeting, facilitated by the testers, each evaluator presents the violations she/he found along with the heuristic that was violated; and a composite list is assembled. Design suggestions for improving the problematic aspects of the system are also discussed at this time.

After the debriefing session, the testers format the composite list of violations as a rating form, and send it (in our case via e-mail) to each evaluator. Evaluators are requested to assign severity ratings to each violation on a five-point scale, ranging from "This is not a problem" to "This is a usability catastrophe.". The evaluators' severity ratings are sent back to the testers, and the individual lists aggregated and analyzed.

[Sidebar #3: Sample output from Heuristic Evaluation]

We have run several heuristic evaluations, and found that they identify specific usability problems such as inconsistent titles and labels, unintelligible jargon, and confusing layout. None of our heuristic evaluations identified system-wide structural problems. Since the card sorting procedure does just that, we find that a card sort complements the heuristic evaluation nicely.

Jakob Nielsen [1994] describes seven other inspection methods in addition to heuristic evaluation. These include cognitive walkthroughs, guideline reviews, pluralistic walkthroughs, consistency inspections, standards inspections, formal usability inspections, and features inspections. What all have in common is having HCI experts, rather than end users, go though a structured process to identify usability weaknesses.

Finding HCI experts to act as evaluators can be a challenge. One possibility is to hire outside consultants. We were able to identify knowledgeable staff within our agency, however, who reported enjoying the experience. We also ran one heuristic evaluation using the system developers as evaluators, and found, to our surprise, that their feedback was very comparable to that of the HCI experts (see the authors' article in Useful Resources below for more detailed information).

Scenario-Based Testing

A scenario-based usability test involves presenting representative end-users with scenarios, or specific tasks, designed to cover the major functionality of the software system and to simulate expected real-life usage patterns. Such scenarios should be formulated by knowledgeable task experts in consultation with the system designers. Results are then tabulated using such measures as whether the participants correctly accomplished the tasks, the time taken for each task, and the number of pages accessed for each task.

[Sidebar #4: Sample scenarios]

It is important to keep in mind, and stress to all participants, that it is not the participants' abilities that are being tested. Rather, it is the system's ability to accommodate the participants that is under evaluation. This is a critical distinction that lies at the very heart of all usability engineering.

Task-based evaluation with end-users as participants is what is commonly referred to as usability testing, and each practitioner has a tool kit of somewhat different techniques. The approach we have used is as follows:

Five to ten participants are given access to the Web site a few days or a week in advance of the test, and are encouraged to browse a little each day. We request an e-mail note briefly describing the participants' experiences each afternoon; this is mainly to ensure that participants do, indeed, visit the site in advance. Compliance with this request has been spotty.

Then the testers meet with the full group of participants for 30 minutes to describe the system in general terms, give an overview of the process, and answer any questions.

Participants are seated in front of a desktop computer and asked to work through the scenario questions, writing their answers on a form we provide. Sessions last approximately one and a half hours. During this time testers are available to assist participants if they get stuck, but such assistance is recorded as a task failure.

At the end of the session a full group discussion is held to gauge the participants' subjective reactions and solicit suggestions for improvement. Our experience is that the conversations between participants during such debriefings are frequently more instructive than the direct participant-tester dialogue.

Web server logs maintain an audit trail of each page visited by each user (identified by IP address of the client machine), and thus provide the testers with a time-stamped record of each participant's session. Unfortunately, the logs that are currently produced are less substantive than they might be. Sessions are not necessarily recorded in their entirety. A known feature of WWW logs is that pages are cached on the client workstation; thus, if a user goes back and forth repeatedly between two pages, only the first access of each page is likely to be logged. In addition, neither CGI scripts nor applets are recorded with all the detail that one would need to effectively trace a session. Despite these shortcomings, the logs provide an excellent approximation of users' journeys through a site.

We are still working on developing a sound quantitative comparison of the users' actual paths versus a predetermined 'ideal' or expected path. In the mean time, we have found that visual inspection of the logs works reasonably well, as long as the number of sessions to be inspected is not too large. The time and number of pages tabulations help testers focus on problem sessions.

[Sidebar #5: Sample report from Scenarios]

Recently, one group within BLS has been experimenting with a PC-to-VCR hookup that captures the screen, mouse movements, and the users' conversation during the tests. The video tape can then be reviewed after the testing session to get a clearer picture of the users' actions. In the words of one project leader: "[what] is hard to describe has to do with 'seeing' the application from the users' point of view, almost like being in their heads. You can get this perspective from observing actual use, but it's hard because you're probably thinking like a developer and mentally urging the user to do the right thing instead of thinking like the users and wondering along with them what the right thing is. Somehow, observing via the tape makes it easier to see like a user." This supplements the hard performance data nicely.

One unanticipated benefit of the video tape is that its' immediacy is very convincing to the developers, who may otherwise dismiss usability test results. A drawback of the video is that it does not allow sessions to be run in parallel, since we can not afford multiple scan connectors.

The great advantage of empirical end-user testing is that the results are incontrovertible. Unlike heuristic evaluation, where HCI experts speculate as to what may cause users difficulties, an end-user test highlights where users actually do have difficulties. It remains up to the testers, however, to interpret the results and determine what caused the problems. End-user testing lends itself very well to an iterative test/fix/retest cycle.

Questionnaire for User Interaction Satisfaction

In addition to evaluating 'hard' measures like task speed and error rates, it is extremely useful to investigate the less quantifiable aspects of interface design that cumulatively (and often subtly) contribute to users' subjective feelings of satisfaction or frustration. The clearest system in the world does no good if users avoid it because they find it annoying.

To this end the authors have employed the Questionnaire for User Interaction Satisfaction (QUIS), developed by the Human-Computer Interaction Laboratory at the University of Maryland. The QUIS is not a perfect survey instrument, but it is as close to an industry standard as exists in the discipline of Human Computer Interaction. Designed to provide reliable and consistent cross-platform and cross-application satisfaction measures, the QUIS does not specifically address Web technology.

The current instrument asks participants about:

- Their demographic background
- Their overall reactions
- The features of individual screens (characters, layout, sequences and moving between screens
- Terminology and system information (system status, instructions, error messages, etc.)
- Learning to use the system
- System capabilities (speed, reliability, and error correction facilities).

All of the questions require the participants to circle a scale value ranging from 1 to 9 to indicate their satisfaction. The scales are constructed so that a value of 1 indicates maximum dissatisfaction and a value of 9 indicates maximum satisfaction. Every section also has space for free-form comments.

We have modified the QUIS slightly, eliminating some irrelevant questions and adding questions that are particular to hypermedia applications such as Web sites. We have taken care to make the smallest number of changes possible, so as not to introduce language bias (the phrasing of a question often influences the answer) or inadvertent redundancy.

The QUIS is best administered immediately after a user has interacted with the system being tested.

Currently in version 5.5, the QUIS is available from the University of Maryland. A Web-enabled version has been promised for the not-too-distant future.

Mining the Logs

Usability evaluation need not end with a system's release. Standard Web server, or httpd, logs are an invaluable source of information about usage patterns once a Web site has gone live. At this point the testers need not find usability experts or representative users; real users' sessions are captured in great detail and are available for analysis.

Some examples:

The BLS Web site has an ad-hoc database query function which is implemented through a sequence of CGI-generated forms. An analysis of the logs shows that a disappointing percentage of users never complete the full sequence, and thus never receive the data we believe they are looking for. This has led to a reexamination of the extract application.

The logs give full details of every access to the Wide Area Indexing Service (WAIS) search on the BLS site, including each search string entered. We separate the user sessions which begin with a search from those sessions where the search comes only after many pages have been accessed. The latter category represent, in our minds, a failure of the site's organization -- users can not find what they are looking for by traversing the hyperlinks, and so fall back on the WAIS engine. When the logs show consistent patterns of this nature, it is time to rethink the page hierarchy.

Finally, the logs can be easily summarized to display the most popular pages on the site. These pages should be readily accessible with a small number of clicks.

The advantage of using httpd logs is that they capture real users going about their tasks. The weakness of using these logs is that the users' goals can usually only be guessed (though search strings may provide strong clues), and there is typically no way to query the users as to what they really were looking for.

Special Challenges of the Web

Web-based systems have both significant commonalties and significant divergences from other software systems, which must be taken into account when performing usability testing. The particular challenges of Web development include:

A highly diverse user population which is non-trivial to predict or measure. This makes finding a "representative" set of test participants difficult.

A highly diverse set of end-user computer configurations, including hardware, systems software, and browsers. Ideally, usability testing will be performed from multiple client machines using multiple browsers. In practice, this geometrically increases the number of required test machines and test participants, and is usually not feasible.

A wide disparity in connectivity speed and bandwidth. Again, in an ideal world testers would have enough client hardware and test participants available to cover the possible permutations. Again, this is typically not feasible.

A deployment environment which gives the illusion of being much more powerful than it actually is. Since most browsers run in a windowed environment, and most Web pages include graphics, different size fonts, etc., the inexperienced user is misled into expecting the full functionality of a graphical user interface application. Java and ActiveX applets may move the capabilities of a Web sites closer to such expectations, but testers must expect and account for user disappointment.

A deployment environment that blurs the distinction between the site content and the browser used to access this content. Test participants frequently comment on deficiencies in the browser being used, and do not understand the distinction between browser and Web site. Though this may be helpful in developing an understanding of users dissatisfactions, the site designer typically has no control over browser development.

For some of these features an intranet may be less of a challenge than the Internet: a particular organization may have a small number of standard end-user hardware and software configurations; all users within an organization may well be connected to the LAN in the same way (though geographically distant users logging in to a WAN may still have markedly different connectivity capabilities); the population of an intranet is far easier to identify than that of the Internet.

Nonetheless, the fundamental challenge remains: how to identify usability shortcomings before releasing a new system (or in the early stages of a redesign), when changes can still be made relatively cheaply.

Conclusion

A comprehensive World Wide Web site may well become one of the major points of contact between a given organization and its user base. For many users this system will be the only grounds on which they can judge the organization. Thousands, or even hundreds of thousands, of users may be obtaining mission-critical data from this source. Ease of learning, ease of use, and general user satisfaction, along with quality and comprehensiveness of content and functional capabilities, will determine the success or failure of the effort.

There are myriad methods for usability testing. We have hardly exhausted the list of possible methods, and continue to research and experiment with new techniques. Our experience to date has been extremely positive. We have, in fact, identified ways of examining and improving the usability of our Web sites before releasing them. The methods we have employed are reasonably easy, reasonably fast, and reasonably cheap. Best of all, they are unintimidating for both participants and testers.

There is no question in our minds that our systems are better because of the usability testing we have performed, and that the end users have benefited in direct, measurable ways.

Some Useful Resources

Usability Inspection Methods. Jakob Nielsen and Robert Mack, eds. 1994 John Wiley and Sons, Inc.

A Heuristic Evaluation of a World Wide Web Prototype. Michael Levi and Frederick Conrad. July/August 1996 *interactions Magazine* (a publication of the Association for Computing Machinery).

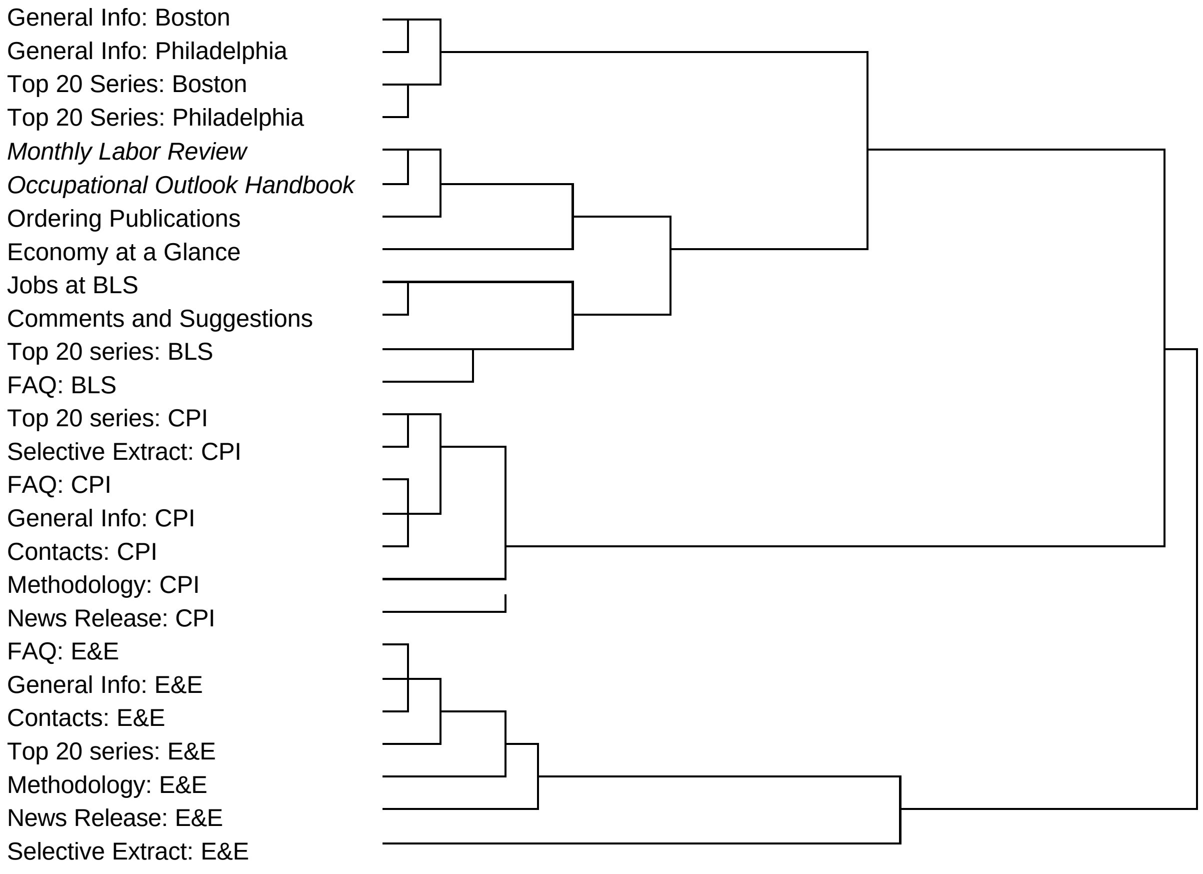
Handbook of Usability Testing. Jeffrey Rubin. 1994 John Wiley and Sons, Inc.

A Practical Guide to Usability Testing. Joseph Dumas and Janice Redish. 1994 Ablex Publishing Corp.

Interface Design for Sun's WWW Site. Jakob Nielsen. http://www.sun.com/sun-on-net/uidesign/.

Questionnaire for User Interaction Satisfaction Home Page. University of Maryland at College Park. http://lap.umd.edu/QUISFolder/quisHome.html.

Sidebar #1: Hierarchical Cluster Analysis for 26 Domain Concepts



Sidebar #2: Usability Principles (Heuristics) Tailored to Web Systems

1. **Speak the users' language**. Use words, phrases, and concepts familiar to the user. Present information in a natural and logical order.

2. **Be Consistent**. Indicate similar concepts through identical terminology and graphics. Adhere to uniform conventions for layout, formatting, typefaces, labeling, etc.

3. **Minimize the users' memory load.** Take advantage of recognition rather than recall. Do not force users to remember key information across documents.

4. **Build flexible and efficient systems**. Accommodate a range of user sophistication and diverse user goals. Provide instructions where useful. Lay out screens so that frequently accessed information is easily found.

5. **Design aesthetic and minimalist systems**. Create visually pleasing displays. Eliminate information which is irrelevant or distracting.

6. **Use chunking**. Write material so that documents are short and contain exactly one topic. Do not force the user to access multiple documents to complete a single thought.

7. **Provide progressive levels of detail**. Organize information hierarchically, with more general information appearing before more specific detail. Encourage the user to delve as deeply as needed, but to stop whenever sufficient information has been received.

8. **Give navigational feedback**. Facilitate jumping between related topics. Allow the user to determine her/his current position in the document structure. Make it easy to return to an initial state.

9. **Don't lie to the user.** Eliminate erroneous or misleading links. Do not refer to missing information.

Sidebar #3: Sample Output from a Heuristic Evaluation

Heuristic Violated	**Location in System (*)**	**Usability Problem**	**Severity**
1	BLS Surveys	Need better list order, e.g. alphabetize	4
6	LABSTAT	Alphabetize items by name, not abbreviation	4
6	BLS Surveys	List needs to be grouped better/weird granularity	3.75
1	LABSTAT	Abbreviations (AP, BG, etc.) are not explained	3.5
2	BLS Surveys	“Return to Home Page” button is missing	3.5
4	LABSTAT	Consequences of incomplete or incorrect parameters are not clear	3.5
2	BLS Home Page	Inconsistent labeling of buttons and bulleted text	3.25
1	BLS Overview	“Mission Statement” is jargon	2.75
5	BLS Surveys	Bullets don’t add anything	2

Note:
Refers to section, or sub-tree, in Web site where problem was found

Sidebar #4: Sample Scenarios from Test on the Current Population Survey Site

1) What is the poverty rate for high school dropouts 25 years or older? How is it different for blacks, whites, and Hispanics? Please give the name of the page on which you found the answer.

2) What page would you use to compute the standard error of the estimates retrieved in scenario #1?

3) Are there more 24 year olds in a professional specialty, or 20 year olds in administrative support (including clerical)? How many of each are there?

4) What major change to the Basic Monthly Survey took place in May, 1955? Please give the name of the page on which you found the answer.

5) Send the following message to the person or group responsible for maintaining this Web site:

"<Your name> was visiting at <current time> on April 18, from IP Address <IP Address>"

What is the e-mail address to which you sent the message?

Sidebar #5: Sample Report from Scenarios

Table 1: Accuracy on scenarios (from written worksheets)

Participant	Scenario #1	Scenario #2	Scenario #3	Scenario #4	Scenario #5
1	Correct	Correct	**FAILURE**	Correct	Correct
2	Correct	**FAILURE**	**PARTIAL**	Correct	Correct
3	**PARTIAL**	Correct	**PARTIAL**	Correct	Correct
4	Correct	Correct	Correct	Correct	Correct

Table 2. Time per scenario, in minutes (from log files)

Participant	Scenario #1	Scenario #2	Scenario #3	Scenario #4
1	1.25	7		.25
2	22.5		6	1
3		11	10	.25
4	1.5	10	7.5	.25

Note:
Blanks represent portions of sessions that could not be reconstructed.

Table 3. Number of pages accessed per scenario (from log files)

Participant	Scenario #1	Scenario #2	Scenario #3	Scenario #4
1	5	14		3
2	38		15	4
3		16	18	3
4	6	8	14	3

Note:
Blanks represent portions of sessions that could not be reconstructed.