Usability Testing Web Sites At the Bureau of Labor Statistics

Transcript from:

National Institute of Standards and Technology Symposium
Usability Engineering 2: Measurement and Methods (UE2)
March 3, 1997

Speaker:

Michael D. Levi
U.S. Bureau of Labor Statistics
Levi_m@bls.gov

Good morning. I'm Michael Levi from the Bureau of Labor Statistics (BLS). I'm going to be talking to you about usability testing Web sites.

My talk is based on the experiences several colleagues and I have had working primarily on three Web sites. The first is the BLS public access Web site, where all of our historical data, including information about the data, is available to anyone with a Web browser. The second site is a joint BLS/Bureau of the Census site for the Current Population Survey. The third is the beginning of a BLS intranet, where BLS is trying to bring Web technology inside our fire wall to coordinate within and between the various program areas.

Why Test Usability?

Now the first question is: "Why do we care, why do we want to test usability?" or the broader question: "Why are we concerned about usability in general within our products?" One answer is that a small group of us just thinks this is kind of cool stuff, and we have an opportunity to mess around, so we do. That may be more true than I'd be willing to acknowledge.

We have, however, also built a business case for usability within the organization. First of all there is the volume of requests. Some 80,000 to 100,000 users every month come to the BLS public access Web site to get our economic data. It makes sense to make user interaction with the system as straight forward and as efficient as possible. It certainly saves us money in terms of hardware, software and support costs. If we can get by with a slightly less powerful machine because the interface is good enough that people can get in, get what they want and get out, then there's a clear dollar savings there.

Beyond reducing the load on the hardware, like every other agency, the BLS current mandate is to do more with less. One of the ways that some groups have identified saving money is to try and redirect at least some of the data requests from the telephone and from letters to the Web site, people being more expensive than the Web server. In order to do this the Web site has to at least approximate the ease of use associated with interacting with a human being. Obviously we will never reach that, and we're not saying that we're going to eliminate a human being at the other end of the telephone, but there's actually a very large set of standard requests that people come to the Bureau and ask for, for instance, this month's consumer price index top level number. Scores of thousands of people want this information every month; if we can make it easy for them to get it by themselves, we have saved ourselves a lot of staff time. Users are going to go to the system that is easiest and cheapest for them. If the Web site is too cumbersome and annoying, they're not going to stop making phone calls. If the Web site approaches the convenience of a human, on the other hand, they may start using it as their primary information gathering tool. So if we want to move standard requests from phone calls that take a lot of staff to the Web, we'd better build a pretty good Web site.

Again related to user support costs, the more usable the system is the less people would be calling and saying "Well I found your Web site, but I can't find the top level CPI number, how do I do it" or sending e-mail or whatever. So the idea is the more usable access is, the less burden on the Help Desk; again, presumably, there is some dollar costs savings.

Finally, and on some level the most important, the various Web sites we're putting up have become the public face of the Bureau of Labor Statistics. There are many thousands of people whose only interaction with BLS is the Web site. More and more people know about us only through our Web site. They never have talked to a staff member or made a personal visit, they may not have seen any of our publications. As far as they're concerned, BLS is this Web site. Simple pride in ourselves and our product mandates that we put our best face forward, that we make the public's interaction with our agency as straightforward, as easy, as pleasant an experience as possible.

This is pretty much the business case that we've made for taking usability seriously as far as the Web is concerned.

Testing as Part of a Process

Now I'm going to be talking about testing, but want to put this in context first. Testing is only one part of the process. You can't "test in" usability. If you have not done any HCI work before the testing phase, you've pretty much already failed. But if you don't do the testing we also think your chances of failure are higher.

When [the previous speaker: Steve Cross, Director of the Software Engineering Institute] talks about the software engineering approach to software development and how that's merging with the HCI approach to software development, I agree with him 100%. I think there's a usability development life cycle just like there is a standard software development life cycle and it follows much the same stages and typically can be done more or less in parallel.

Like software engineering, an HCI development process starts with user task and requirements analysis, goes through a design phase, and ends with a testing stage. As Steve Cross said earlier, we have moved away from the waterfall metaphor where one phase flows neatly into the next phase, and processing keeps going from phase to phase, never backing up, until you're done, and then you sign off. Instead we are looking at a circular process where you go through requirements, analysis, design, implementation, testing and then you do it again and you do it again and you do it again until you get it right (or until you're close enough).

Now the first time I heard about an iterative development life cycle it made sense to me and I thought "I can do that". Having tried it a couple of times there are two dangers to look out for that I've come across. First is that you really do need to iterate. It is not uncommon to get a release out and then be assigned to the next project and have your staff taken away because they also have another project. During project planning you have every intention of iterating the process, but you never actually get to do it, so what goes out is essentially a prototype or early version of a piece of software. This early version never gets revised and the process breaks down.

The other danger is that if the first release you put out is no good, then it will take a long time for your user base to regain any confidence in the product. Regardless of how often you say "We're putting something out for you to look at and to start using and we will incorporate your feedback rapidly and improve the product", if the first release is not good enough people may never come. So it is important that the first release be pretty decent or at least good enough so that the user community is willing to use it for a while and to give you feedback and to look for the second release. That way you can iterate problems from a position of strength and then you move forward, but if the first iteration doesn't work, you're in big trouble.

Testing Site Structure

In my mind, there are two significant aspects of Web site design: site structure (What is the organization of the site as a whole? What is the dialogue between the user and the site? How do you navigate through?) and individual page design (How does the page look? Where do the individual elements go? What typeface is used? How many graphics on a page? What are the standard elements on each page?) These two design modes can be, and I think need to be, approached somewhat differently as far as analysis, design, and also testing is concerned.

Starting with the site structure, at BLS we have fairly successfully implemented three testing techniques. I'm not going to go into minute detail as to how all these works. Instead I'll refer you to a paper that Fred Conrad and I wrote. This paper is a cook book approach to most of the techniques I'll be talking about during the remainder of my presentation.

The idea in evaluating site structure is to determine from the users how they partition the information space, what is their mental model of the information you will be providing in your Web site. We have used a fairly standard card sort for this. You write down a number of leaf pages, like "Consumer Price Index News Release" or "Employment and Earnings Statistical Methodology" or "BLS Contacts" on index cards, one title per card. Give the stack of cards to a group of users and ask them to sort these into meaningful piles and put a label on each pile. You aggregate all your users together (we use the SPSS hierarchical cluster analysis function for that, but often simply eyeballing the results is sufficient) and you can come up with a pretty good hierarchy of users' expectations when they come into a site.

Another technique, which we call a "Category Membership Expectations Test" works the other way. You say here are six or eight category names, what sort of information would you expect to find within each category? Card sort is from the bottom up, category membership is from the top down.

Icon recognition is where you produce a number of different possible icons or possible graphics for portions of the site and you ask the user to match an icon with a category. You are looking for high recognition and low interference (interference being where one icon is identified with more than one category.)

These techniques are usually identified as analysis and design tools, and, in fact, I believe that is when they are best employed. But we found that they also work pretty work as evaluation tools to validate a design after it has been completed. This is especially useful when a Web site has been built without much up-front analysis or design (and certainly many sites just grow without careful planning). This kind of structural test can be very illuminating in determining whether the site creators successfully met their users' expectations.

Testing Site Pages

As far as page design is concerned, the best place to start is with a style guide that is given to content providers before they begin work. There are a lot of good references out on the Web now. I like the Yale style guide in particular, but there are a growing number of quite decent guides. The goal is to get consistent, readable, legible, comprehensible pages and make it as easy as possible for a reader to find information on any given page (or recognize that the information does not exist on a given page).

The technique that we've used to evaluate page level design is a heuristic evaluation. It's an inspection technique. You're not actually going to end users, but instead you're having usability experts go through a site and analyze it. There are a number of inspection methods. Jakob Nielsen has written a book called "Usability Inspection Methods", where he catalogs quite a few of them and discusses some of the advantages and drawbacks to the different methods. We found that a heuristic evaluation really does a very good job at identifying things like inconsistencies between pages, inconsistencies between the words you use as a link and the title of the page that you're jumping to, differences in layout, jargon, incomprehensible acronyms and things of this nature.

Testing Site Usage

Now it's all fine and well to have a style guide and to follow it, and it would be nice if we could say "Well if you follow the style guide then you will have a perfectly usable site", but it often doesn't work quite as cleanly as that. There is certainly no substitute for having actual end users run through the tasks that they actually will do on the site and to pay attention to them. That's what we call Scenario-Based Testing. This is where you assemble a group of (hopefully) representative end users, you give them a set of tasks that reflect what users will be doing with the site once it goes up, and then analyze their performance. Some of the things to look for are how quickly users are able to accomplish the tasks, how many errors they made, how many intermediate pages they went through. Did they follow the path that you expected them to follow? Did they find a more efficient path or conversely did they find a much less efficient path?

There are various ways of capturing the interesting information: talk aloud protocols, the Web logs, video taped sessions. The idea is to observe and record what real users are doing on your system.

Then there's the issue of subjective user satisfaction. Regardless of how quickly a user gets through the task and how accurate they are, the question is whether they liked the system, whether they enjoyed the experience or whether they found it terribly frustrating and annoying. One tool that we have used to approach this is the Questionnaire on User Interface Satisfaction (QUIS), which was developed by the University of Maryland and can be licensed from the University of Maryland. I recently heard about a similar questionnaire called SUMI, developed and used in Europe. I hope to learn more about that. In fact I think one of the speakers later today knows a little more about it than I do and maybe we can ask him.

Post-Implementation Testing

I believe that there's no reason why usability testing should stop after a system has been deployed. One of the nice things about the Web is that the Web daemon does capture a reasonably comprehensive log of usage and there's an incredible wealth of information as to how real users are using the system in a production environment. Let's say BLS has a million hits a month. There's a million plus entries in the log; there's a lot of things that we can learn from that. Find out what users are searching for, find out where they appear to give up in the middle of a session, find out what the most popular pages really are, things of this nature. Of course, there is also direct user feedback to the Help Desk, telephone calls, e-mails, etc., all of which we try to gather and feed back to the developers, feed back to the design team for the next iteration of the system.

Special Characteristics of the Web

A lot of this is very similar to usability testing in the other system. On some levels, a Web site is just another software system and all the techniques that I've described were originally developed for non-Web systems.

We believe, however, there are some real differences between a Web site and a Windows or Text based application. There are some characteristics I believe that make the Web unique and we have been working within BLS to fine tune some of the testing techniques specifically for Web sites and for some of the specific characteristics of the Web.

The user population interacting with a Web site is typically much more diverse in terms of goals, experience level, domain expertise, and also system configurations (what hardware platform they're using, what operating system, what browser). A Web site seen through a windowed browser gives an illusion of power insofar as the site looks like a windowed system, and so naïve users are deluded into expecting the kind of interactivity that they get from their Windows word processor, but they don't get it because the Web doesn't actually support much in the way of real interactivity. That's changing a little now with Java, ActiveX, some of the other more powerful languages, but it will be a long time, if ever, before the Web has the same immediate capabilities that a desktop use of software does. Finally, naive users are not always clear as to what is a feature (or shortcoming) of the site and what is a feature (or shortcoming) of the browser being used.

The Organizational Context

To conclude, I wanted to talk a little bit about the organizational context. Certainly our experience at BLS is that there has been a gradual evolution or capabilities. We started, as an organization, seriously thinking about HCI seven or eight years ago. There was a long period of internal education where a number of the analysts trained themselves and started going to classes.

We then started bringing instructors in-house, developed a HCI curriculum, taught in-house for our analysts and started expecting a certain level of understanding. We developed a set of guidelines for interactive systems. Then usability testing kind of followed out of all of that. There has been a progression, and usability testing is the latest stage in our progression.

Given the realities of resource constraints, we had to look for techniques that were safe to implement. For me personally, there's been an interesting evolution from being a developer and code jockey frustrated by management not letting me do the usability analysis and testing that was clearly necessary, to being a manager who has to deal with budgets and delivery dates and things of this nature. And the fact of the matter is, there is a real deep faith involved here. It doesn't matter how many times someone tells me that if I do the up front work it'll cost me less in the long run, at some point I have to take a deep breath and say, "OK, I'll give it a shot, I am going to invest X person-weeks and Y dollars in this effort and hope that it will all work out for the best at the end". So to minimize my risk what I'm looking for is "easy, fast and cheap". I'm looking for techniques that do not require Ph.D. level human factors expertise, because I don't have it and I'm not going to be able to hire it; techniques that are reasonably fast so if things go wrong I haven't delayed delivery too much; and techniques that don't cost me much because I don't have much to spend.

And in fact all the methods that I've talked about so far are all three of them, they're easy, fast and cheap. In spite of that they have made a real difference in our systems. Having done a lot of these we are now slowly beginning to invest in what we're calling a usability lab. We've got a little more equipment so that we can do some slightly more sophisticated things. I expect if our efforts continue to be successful, we will get marginal increases in a non-existent budget, and be able to get some of the nifty and more expensive tools, but the fact of the matter is we've come a long way without spending much money and it's made all the difference in terms of moving forward. I think that's pretty much it. Are there any questions?

Questions and Answers:

Q. Do you have a formula for costing a usability test on a Web site?

Levi. No we really don't. Our experience is that the only dollar cost is really in staff time and depending on the task it can take anywhere from one person-week to three or four person-weeks, depending a lot on the expertise of the staff, the kind of tests and how thorough we want to be.

Q. Have you found a good log analysis package?

Levi. Not yet and in fact that's one of the things that we are most interested in finding at some point. What we have done is hacked together a collection of Perl scripts that will go through the logs and pull out some of the things we're interested in. I've heard recently about some log analysts tools that are coming out of Europe. Apparently there is at least one piece of software that has been developed specifically for this purpose, but I don't know anything about it yet. I'm just hoping that it's going to be something valuable.

In particular what we don't have a good handle on is measuring deviation from an expected or ideal path. At least in the designer's mind there is an ideal path from one point to the next. That's the expected, fairly efficient way of getting from point A to point B. Now what we find is that users frequently do not follow that path. They wander around. What we'd like to be able to do is to compare the user's actual path to the presumed ideal path and make some deductions. Where do they diverge? What is the average number of steps? What can we expect in terms of use? I've found nothing so far that supports such investigation. If anyone knows of anything, please let me know; I'd be most grateful.

Q. In reference to what you said about iterating and the first release needing to be pretty good: I've seen examples where that's really not the case. Where the first one is really atrocious but if you have strong management support for this iterative process you can recover. Without management support, you're right, you're dead.

Levi. I think management is actually likely to recognize a dismal failure and say, "You've got to do it again." The problem is the user base. If you have users who must use the system, they're going to keep coming back anyway because they don't have any choice. No matter how horrible it is. There may be high costs for the agency but the users have to use the system because it's their job. If, on the other hand, you have discretionary users who get to decide, "Am I going to use this system or not, am I going to use the Web site as opposed to making a phone call?" or something like that, they may not come back for a second try. In fact they probably won't, until you persuade them that it is worth their while. It's that initial credibility that makes all the difference, that determines whether they will give you another chance

Q. Are you set up to do usability testing on other software products or do you just concentrate on the Web site?

Levi. To date, we have done most of the testing on Web sites just because that is where the interest of people were. But we also have started applying various usability tests on non-Web systems and find the testing extremely successful.

Q. Some friends were over for dinner the other night and mentioned being frustrated with one of the sites you mentioned, the Census Bureau site. At work my friend has Windows 3.1, and on the basic Census site she can get only 2/3 of the way through the screens and then it would stop. I knew it was supposed to work, so we came to my house and tried it on Windows '95 and got all the way through. I didn't notice a good, simple way to provide user feedback.

Levi. I think that one of the things that's hard about testing Web sites is that ideally you'd run from all standard user configurations. Unfortunately there are too many possibilities. If you have ten standard scenarios and you break your user population into three major groups, that's 30 major tests right there. Now multiply by all the different browsers and configurations -- there's no way you can do it, it's impossible. All you can do is try to pick the most important and hope for the best.

Q. I was browsing through one site and they said their site worked best with Netscape 2, and they downloaded a copy of Netscape 2 for you to run.

Levi. A lot of sites do that, but a lot of users don't like it, because they've already customized their browser. If they were to accept a new download it might write over what existed.

Q. I've seen situations where the site provides a button that the user can click on and everything is taken care of.

Levi. But you still have the same situation. There is an arrogance here. The site designer knows best, even as far as what resides on the users hard disk.

Q. I have a comment and a question. The comment is sometimes if I'm stuck in the browser, particularly if my PC has hung, I can't send a message to the webmaster or anybody else, and it's very frustrating because I want to get certain document, I really want to get this and I don't know how to get it because I can't make it download and I can't send E-mail to get help either.

My question is: What sort of people do you have designing your Web sites? In my organization just about anybody feels competent to do Web design.

Levi. That seems to be one of the special characteristics of the Web. Most applications software at this point is developed by professional software developers. Most Web sites are not developed by professional Web developers. If you're lucky you've got a central group that reviews pages as they come in, but a lot of organizations don't even have that level of control. It's pretty much anyone who wants to put anything out can, and that makes for highly unusable systems.

Certainly it's unlikely that you will achieve consistency under those circumstances. Everybody is likely to solve the same problem in their own way. One way to address it is to at least try to route everything through one group who checks for adherence to standards. Be very clear up front what you're looking for, what you and will not accept and why.

Q. That central group you talk about, did they do editorial review?

Levi. At BLS there are two independent groups. One does editorial, or content, review. The other does Web standards review.

Q. What other functions might a centralized group perform?

Levi. Build common functionality, common graphics, establish the common graphic design.

Q. And you mentioned the central feedback, so mail to the site comes to one place then gets spread out?

Levi. We have always given users two options. The nice thing about the Webmaster is it's easy. The user can remember it, or at least there's some chance that the user will remember Webmaster @ bls.gov or HelpDesk @ bls.gov, so even if they end up stuck somewhere and have to get out of their browser then they would fire up their e-mail and probably reach somebody. On the other hand the Web Master cannot answer real questions about content. If someone wants to know why the CPI minus food and energy is X% this month, when blah, blah, blah, the Web master is not going to be able to answer that question and will have to forward it to the right person. Best case, there is a loss of time because it takes a while for the Web master to read the mail, to forward it to the right person, and that person to read it; there's an extra step. Worse case is that it will never get sent to the right person.

So we've also tried to have feedback to the content expert who can actually answer content questions. We try very hard to explain the difference between a content question and a system question, and my guess is about 60% of our users understand what we're trying to tell them.