intranet librarian

Darlene Fichter
Northern Lights Internet Solutions, Ltd.

Heuristic and Cognitive Walk-Through Evaluations

Usability inspection methods and walk-through techniques are excellent supplements to empirical testing in the lab or field.

Intranet teams must continually evaluate, adapt, and modify their sites. As more features and applications are added to the intranet, the user interface inevitably becomes more complex. To keep pace with the new projects and tools coming downstream, intranet teams need a range of tools to effectively identify usability problems and remedy them.

Empirical testing with real users is the best way to evaluate user interfaces. However, sometimes it is difficult to recruit enough real users to test all aspects of design. Usability inspection methods and walk-through techniques are excellent supplements to empirical testing in the lab or field, working hand in hand with empirical testing.

There are a number of different usability inspection methods, including heuristic evaluation, cognitive walk-throughs, pluralistic walk-throughs, feature inspection, consistency inspection, and standards inspection. Two techniques worth adding to your toolkit are heuristic evaluations and team cognitive walk-throughs.

HEURISTIC EVALUATION

A heuristic evaluation is an informal inspection method in which evaluators assess whether an interface complies with recognized usability principles or heuristics.

Jakob Neilsen [www.useit.com/papers/heuristic/heuristic_list.html] has identified 10 recommended heuristic principles:

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Error recovery
10. Help and documentation

Keith Instone Web [http://userexperience.org/uefiles/writings/heuristics.html] offers a useful overview for first time-evaluators on how to apply these principles to the Web. While Neilsen's list of 10 heuristics is commonly used, there are

➤

other lists available, such as OCLC's list of 14 principles [www.oclc.org/policies/usability/heuristic/set.htm].

How Heuristic Evaluation Works

Your evaluation can be structured or unstructured. In an unstructured evaluation, people find the problems as they occur. This allows for "free-form" discovery of the problems. An unstructured evaluation works best if the area under investigation is discrete. In a structured evaluation, evaluators are given a scenario, a particular task or list of questions to address during their evaluation. This ensures that a particular area or task is assessed.

Regardless of whether the evaluation is unstructured or structured, your evaluators are asked to review the interface individually and make note of problems as well as the heuristic violated. Each evaluator may assign a severity rating as they proceed through the problems or assign the rating at the end. They use a scale such as the following:

Rating scale

0 = Not a problem
- I don't agree that this is a usability problem at all

1 = Cosmetic problem only
- Need not be fixed unless extra time is available

2 = Minor usability problem
- Fix should be given low priority

3 = Major usability problem
- Important, so should be given high priority

4 = Usability catastrophe
- Imperative to fix this before release

Usually the rating of 0—"this is not a problem"—is assigned when the ratings from all evaluators are combined.

The severity is based on the extent and impact of the problem. Is it a common or rare problem? What is the impact of the problem—is it easy or difficult to overcome? Is this a one-time problem, or will the user be repeatedly bothered by this problem? For example, some commercial database vendors require users to "select a database" before starting a search, even though a search box is clearly presented on the screen. Usually, people miss checking off a database name on their first attempt at searching, focusing only on the entry box. The user receives a pop-up error message. On subsequent searches, most users make the database selection before clicking on the search button.

Each evaluator reports an individual list of problems to the coordinator of the evaluation. The coordinator combines the problems and removes duplicates. Depending on your situation, you can have either the coordinator or the evaluators assign the severity ratings. The list of problems is then reviewed by the Web development team and solutions are recommended.

Heuristic Strengths and Weaknesses

Heuristic evaluation can quickly and effectively identify major and minor problems in an interface. With evaluations of academic and company Web sites, I have found this method to be particularly effective at identifying design inconsistencies and problems that impede the user's control and freedom. In some cases, problems may be identified in a heuristic evaluation that may not affect the user. These are considered "false positives."

Heuristic evaluation can be overwhelming due to the large number of potentially reportable problems. Ranking these problems by severity alleviates the stress. In a recent evaluation, 170 problems were identified in a working site, but less than 20 had a severity rating of three or higher. Many of the minor and major problems were "no brainers"—easy to fix. Others required a major overhaul of the architecture or navigation. In those cases, you would probably want to verify the problems with user testing and test mockups of new designs that attempt to remedy the problem.

This method works best when you have at least three to five evaluators. Provided you have a team of evaluators, a heuristic evaluation will usually uncover many of the major problems in an interface. A single expert evaluator will typically only discover about 35 percent of the problems. When evaluators are *both* usability specialists and domain experts, only two or three evaluators are needed to uncover 81-90 percent of the interface problems [1]. Don't be deterred from using this method if you don't have usability specialists. Heuristic evaluation is easy to learn and a team of five properly briefed novice evaluators can carry out an evaluation and discover approximately 50 percent of the problems. With 14 novice evaluators, about 75 percent of the site problems will be identified.

You can also use heuristic evaluation as the first step in assessing a Web site and planning for task-based testing. Problem areas identified in the evaluation could receive special attention in empirical task-based user testing.

An unstructured evaluation works best if the area under investigation is discrete.

COGNITIVE WALK-THROUGHS

A cognitive walk-through is another usability inspection method. It works best when performed by a team rather than a single evaluator. Usually your development team, with its designers, programmers, managers, and content developers, will walk through the design together.

The cognitive walk-through focuses on an aspect of usability, called "exploratory learning," that is crucial for successful intranets. The walk-through focuses on how easy the system is to learn. Is it confusing? Is anything missing? What must the user know to make the next correct step?

The cognitive walk-through method is based on Lewis and Polson's CE+ theory of exploratory learning. This theory identifies four steps in an information processing model of cognition (see table at right).

A cognitive walk-through can be used to evaluate a new design at an

early stage of development. You do not need a working prototype, although one is useful. A walk-through can be conducted based on a detailed description of the system that should give the location of items on the screen and the wording of key pages or areas.

Representative tasks are selected for the cognitive walk-through by the moderator. Each of those tasks is broken down into a series of steps along with a description of users, their experience, and prior knowledge.

The session is facilitated by a moderator. A recorder is also designated. Some teams choose to videotape the walk-through. In a detailed description about how to carry out a walk-through, Wharton, Reiman, Lewis, and Polson describe four criteria that need to be addressed:

1. Will the user be trying to achieve the right effect?
2. Will the user know that the correct action is available?
3. Will the user know that the correct action will achieve the desired effect?
4. Will the user interpret the system's response to the chosen action correctly?

As the task is presented step by step, the team is asked to craft a credible story explaining why the expected users would choose the correct action. If a credible story cannot be told, then suggestions for fixing the problem are noted. The stories can draw upon what the user must know prior to the task and what the user is learning while using the site. The recorder documents problems on the detailed task sheet, giving a description of the problem and noting its severity [3].

Step	Example
The user sets a goal.	Find the phone number of the head of HR.
The user searches the interface for currently available actions.	Menu item, link, search box.
The user selects the action that seems likely to make progress to the goal.	Menu label "Staff directory."
The user evaluates the response to the selected action to assess if progress is being made to the goal.	Skims the next page looking at title and content.

Strengths and Weaknesses

This method has a number of key advantages. The cognitive walk-through method is easily learned by software developers and other team members. Scheduling a walk-through is usually a simple undertaking, as development teams often meet on a regular basis. The fact that this method can be used to assess designs at the early stages of development makes it a very useful tool. The particular focus of the cognitive walk-throughs, learning a system by exploration, meshes well with the design goals of intranets, which are designed to for "walk up and use" without training.

Cognitive walk-throughs are also wonderful for generating lots of design ideas and cross-fertilization of ideas amongst team members with different backgrounds and perspectives. Some of the most useful insights in a recent walk-through came from a new team member who not only got up to speed quickly with the project, but identified several gaps in the working prototype as we proceeded step by step through the task. In fact, every team member became more knowledgeable about the whole system by participating in the walk-through.

The major drawback with cognitive walk-throughs is the fact that the process can be time consuming for assessing major tasks. Personal experience and various research studies document the time-consuming nature of the analysis and recording, as well as the tendency to get pulled off track into lengthy design discussions. Case studies like Rowley and Rhoades' proposed a "cognitive jog-through" as a way to speed up the process, but still maximize the information gleaned. They recommend using video recording sessions to lessen the reporting requirements.

Spencer described ways that the cognitive walk-through was adapted to work at Microsoft [4]. He identified the following key points: defuse design defensiveness up front, minimize design discussions, and streamline the method and data collection. To streamline the method, he reduced the four questions to two:

1. Will the user know what to do at this step?
2. If the users do the right thing, will they know that they did the right thing and are making progress to towards their goal?

PUTTING A COGNITIVE WALK-THROUGH INTO PRACTICE

Recently, I moderated a cognitive walk-through with a multidisciplinary team that had never used this method. We were assessing a working prototype of a scholarly portal prior to conducting task-based testing. The team walked through 13 user scenarios that identified a clear, information-seeking goal, describing each person and his or her past experience. To ensure the walk-through moved along quickly, only two questions were posed: Would the user notice/select the correct action? Rather than identifying just *one* correct action, we allowed for a range of correct options and estimated the percentage of users that might select a particular route, such as browse or search. We examined the specific tasks one by one and took note of problems.

The first session was a challenge. The pitfalls identified by Spencer and others were evident. The team wanted to solve every problem that they noted and, despite the moderator's efforts and ground rules at the beginning of the session, entered in lengthy design discussions. One of the programmers said at the end of the first session: "Don't make me do this. My job is to fix things; and when something is broken, it's in my nature to start talking about how to repair it." The recorders, inexperienced with the walk-through

➤

method, found it a challenge to note the relevant information and keep pace with the discussion.

With any new technique or method, a learning curve is expected. At the beginning of the next session, I reiterated the ground rules, emphasizing the need to just note design ideas and keep moving. I renamed the session a cognitive "run-through," emphasizing the need for speed. The second and third sessions moved along briskly, despite tackling some complex tasks. In retrospect, I would probably hold a practice session with a team new to this method following through a task like finding a book in a library catalog to allow them to practice and get comfortable with the method. Most of us don't expect to have the power to redesign the catalog interface and functionality, so it should be easier to keep the team focused on the job at hand and minimize lengthy design discussions.

WALKING THROUGH THE WEB

Some new research carried out by Blackmon, Polson, Kitajima, and Lewis has resulted in the development of Cognitive Walk-Through for the Web (CWW) [5]. CWW examines the degree of semantic similarity (information scent) between the user goal statements and heading/link texts on each page. Latent semantic analysis is used to measure the degree of similarity. While this technique is not something developers can easily use today, the research team notes that they are well on the way to building an AutoCWW that will be useful to developers.

A key adaptation of CWW is the assumption that generating an action, such as clicking on a link, is a two-step process. There's the attention phase, in which the user parses the page into subregions and focuses on the correct subregion, and the action-selection phase, in which the user makes a choice from the subregion. They note that problems arise due to three types of semantic problems.

1. Unfamiliar heading/link—User does not know what the heading or link label means.
2. Confusable heading/link—Can two or more headings look like they mean the same thing to the user?
3. Goal-specific competing heading/link—Two headings or links that are different, but equally similar to a possible goal.

Detecting problems early in the design cycle increases the chance that they will be corrected.

PICKING THE METHODOLOGY

Intranet teams need effective tools and techniques to get the job done. Under the pressure of time and budgets, the methods that you select must fit in with your culture and team. Heuristic evaluation and cognitive walk-throughs are two methods that you could consider using to identify usability problems. These two methods can be employed by your intranet team on their own or in conjunction, ideally, with empirical testing to help improve your intranet.

Detecting problems early in the design cycle increases the chance that the problems will be corrected. After months of development, making a major change in the interface can be like turning the Queen Mary around. Real-world constraints make it unlikely that it will happen. Cognitive walk-through methods can be used by teams to analyze early mockups. Heuristic evaluation can also be used to help identify problems early on by evaluating a working prototype.

One way that I have found cognitive walk-throughs and heuristic evaluation to be particularly helpful is to identify low-hanging fruit—problems that can easily be remedied prior to usability testing. Often the problems, once noted, are obvious and the team wonders how they were overlooked. Obvious interface problems of high severity and frequency can be identified and remedied promptly rather than jumping up repeatedly during user testing. The user testing can focus in on problems that are hard to assess in these methods—what real-world knowledge and expectations about how things work are brought by the user to the intranet or a particular application? Are we on track? Both usability inspection methods are excellent for flagging areas that warrant more testing and where ideas and assumptions that we have about the users of the system can be validated.

Darlene Fichter *[fichter@lights.com is president of Northern Lights Internet Solutions, Ltd.*

Comments? E-mail letters to the editor to marydee@xmission.com.

References

[1] Nielsen, Jakob, "Finding Usability Problems Through Heuristic Evaluation," *Proceedings of the ACM CHI '92* (May 1992), pp. 373-380.

[2] Lewis, Clayton, Peter G. Polson, Cathleen Wharton, and John Rieman, "Testing a Walkthrough Methodology for Theory-Based Design of Walk-Up-and-Use Interfaces," *Proceeding of the ACM CHI '90* (April 1990), pp. 234-242.

[3] Wharton, C. W., J. Reiman, C. Lewis, and P. Polson. "The Cognitive Walkthrough Method: A Practitioner's Guide," *Usability Inspection Methods*, J. K. Nielson, & R. L. Mack (eds.), Wiley, 1994.

[4] Spencer, Rick, "The Streamlined Cognitive Walkthrough Method, Working Around Social Constraints Encountered in a Software Development Company," *Proceedings of the ACM CHI 2000* (April 2000), pp. 353-359.

[5] Blackmon, M. H., P. Polson, M. Kitajima, and C. Lewis. "Cognitive Walkthrough for the Web," *Proceedings of CHI 2002* (April 2002), pp. 463-470.