An Integrated Method for Evaluating Interfaces

Heather L. McQuaid
David Bishop
MAYA Design
mcquaid@maya.com
bishop@maya.com

ABSTRACT

When we added heuristic evaluation to our expert analysis method, we found that it worked well in our interdisciplinary environment, but that its recommended prioritization strategy (which ranks problems according to severity) has some limitations. Specifically, it does not address how much it will cost the developers to fix the problems, nor does it adequately capture the distinction between high-level (global) and low-level (specific, screen-level) problems. To address these limitations, we developed a method that integrates user research, heuristic evaluation, affinity diagramming, cost-benefit charts, and recommendations into a report that others can use to plan both short and long-term improvements.

BACKGROUND

Our company is an interdisciplinary design consultancy that helps high-technology companies develop products that are functional, aesthetic and easy to use. We use a team-based approach to design [1], creating teams that include at least one member from each of our core disciplines, or departments: Human Sciences (Human Factors and Cognitive Psychology), Engineering (Software, Electrical, and Mechanical), and Design (Visual and Industrial). As part of our design process, we often evaluate existing products. One of the more cost-effective methods we use to evaluate products is an "expert analysis." Until several years ago, this analysis consisted of one or two members of the Human Sciences group carefully inspecting an interface looking for and commenting on both high-level and low-level problems. Low-level, or screen level, comments referred to the usability of buttons, dialog boxes, and other elements that appear on a single screen. High-level comments referred to the usability of the global interaction with the system, including problems with navigation and task sequence.

Although this method was fast and clients found the results valuable, it did not take full advantage of one of our key strengths—the interdisciplinary experience of our team members. We believed we could improve our expert analysis and make it even more valuable if we could find a team-based technique that would allow members of other disciplines to participate, but would still be cost-effective and well-integrated with our other methods.

Heuristic evaluation [6, 7]—a technique that allows usability specialists to use a list of heuristics, or guidelines, to evaluate products at virtually any point in the design process—met the criteria. However, we noticed a few limitations. One limitation is that the prioritization strategy did not address the cost factor, that is, how much it would cost the client in both time and effort to fix the problems. Additionally, the prioritization strategy did not adequately capture the distinction between high-level and low-level problems, a distinction that we had found so useful in our earlier expert analyses. To address these concerns, we took heuristic evaluation and combined it with other techniques to create a repeatable method that produces a clear and immediately executable plan of attack for our clients.

OUR METHOD

Our method consists of five steps:

1. Gathering domain knowledge.
2. Conducting the heuristic evaluation.
3. Categorizing the issues.
4. Prioritizing the categories according to how important it is to fix them (from the users' perspective) and how difficult it is to fix them (from the developers' perspective).
5. Writing the report, including recommendations for solving the problems.

Gathering domain knowledge

As with any evaluation or design work, it's imperative to understand the domain (i.e., users, tasks, environments, and level of training needed) in which the product will be used. To gather this knowledge we may interview the client, analyze competitor's products, and observe and interview users using techniques such as contextual inquiry [2] and task analysis [4, 5]. After gathering this information, we organize and analyze it using any number of techniques, including cultural models [2], information architectures [8], tasks flows [4, 5], user profiles, personas [3], and scenarios. We then use these analyses to provide context for evaluators as they conduct the heuristic evaluation.

Conducting the heuristic evaluation

After gaining an understanding of users and their tasks, we're ready to conduct the heuristic evaluation. In general, we follow the instructions for heuristic evaluation originally outlined by Nielsen [5] and Nielsen and Molich [7]. Into these instructions we incorporate user personas and task lists. Consequently, the booklet that we give each evaluator contains:

1. An overview of heuristic evaluation, including a list of heuristics.
2. A summary of the product (its purpose) and its intended audience (user groups or profiles).
3. An outline of the tasks or areas of the product that evaluators should focus on.
4. A description of the materials that evaluators will use to conduct the evaluation.
5. The procedure that the evaluators should follow.
6. A sample worksheet on which evaluators record where they discovered the problem, which heuristic was violated, and why the heuristic was violated.

We strive to assemble a team of evaluators that includes not only members of the Human Sciences Group, but also at least one member from our Engineering and Design Groups. We have found that including team members from other disciplines greatly enhances the quality of the evaluation as each person approaches the product with a different perspective and finds problems related to his or her core discipline. Moreover, if the people who evaluated the product are also the ones who will eventually redesign it, the heuristic evaluation gives them first-hand experience with the product and allows them to begin formulating solutions to the problems.

Categorizing the problems

Because the number of problems identified during a heuristic evaluation can be rather large (typically between 100 and 200 issues), we use affinity diagramming to group the problems into manageable categories (generally between 10 to 15). Affinity diagramming is a technique that allows people to organize a large number of issues into categories based on each issue's affinity with, or similarity to, other issues. This technique enables us to see patterns in the problems—which problems tend to group together and which seem unrelated. These patterns, in turn, give us a more integrated view of the problem space. Instead of focussing narrowly on each individual problem, we can expand our focus and get a high-level view of all the problem areas.

Having a higher-level perspective on the problems is incredibly valuable because it allows us to envision solutions not to each problem, but to whole categories of problems. For example, while evaluating an interface, one evaluator might comment that he had difficulty selecting items from cascading (fly-out) menus, another might have a problem understanding how the navigation areas at the top, left-hand side, and right-hand side of screen relate to one another, while another evaluator might have problems determining where she is in the system. Although we can, and do, recommend solutions to each of these individual problems, we can also see that they are symptoms of a larger problem, namely a poorly designed, or undesigned, information architecture [8].

Creating the affinity diagram

In general, we follow the steps for creating an affinity diagram outlined by Hackos and Redish [4]. Typically the evaluators have typed their comments into a spreadsheet which we then print out and slice into pieces such that there is one comment per piece. Occasionally the evaluators will write their comments directly onto Post-It notes. In either case, we instruct the evaluators to:

1. Pick up a comment and read it.
2. Examine the other comments on the whiteboard and place the comment within an existing group (if there are similarities) or elsewhere on the whiteboard (if it isn't similar to other comments).
3. Label and rearrange the groups along the way.

Prioritizing the problems

Once the problems are grouped into categories, each evaluator ranks the categories according to how important it is from the users' perspective to fix the problems in that group. Each evaluator votes on the four or five most problematic categories. Categories that receive the most votes are usually placed at the top of the "importance to fix" list. Eventually all the categories are placed on the list, with the most important categories at the top and the less important at the bottom. For example, we might create a list that has the categories ranked as follows:

1. Unstable system (crashes after Flash demo)
2. Competing navigation schemes
3. Violating users' expectations
4. No sense of place
5. Ambiguous cues
6. Misleading instructions
7. Inconsistent visual design
8. Confusing layout
9. Inappropriate marketing messages
10. Jargon

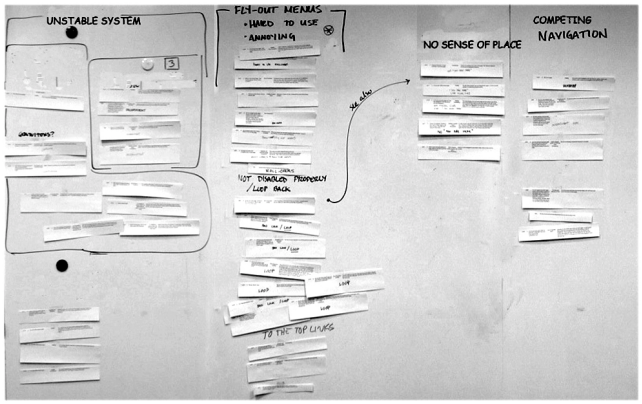

Figure 1: A section of an affinity diagram

The next step is to take the numbered list of categories and place them on a chart according to how difficult it would be, from the developers' and designers' perspective, to fix the problems in the categories. Because the evaluators have substantive experience in their core disciplines, they are able to make reasonable judgements about how difficult it would be fix problems related to their disciplines (e.g., an engineer can judge the effort required to fix the "unstable system" problem, whereas a visual designer can judge how difficult it is tackle the "inconsistent visual design" problem). Since the chart displays both the benefit to users of fixing the problems and the cost to the client of fixing the problems, we call it a Cost-Benefit Chart.

Cost-Benefit Chart

The figure on the next page is an example of a Cost-Benefit Chart. The x-axis (horizontal) represents Importance (or benefit)—how important it is to fix the categories from the users' perspective, that is, how much benefit users will experience if the problems are fixed. It is meant to be viewed as a continuum from lesser importance (!) to greater importance (!!).

The y-axis (vertical) represents Difficulty (or cost)—how much time, effort, and cost the client must expend to fix the problem. Likewise, it is meant to be viewed as a continuum from lesser difficulty ($) to greater difficulty ($$).
By mapping both Importance and Difficulty, the Cost-Benefit Chart essentially depicts the Return on Investment (ROI) associated with addressing each category of issues. To further emphasize the ROI principle, we have divided the chart into four cells, or quadrants, which we've labeled:

- High-value—contains very important issues that require less effort to fix (greatest ROI)
- Strategic—contains very important issues that require more effort to fix
- Targeted—contains less important issues that require less effort tofix
- Luxuries—contains less important issues that require more effort to fix (lowest ROI)

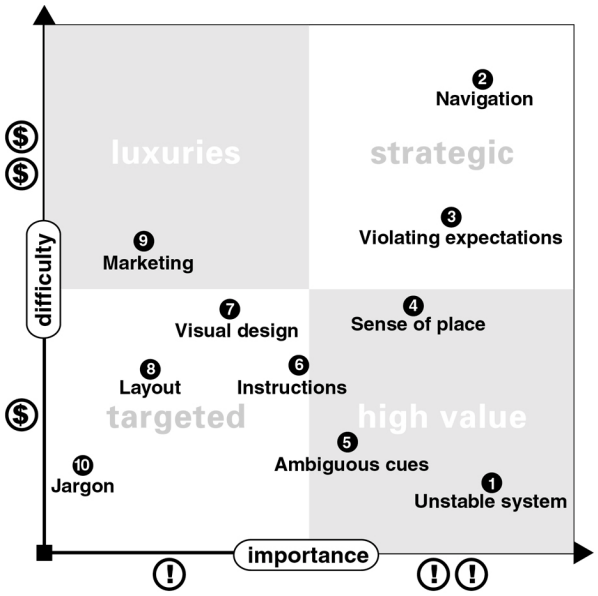

Figure 2: A Cost-Benefit Chart

Plan of attack

Because clients cannot solve all the problems simultaneously, we advise them to use the Cost-Benefit Chart to plan the order in which they should attack the problems. We recommend that they start with categories in the high-value quadrant. These categories represent the "low hanging fruit"—important problems that can be fixed relatively easily. In other words, these are the problems that, if fixed, would have the greatest ROI.

Next, we recommend that clients address the categories in the strategic and targeted quadrants. The problems in the strategic quadrant may require structured rethinking, or significant redesign of a product. For example, the problems in the "Competing navigation schemes" category may only be solved by creating solid information architecture, a process that requires careful planning and implementation. In contrast to strategic issues, targeted issues may have solutions that are easier to envision and implement. For example, the problems in the "Jargon" category can be solved relatively easily—jargon should be replaced with more common words or phrases. Finally, we recommend that clients address the categories in the luxuries category last, since they represent the lowest ROI.

It's important to note that the chart only suggests a plan of attack—the client's development team may want to move issues up or down the Difficulty axis depending on their insider knowledge of budgetary or technical constraints.

Creating the recommendations report

After we've categorized and prioritized the problems, we generate recommendations for fixing them. To help the client understand problems with specific screens, we often include a picture of the screen with our annotations overlaid.

In general the structure of our document is as follows:

1. Cost-Benefit Chart
2. Statement of the problem with each category.
3. Explanation of why it is a problem (i.e., why it compromises usability).
4. Recommended solution(s).

Why the report is valuable

The report often opens clients' eyes to significant problems with their system or product. Rather than giving them a long list of individual problems, which might lead clients to believe they have a lot of busy-work to do, we point out that many of the individual problems are actually symptoms of a larger problem. By focusing on the larger problems, clients are able to think about architectural solutions rather than temporary remedies. This is not to say that quick fixes aren't valuable; on the contrary, we encourage clients to fix the issues in the high-value quadrant before tackling the longer-term, more strategic issues. Our point is that the Cost-Benefit Chart gives clients a sense of direction and enables them to determine when they will fix specific problems and when they will work on more significant redesigns.

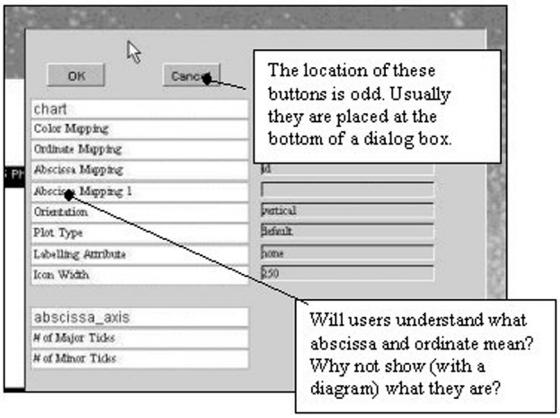

Figure 3: An example of a screen shot with annotations

Clients also find our document valuable because it takes a large number of problems and organizes them into digestible chunks, provides a rational way to prioritize clusters of problems, and not only presents the problems but also explains why they are problems. In addition, the annotated screen shots make the document more concrete and compelling.

The document also fits most organizational cultures. People concerned with the big picture can use the Cost-Benefit Chart to get a high-level view of the problems, while those responsible for fixing specific problems can skip to the relevant section of the document to read our recommendations and look at the annotated screen shots.

CONCLUSION

To take advantage of the interdisciplinary backgrounds of our team members, we adapted our expert analysis method to include heuristic evaluation. Even though we found heuristic evaluation to be a useful and efficient technique for evaluating interfaces, we noticed that it had several limitations: It did not address how costly it is for developers and designers to fix the problems identified, nor did it adequately capture the distinction between high-level (global) and low-level (specific, screen-level) problems. To overcome these limitations, we've developed a method that integrates heuristic evaluation with other techniques such as contextual inquiry, task analysis, affinity diagramming, and Cost-Benefit Charts.

Our method lets clients see the individual problems identified during the evaluation and enables them to see that many of these problems are symptoms of larger, more structural problems. It also gives clients a sense of direction and allows them to determine when to fix specific problems and when to tackle more strategic redesigns.

REFERENCES

1. Ballay, J. (1994). Designing Workscape: An Interdisciplinary Experience, in *Proceedings of CHI'94*, Boston, MA, April, ACM Press, 10-15.
2. Beyer, H., and Holtzblatt, K. (1998). *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers, Inc., San Francisco CA.
3. Cooper, A. (1999). *The Inmates are Running the Asylum*. SAMS, a division of Macmillan Computer Publishing, Indianapolis IN.

4. Hackos, J. T., and Redish, J. C. (1998). *User and Task Analysis for Interface Design*. John Wiley & Sons, Inc.
5. Lewis, C., and Rieman, J. (1993, 1994). *Task-centered User Interface Design: A Practical Introduction.* Available at: http://www.acm.org/~perlman/uidesign.html#chap0
6. Nielsen, J. Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.). (1994), *Usability Inspection Methods*. John Wiley & Sons, New York NY.
7. Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces, in *Proceedings of CHI'90*, Seattle WA, April, ACM Press, 249-256.
8. Rosenfeld, L., and Morville, P. (1998). *Information Architecture for the World Wide Web*. O'Reilly & Associates, Inc., Sebastopol CA.